HAL
open science

Specification et implementation d'une architecture de signalisation a gestion automatique de la QdS dans un environnement IP multi domaines

Guillaume Auriol

To cite this version:

Guillaume Auriol. Specification et implementation d'une architecture de signalisation a gestion automatique de la QdS dans un environnement IP multi domaines. Réseaux et télécommunications [cs.NI]. INSA de Toulouse, 2004. Français. NNT : . tel-00009244

HAL Id: tel-00009244

https://theses.hal.science/tel-00009244

Submitted on 12 May 2005

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

Préparée au

Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

En vue de l'obtention du

Doctorat de l'Institut National des Sciences Appliquées de Toulouse

École doctorale : *Informatique et Télécommunications*

Spécialité : *Réseaux et Télécommunications*

par

Guillaume AURIOL

SPÉCIFICATION ET IMPLÉMENTATION D'UNE ARCHITECTURE DE SIGNALISATION À GESTION AUTOMATIQUE DE LA QDS DANS UN ENVIRONNEMENT IP MULTI DOMAINES

Soutenue le 16 novembre 2004 devant le jury :

Rapporteurs	M. Andrzej DUDA
	M. Serge FDIDA
Président	M. Jean-Marie DILHAC
Examinateurs	M. Christophe CHASSOT
	M. Michel DIAZ
	M. Christian FRABOUL
Invités	M. Olivier DUGEON
	M. Laurent PLATEAUX

REMERCIEMENTS

Les travaux présentés dans ce mémoire ont été effectués au Laboratoire d'Analyse et d'Architecture des Systèmes du Centre National de la Recherche Scientifique (LAAS-CNRS), dirigé au cours de mon séjour successivement par MM. Jean Claude Laprie et Malik Ghallab, que je tiens à remercier cordialement pour leur accueil.

J'adresse également mes plus sincères remerciements à MM. Michel Diaz et Jean-Pierre Courtiat, directeurs de recherche au CNRS et responsables du groupe Outils et Logiciels pour la Communication, pour m'avoir accepté au sein de leur équipe.

Je suis également très reconnaissant pour le temps et le travail accordés par l'ensemble des membres du jury de ma thèse :

M. Jean-Marie Dilhac, professeur à l'INSA de Toulouse ;

M. Andrzej Duda, professeur à l'ENSIMAG ;

M. Olivier Dugeon, expert senior à France Télécom /DR&D ;

M. Serge Fdida, professeur à l'Université Pierre et Marie Curie ;

M. Christian Fraboul, professeur à l'ENSEEIHT ;

M. Laurent Plateaux, représentant DGA du Ministère de la Défense.

En particulier, je remercie MM A. Duda et S. Fdida d'avoir accepter d'être les rapporteurs de cette thèse.

Je tiens à remercier M. Christophe Chassot, mon directeur de thèse, qui m'a apporté bien plus qu'un encadrement scientifique. Il a su par son soutien me donner confiance et me pousser à m'améliorer. Il m'a offert un contexte de travail de très bonne qualité (en « sacrifiant » parfois même jusqu'à son propre matériel !!!) et qui m'a permis d'explorer le domaine des réseaux de télécommunication dans les meilleures conditions. Je lui suis également reconnaissant pour tous ses gestes (le temps et l'énergie dispensés !) d'encouragement.

Je remercie également M. André Lozes, partenaire de travail indéfectible de M. Christophe Chassot, qui, par son approche personnelle des problèmes, a su me convaincre qu'un problème d'apparence complexe peut s'avérer encore plus complexe et impénétrable que prévu !!! Je le remercie pour ses marques d'encouragement maintes fois témoignées et exprimées différemment de M. Chassot mais tout aussi efficaces (en particulier depuis que nous partageons le même bureau). Merci aussi pour ses conseils qui vont au-delà du cadre abordé dans ce manuel.

Je veux remercier toutes les personnes directement liées à mes travaux, en premier lieu, M. Jean-Yves Fourniols pour m'avoir orienté à la fin du stage effectué au cours de l'été 2000 vers un stage de DEA qui m'a ouvert les portes de cette thèse. Je remercie également : M. Fabien Garcia, qui a soutenu sa thèse en décembre 2002 et qui m'a préparé un terrain de recherche fertile, M. Florin Racaru, qui a commencé sa thèse en septembre 2004 et m'a particulièrement bien aidé au cours de son stage de DEA. Je remercie également les stagiaires que j'ai co-encadrés avec mon directeur de thèse : MM. Olivier Roche(s), François Armando et Badrounet Daka. Enfin, je remercie M. Vincent Nicomette, maître de conférence à l'INSA pour tout l'enseignement technique qu'il m'a transmis sans compter et sans lequel un bon nombre des travaux reportés ici n'auraient pas pu aboutir. Merci à tous pour votre aide.

Merci à toute l'équipe OLC pour son accueil et sa bonne humeur : M. T. Gayraud, M. P. Owe, Pakal, Olivier, Nico L. (pour son aide lors de la mise en place de la plate-forme expérimentale décrite en fin de ce manuel), Nico R. et Yann (mes premiers modèles de thèsard !), David G., MM. T. Villemur et K. Drira, pour l'accueil dans leur bureau. Merci également aux personnes de l'équipe de recherche DMI de l'ENSICA avec qui j'ai pu avoir des contacts et des activités de recherche riches (DGA… ?!). Je remercie aussi toutes les personnes du LAAS qui m'ont permis d'avoir un cadre de travail agréable : Mmes A. Bergez, A. Evrard, C. Moulin et J. Penavayre ainsi que MM. C. Berty, D. Daurat, E. Le Denmat, Meunier … Bernard et P. Pichon. Ils ont tous contribué à m'offrir de bonnes conditions matérielles de travail. Merci aussi à tous les autres amis que j'ai côtoyés au cours de ces trois années : Farcy, Patricia, Ayda, Bruno, Alex, Georges, Samuel, Taha, Slim, Karim, Petra, Nico M., Christophe E., Michel,…

Je souhaite remercier tous mes proches (mes parents, mon frère, ma tante – qui a intégralement relu scrupuleusement ce document – M et Mme Jean et Anne-Marie Muratet, Guilhem) qui ont été soumis à rude épreuve lors de l'accompagnement de mes travaux et qui ont tous su trouver les mots justes pour m'encourager.

Enfin pour terminer, je m'associe à tous ceux qui ont eu à travailler de près avec moi pour remercier une personne qui m'est très chère et qui a été (et qui sera) mon soutien le plus efficace, Mlle Sylvaine Muratet (appelée affectivement par certains Mlle Gauriol). Merci.

TABLE DES MATIERES

INTRODUCTION GÉNÉRALE **1**

1 PROBLÉMATIQUE ADRESSÉE ET SOLUTION PROPOSÉE **2**

1.1 PROBLÉMATIQUE ADRESSÉE DANS CE MÉMOIRE 2

1.2 SOLUTION PROPOSÉE 2

2 DÉMARCHE ADOPTÉE DANS LE MÉMOIRE **3**

3 PLAN DU MÉMOIRE **4**

CHAPITRE 1 ÉTAT DE L'ART **7**

INTRODUCTION **7**

1 APPLICATIONS MULTIMÉDIAS : CARACTÉRISTIQUES MAJEURES **7**

1.1 CARACTÉRISTIQUES MAJEURES DES APPLICATIONS MULTIMÉDIAS 7

1.2 BESOINS EN QDS DES APPLICATIONS MULTIMÉDIAS 8

1.3 BESOINS EN COMMUNICATION DE GROUPE 12

1.4 BESOINS EN SÉCURITÉ 12

2 INTERNET NOUVELLE GÉNÉRATION **13**

2.1 NOUVELLES SOLUTIONS POUR AMÉLIORER LES RÉPONSES AUX BESOINS DES APPLICATIONS 13

2.2 NOUVELLES PROPOSITIONS D'ARCHITECTURE DE COMMUNICATION 19

3 PROTOCOLES DE SIGNALISATION ADAPTÉS AU MULTI DOMAINES **23**

3.1 INTRODUCTION 23

3.2 DESCRIPTION DES PROTOCOLES 24

3.3 ARCHITECTURES DE SIGNALISATION EN MULTI DOMAINES 33

4 CONCLUSION **40**

4.1 OBJECTIFS DES PROPOSITIONS 40

4.2 LIMITES DES PROPOSITIONS 41

4.3 NOTRE PROPOSITION 42

CHAPITRE 2 ENVIRONNEMENTS IP CIBLÉS ET ARCHITECTURE DE COMMUNICATION **43**

1 ENVIRONNEMENTS IP CIBLÉS **43**

1.1 ENVIRONNEMENT DIFFSERV MONO DOMAINE CIBLÉ 43

1.2 ENVIRONNEMENT DIFFSERV MULTI DOMAINES CIBLÉ 46

2 ARCHITECTURE DE COMMUNICATION **49**
2.1 CONTEXTE ET OBJECTIFS DE L'ARCHITECTURE 50
2.2 PROPOSITION D'ARCHITECTURE AU NIVEAU DES HÔTES 50
3 INTERFACE DE PROGRAMMATION D'APPLICATION (API) **52**
3.1 PARAMÈTRES DE QDS 52
3.2 SÉMANTIQUES DE GARANTIE 53
3.3 PRIMITIVES 54
3.4 RÈGLES D'ENCHAÎNEMENT DES PRIMITIVES 55
4 MESURES DE PERFORMANCES **55**
4.1 PLATE-FORME ET OUTILS DE MESURE 56
4.2 SPÉCIFICATIONS DES MESURES 57
4.3 RÉSULTATS ET ANALYSE 59
4.4 CONCLUSION SUR LES MESURES DE PERFORMANCE 64
5 CONCLUSION **65**

CHAPITRE 3 CARACTÉRISATION DE LA QDS DES SERVICES DE BOUT EN BOUT ET MÉCANISMES DE SÉLECTION AUTOMATIQUE **67**

INTRODUCTION **67**
1 CARACTÉRISATION DES SERVICES IP EN ENVIRONNEMENT MONO DOMAINE **68**
1.1 VALIDATION EN SIMULATION DES MESURES @IRS 68
1.2 MODÈLE ANALYTIQUE RETENU 83
2 CARACTÉRISATION DES SERVICES IP EN ENVIRONNEMENT MULTI DOMAINES **87**
2.1 ÉTUDE MATHÉMATIQUE 87
2.2 APPLICATION AU MODÈLE ANALYTIQUE RETENU ET VALIDATION 89
3 CARACTÉRISATION DU COUPLAGE TRANSPORT / IP **94**
3.1 CARACTÉRISATION DE LA FIABILITÉ 94
3.2 CARACTÉRISATION DU DÉLAI 95
4 ALGORITHME DE SÉLECTION **96**
4.1 HYPOTHÈSES 96
4.2 PRINCIPE DE L'ALGORITHME DE SÉLECTION AUTOMATIQUE DES SERVICES 96
4.3 EXEMPLE 99
5 CONCLUSION **100**

CHAPITRE 4 ARCHITECTURE DE SIGNALISATION MULTI DOMAINES **101**

INTRODUCTION **101**

1 DÉFINITION DE L'ARCHITECTURE **101**
1.1 LOCALISATION 102
1.2 RÔLE DES DIFFÉRENTS PROTOCOLES 104
1.3 ENCHAÎNEMENT 107
1.4 FORMAT DES PDU 108
2 SPÉCIFICATION DE L'ARCHITECTURE EN UML **114**
2.1 PRÉSENTATION DE UML / TAU 115
2.2 SPÉCIFICATION DE L'ARCHITECTURE AU NIVEAU D'UN HÔTE 119
2.3 SPÉCIFICATION DE L'ARCHITECTURE AU NIVEAU D'UN BB (*LOCAL* OU *REMOTE*) 125
2.4 SPÉCIFICATION DE L'ARCHITECTURE AU NIVEAU D'UN ROUTEUR DE BORDURE 132
3 CONCLUSION **133**

CHAPITRE 5 TESTS DU SYSTÈME DE COMMUNICATION **135**

1 OBJECTIF **135**
2 INTÉGRATION DES APPLICATIONS TESTS **136**
2.1 DESCRIPTION DES APPLICATIONS 136
2.2 MODULES D'ADAPTATION 137
2.3 MODIFICATION MINIMALE DES APPLICATIONS 139
3 DESCRIPTION DE LA PLATE-FORME DE TEST **140**
3.1 PLATE-FORME D'ÉMULATION 140
3.2 CHOIX D'IMPLÉMENTATION RETENUS 143
4 TESTS EXPÉRIMENTAUX **147**
4.1 CONFIGURATION DE LA PLATE-FORME 147
4.2 DESCRIPTION DES SCÉNARIOS 150
4.3 RÉSULTATS DES TESTS 150
4.4 PROBLÈME DE LA GÉNÉRALISATION À *N* DOMAINES 154
5 CONCLUSION **155**

CONCLUSION GÉNÉRALE ET PERSPECTIVES **157**

1 CONCLUSIONS **157**
1.1 ENVIRONNEMENT IP DIFFSERV 158
1.2 MODÉLISATION DES PERFORMANCES DES SERVICES 158
1.3 ARCHITECTURE DE COMMUNICATION 159
1.4 ARCHITECTURE DE SIGNALISATION 159
1.5 TESTS DE DÉPLOIEMENT DU SYSTÈME DE COMMUNICATION 159

2 PERSPECTIVES 160
2.1 PERSPECTIVES IMMÉDIATES 160
2.2 PERSPECTIVES À PLUS LONG TERME 160

BIBLIOGRAPHIE 163

BIBLIOGRAPHIE DE L'AUTEUR 171

Introduction générale

Après avoir introduit le contexte général de nos travaux, ce texte présente la problématique adressée et la solution proposée en conséquence. Il donne ensuite la démarche adoptée de notre mémoire pour soutenir cette thèse, et enfin sa structuration.

Ces dernières années, les évolutions conjointes de l'informatique et des télécommunications ont conduit à l'émergence de nouveaux types d'applications distribuées contraintes, telles que les applications multimédias (visioconférences, vidéo à la demande, …) ou la simulation interactive distribuée.

Contrairement aux applications classiques, du type transfert de fichiers, consultation de pages du Web ou encore messagerie électronique, ces applications présentent des exigences nouvelles en terme de communication : délai de transit borné, fiabilité non plus totale mais partielle, garantie de bande passante …, auxquelles ont à faire face tant les réseaux physiques qui les supportent, que l'Internet, la solution incontournable d'interconnexion de ces réseaux.

Dans ce contexte et en raison des limites conceptuelles de l'Internet (initialement conçu pour assurer le transport des applications classiques), résulte la problématique très générale consistant à faire en sorte que ces nouvelles applications puissent être distribuées en satisfaisant leurs contraintes entre deux ou plusieurs hôtes de l'Internet, indépendamment de leur point d'accès (fixe ou mobile).

Dans cette optique, de nombreux travaux de recherche se sont ainsi focalisés depuis dix ans sur cette problématique suivant deux approches complémentaires :

- la première approche préconise d'accroître la complexité des applications pour adapter les logiciels d'application aux variations de performance (gigue et/ou pertes) de l'Internet. Initiée par (Bolot, 1994), cette approche est basée sur l'utilisation des protocoles RTP/RTCP (Schulzrinne, 1999) et est implémentée dans plusieurs applications audio et vidéo de l'Internet actuel ;
- la seconde approche préconise de revoir les niveaux Transport et IP de l'architecture de l'Internet afin d'offrir aux applications des garanties de « Qualité de Service » (QdS). Suivant cette approche, différents travaux ont été initiés et sont encore menés, tant au niveau IP (nouveaux modèle de services, nouvelles architectures - IntServ, DiffServ, …) qu'au niveau Transport (amélioration des mécanismes existants, extension des fonctionnalités, nouveaux protocoles, …).

Plus récemment, d'autres travaux se sont orientés vers la définition d'architectures de communication adressant de façon indépendante les problèmes liés :

- à la complexité croissante (pour les programmeurs d'applications) des interfaces d'accès aux nouveaux systèmes de gestion de la QdS (Garcia, 2002), (Exposito, 2003) ;

– au fait que l'Internet est constitué de différents domaines autonomes (AS : *Autonomous System*) par définition administrés par des fournisseurs distincts (ISP : *Internet Service Provider*).

1 Problématique adressée et solution proposée

Les travaux décrits dans ce mémoire se situent dans le cadre de la seconde des approches précédentes et proposent une solution couplée aux deux problèmes introduits ci-dessus.

1.1 Problématique adressée dans ce mémoire

La problématique générale adressée dans ce mémoire est de définir comment maîtriser la QdS d'un bout à l'autre d'une communication impliquant plusieurs domaines. Plus précisément, les deux principaux problèmes posés sont les suivants :

– il s'agit de **quantifier les performances du service de bout en bout résultant de la mise en œuvre, dans un environnement IP multi domaines, de différents services de niveaux Transport et IP** (ayant chacun un impact sur la QdS), de façon à effectuer un choix d'architecture en adéquation avec des besoins applicatifs exprimés en termes de paramètres de QdS identifiés. Rappelons que ce choix résulte :
  - au niveau Transport, de la possibilité d'utiliser les services de différents protocoles tels que UDP, TCP, SCTP, DCCP ou FPTP ;
  - au niveau IP, de la possibilité d'utiliser plusieurs architectures de gestion de la QdS : IntServ, DiffServ, et pour chacun d'eux de disposer d'autres types de service que le seul *best effort ;*
– sur ces bases, il s'agit alors de **définir l'architecture de signalisation nécessaire à l'obtention de garanties de QdS de bout en bout dans un environnement IP multi domaines**.

Les protocoles de Transport et architecture de gestion de la QdS cités précédemment seront introduits au chapitre 1 relatif à l'état de l'art.

1.2 Solution proposée

Les hypothèses sous-jacentes à notre proposition de solution sont les suivantes :

– vis-à-vis des services et protocoles de l'Internet :
  - trois protocoles de Transport seront considérés : UDP, TCP et FPTP ;
  - les différents domaines IP seront supposés déployer l'architecture DiffServ ;
  - le routage sera supposé faiblement dynamique.
– vis-à-vis des technologies réseaux supports :
  - les domaines seront supposés constitués de liens physiques de point à point, filaires, et dont la topologie d'interconnexion est statique ;
  - aux extrémités, les sites locaux (constitués d'hôtes isolés ou de réseaux locaux) seront supposés suffisamment dimensionnés en bande passante pour accepter toutes les requêtes.

Face à cette problématique, la solution proposée dans cette thèse concerne la définition et la conception d'une **architecture de signalisation globale pour un Internet de type DiffServ multi domaines** :

- **s'intégrant dans une architecture de communication multi services vis-à-vis des niveaux Transport (UDP, TCP, FPTP) et IP (DiffServ) ;**
- **déployant des protocoles et des mécanismes de gestion de la QdS :**
  - **permettant au système de communication (identifié comme le couplage des deux architectures) d'offrir des garanties de QdS sur le transfert de chacun des flux d'une application ;**
  - **simplifiant l'accès au système de communication du point de vue du programmeur d'applications.**

Cette architecture sera implantée dans les hôtes ainsi que dans différents équipements de l'environnement multi domaines considéré.

2 Démarche adoptée dans le mémoire

Pour soutenir cette thèse, la démarche adoptée dans ce mémoire est la suivante :

- nous définirons tout d'abord les modèles d'environnements IP ciblés, mono puis multi domaines ;
- nous présenterons ensuite les principes de conception de l'architecture de communication dans laquelle s'insère notre proposition d'architecture de signalisation. Nous détaillerons en particulier son interface de programmation[1] dont l'une des caractéristiques majeures est qu'elle soustrait son utilisateur du choix des services Transport et IP sous-jacents : en d'autres termes, c'est le système de communication qui, sur la base d'une certaine connaissance (que nous définirons ultérieurement) des performances du service de bout en bout, effectue ce choix dans le cadre d'un mécanisme de sélection automatique, qui sera invoqué dans le cadre de notre architecture de signalisation ;
- nous proposerons alors un modèle de caractérisation des performances du service de bout en bout résultant du couplage d'un protocole de Transport et d'une classe de service IP donnés dans un environnement multi domaines. Pour cela, nous nous appuyons sur des résultats d'expérimentations réalisées sur une plate-forme nationale DiffServ mono domaine (issu du projet RNRT @IRS)[2] et que nous conforterons par des résultats de simulations réalisées à l'aide de l'outil ns-2. Pour le passage au multi domaines, cette étape sera complétée par une étude mathématique qui sera confortée par des résultats de mesures effectuées sur une plate-forme d'émulation de services différenciés en environnement multi domaines ;

[1] API : *Application Programming Interface*

[2] Le projet RNRT @IRS sera introduit au chapitre 2.

- nous présenterons ensuite notre proposition d'architecture de signalisation. La spécification de l'architecture déployée sur chacun des équipements principaux de notre système sera décrite suivant le standard UML 2.0, au moyen de l'outil Tau développé par la société Télélogic ;
- enfin, nous présenterons l'implémentation et le déploiement du système de communication dans son ensemble sur une plate-forme d'émulation de services différenciés (développée dans le cadre de cette thèse) dont nous détaillerons la conception. Deux applications multimédias serviront d'applications tests de notre système.

3 Plan du mémoire

En plus de cette introduction, ce mémoire comporte les chapitres suivants :

- le Chapitre 1 présente un état de l'art des différents éléments du contexte ciblé. Il se divise en trois parties majeures : les applications multimédias et leurs besoins en QdS, puis l'évolution des couches Transport et IP de l'Internet sous l'angle de la QdS, et enfin les propositions de protocoles et d'architectures de signalisation pour les environnements multi domaines. Les conclusions de ce chapitre positionnent notre proposition en analysant les limites des solutions actuelles vis-à-vis de la problématique ciblée ;
- le Chapitre 2 définit tout d'abord les modèles d'environnement IP (mono puis multi domaines) ciblés dans le cadre de nos travaux. Il présente ensuite l'architecture de communication (issue du projet @IRS) dans laquelle s'inscrit notre contribution. Il présente enfin les principaux résultats de mesure (attenants à cette architecture) sur lesquels s'appuient les points majeurs de notre contribution (qui font l'objet des chapitres 3, 4 et 5) ;
- le Chapitre 3 présente tout d'abord les travaux relatifs à la caractérisation des services IP dans un environnement mono domaine ; ces travaux consistent en une campagne de simulations destinée à conforter les mesures présentées au Chapitre 2, suivie par l'élaboration d'un modèle analytique retenu pour caractériser les services IP. Il présente ensuite les travaux relatifs à la caractérisation des services IP dans un environnement multi domaines ; ces travaux consistent en une étude mathématique conduisant à l'élaboration d'un modèle générique, suivis par l'application de ce modèle au modèle analytique retenu pour le mono domaine ; cette étude est validée par une campagne de mesures menée dans un environnement réseau émulant (via le logiciel *Dummynet*) le comportement d'un environnement multi domaines. La troisième partie du chapitre décrit les travaux relatifs à la caractérisation de la QdS des services de bout en bout <u>couplant niveaux IP et Transport</u> offerts par notre architecture de communication dans un environnement multi domaines. Enfin, la dernière partie du chapitre détaille l'algorithme de sélection automatique des services IP et Transport permettant de satisfaire la QdS requise par une application pour le transfert de l'un de ces flux, dans un environnement multi domaines. En conclusion de ce chapitre sont discutées les principales limites des modèles et de l'algorithme proposés.

- le Chapitre 4 présente notre proposition d'architecture de signalisation multi domaines. La première partie du chapitre expose le rôle des différents protocoles de l'architecture, leur localisation, leurs règles d'enchaînement ainsi que le format de leurs PDU. La deuxième partie du chapitre décrit la spécification en UML 2.0 de l'architecture déployée sur chacun des équipements du système. Les diagrammes de séquences d'un certain nombre de scénarios sont également fournis. En conclusion de ce chapitre sont discutées les principales limites de notre proposition d'architecture ;
- le Chapitre 5 présente l'implémentation et les tests de notre système de communication. Dans un premier temps, nous décrivons la plate-forme de test retenue, l'implémentation des différents protocoles de notre architecture et leur déploiement sur la plate-forme. La deuxième partie du chapitre expose les résultats d'expérimentations menées pour deux applications multimédias (un serveur vidéo et une application de visioconférence), retenues comme applications tests de notre système de communication.

En conclusion générale de ce mémoire, nous résumons les contributions majeures de nos travaux, leurs limites ainsi que les perspectives majeures découlant de cette thèse.

Chapitre 1 État de l'art

Introduction

Ce chapitre présente un état de l'art des différents éléments du contexte ciblé.

Nous y décrivons tout d'abord les caractéristiques générales et les besoins en communication des applications multimédias, en particulier leurs besoins en QdS.

Nous présentons ensuite les solutions déployées dans l'Internet, aux niveaux IP et Transport, pour améliorer la QdS des transferts de données.

La troisième partie expose les approches proposées pour donner une solution aux problèmes spécifiques liés à un environnement multi domaines.

En conclusion de ce chapitre, nous comparons les différentes solutions proposées et nous positionnons notre contribution au regard des limites de ces solutions.

1 Applications multimédias : caractéristiques majeures

Ces dernières années, les applications de l'Internet ont considérablement évolué, passant des applications *classiques* basées sur l'échange de données essentiellement textuelles aux applications dites *multimédias*, impliquant la manipulation coordonnée et le transfert de plusieurs types de *média*[3] (texte, graphisme, audio, vidéo, …). Comparativement aux applications classiques, les applications multimédias présentent des contraintes nouvelles sur le transfert de certains de leurs médias (plus spécifiquement l'audio et la vidéo).

Dans ce paragraphe, nous résumons les caractéristiques majeures et les besoins en communication des applications multimédias les plus courantes. Le lecteur pourra se référer en particulier à (Garcia, 2002) pour de plus amples informations sur ces applications, et plus généralement sur les applications temporellement contraintes, telles que la simulation interactive distribuée ou les applications de contrôle commande.

1.1 Caractéristiques majeures des applications multimédias

On distingue deux familles de médias, discrets (par exemple, un fichier texte) ou continus (par exemple, un flux vidéo). Pour chacun d'eux, on distingue trois types d'utilisation :

– la *diffusion différée*, qui implique une transmission intégrale du média avant utilisation ;

[3] Dans la suite de ce document, nous adoptons comme convention d'appeler *média* le média de représentation de l'information, c'est-à-dire l'information codée.

- la *diffusion en quasi temps réel*, qui implique une présentation du média (quasiment) au fur et à mesure de son arrivée ; un exemple type de ces applications est les applications dites de *streaming* audio ou vidéo ;
- la *diffusion interactive*, qui implique une présentation au fur et à mesure du média avec possibilité pour les utilisateurs d'interagir entre eux via ces médias ; un exemple type de ces applications est les applications de visioconférence.

1.2 Besoins en QdS des applications multimédias

1.2.1 Besoins en terme de fiabilité

Les médias continus (audio et vidéo) ont comme caractéristique d'être plus ou moins redondants. Ainsi deux images successives d'une transmission vidéo comportent généralement peu de différences. De cette redondance résulte la possibilité que des pertes d'information (image ou fragment de son) soient acceptables du point de vue de l'utilisateur final.

Il apparaît donc ici pour les médias les plus importants d'une application multimédia (audio et vidéo) une contrainte de fiabilité du transfert des données non plus totale mais partielle, la perte de certaines informations pouvant être acceptable.

Notons cependant que l'expression d'une contrainte de fiabilité partielle est à coupler avec la façon dont sont codées les données audio et vidéo. En effet, certains codages (MPEG par exemple) introduisent une dépendance entre les images qui peut rendre indécodables plusieurs images consécutives en cas de perte de l'une d'entre elles, plus importante que les autres. A l'inverse, un codage de type JPEG n'introduisant aucune dépendance entre les images, une contrainte de fiabilité exprimée en termes d'un pourcentage maximum de pertes admissibles et d'un nombre maximum de pertes consécutives s'avère alors valide.

Tel que nous le présenterons dans le chapitre 2, notre système de communication n'est pas destiné à une famille d'applications particulières (même s'il intéresse en premier lieu les applications multimédias), et encore moins à un type de codage spécifique de l'audio ou de la vidéo. Dans l'interface de programmation qu'il offre aux applications, il inclut ainsi des paramètres de QdS génériques, et c'est à l'application ou à un module d'adaptation de l'application à l'API d'utiliser ces paramètres à bon escient, moyennant une heuristique ad hoc. Nous illustrerons cette dernière remarque au chapitre 5, qui est dédié au déploiement et au test de notre système par deux applications multimédias.

1.2.2 Besoins temporels

Les applications de diffusion différée de médias continus traitent leurs données de la même façon que s'il s'agissait de médias discrets. Elles présentent donc les mêmes contraintes temporelles que celles des applications classiques, à savoir peu ou pas. Deux types d'applications multimédias

présentent donc des contraintes temporelles : les applications de diffusion en temps réel de médias continus et les applications multimédias interactives.

Les contraintes temporelles s'expriment généralement par le biais de deux paramètres : le délai de transit des données et la gigue.

Délai. Pour les applications interactives (telle que la visioconférence), et à un degré moindre pour les applications de diffusion en temps réel (telles que le *streaming* audio ou vidéo), afin que la communication se déroule comme si elle avait lieu localement, il faut que les données soient transmises en un temps inférieur au seuil de perception humain lié au média considéré. Il apparaît ainsi une contrainte sur le délai de bout en bout du transfert des données.

Gigue. Les médias continus (tels que l'audio et la vidéo) présentent des contraintes temporelles dues à leur caractère isochrone. Ces contraintes s'expriment en terme de régularité dans l'arrivée des données (on suppose que la source des données émet à un débit correspondant au débit idéal de présentation). Cette régularité s'exprime par une contrainte sur le temps inter arrivées des données, c'est-à-dire sur la différence entre les dates d'arrivée de deux données successives. Cette contrainte est appelée la « gigue ».

La date d'arrivée d'une donnée étant calculée par :

$t_{réception} = t_{émission} + dt_{min} + \Delta dt$, où :

- dt_{min} désigne le temps de transmission optimal (sans attente dans le réseau).
- Δdt désigne le temps d'attente dans le réseau.

On peut alors exprimer la gigue par :

$t_{inter\ réception} = t_{inter\ émission} + \Delta dt_2 - \Delta dt_1$, où :

- Δdt_1 et Δdt_2 représentent les temps d'attente dans le réseau pour deux données successives.

1.2.3 Besoins en QdS associés aux dépendances entre les données

En plus des besoins exprimés dans les paragraphes précédents, les applications citées peuvent avoir des besoins en terme d'ordre d'arrivée des données échangées. Ces besoins se traduisent pour le système de communication par des contraintes d'ordre partiel. Ces contraintes peuvent s'exprimer sur les données d'un flux ou sur les données de plusieurs flux différents, ce dernier cas concernant directement les applications multimédias.

Les contraintes d'ordre sur plusieurs flux (ordre multimédia ou ordre inter flux) découlent des besoins en synchronisation présentés en particulier par les applications multimédias. Étudions ce type de contraintes au moyen de l'exemple suivant, extrait de (Garcia 2002).

Soit une application consistant en la diffusion d'un bulletin météorologique sur deux fenêtres vidéo d'une station de travail distante (Figure 1). La plus grande de ces fenêtres comporte les images relatives à la présentatrice du bulletin dont les propos sont délivrés par deux sorties audio. La plus

petite des fenêtres, en bas à droite de l'écran, fait apparaître les images d'une seconde personne qui traduit les paroles du présentateur en langage des signes.

Figure 1 : Un exemple d'application multimédia

Trois flux de données sont donc véhiculés dans cette application : deux pour les flux vidéo et un pour le flux audio stéréophonique. Les caractéristiques de présentation de ces différents flux sont les suivantes : (1) le flux relatif à la séquence vidéo 1 est composé de 30 images par seconde, (2) l'incrustation muette (vidéo 2) est animée à raison de 10 images par seconde, et (3) le flux audio est composé de deux fois 60 échantillons par seconde.

Pour cette application, les besoins en synchronisation (ou contraintes d'ordre) se situent non seulement au sein de chaque flux (synchronisation intra flux) mais également entre les flux considérés (synchronisation inter flux).

On distingue ainsi dans la Figure 2 une modélisation possible (basée sur un réseau de Petri série parallèle) de ce type de contraintes :

– les relations de dépendance intra flux, telles que « la donnée 4 précède les données 5 et 6 dans l'ordre de délivrance des données du flux audio » ;
– les relations de dépendance inter flux, telles que « la donnée 2 précède les données 8 et 9 dans l'ordre de délivrance des données vidéo 1 et audio ».

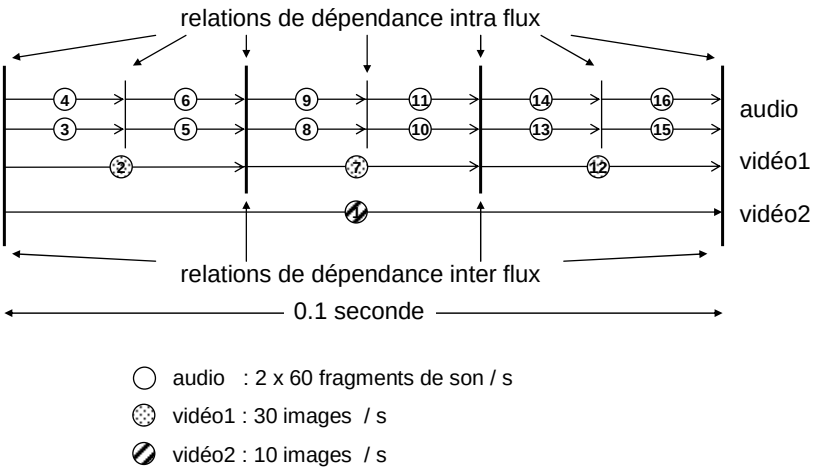


Figure 2 : Un exemple de contraintes de synchronisation intra et inter flux

D'autres modèles de représentation plus évolués, explicitant les exigences de temps, devront être adoptés si nécessaire ; citons par exemple les réseaux de Petri à flux temporisés (*Time Stream Petri Net* : TSPN) qui permettent d'affecter des intervalles de validité temporelle aux événements de réception de chaque donnée (Diaz, 1993), et plus récemment le modèle HTSPN (Senac, 1996), *Hierarchical Time Stream Petri Net*.

1.2.4 Expression des besoins en QdS formulée par l'ITU

Extrait de (ITU, 2001), le Tableau 1 illustre les principales applications ainsi que leurs paramètres de QdS.

Type de médias	Applications	Communication	Taille des données	Délai[1]	Gigue	Pertes[2]
Donnée	Images	unidirectionnelle	< 100 KB	De < 15 s À < 60 s	N/A	Aucune
Donnée	Jeux interactifs *Telnet* Contrôle com.	bidirectionnelle	< 1 KB	De < 200 ms À < 250 ms	N/A	Aucune
Donnée	Email (accès) WEB e-commerce	unidirectionnelle	< 10 KB	De < 2 s À < 4 s	N/A	Aucune
Donnée	FTP	unidirectionnelle	10 KB – 10 MB	De < 15 s À < 60 s	N/A	Aucune
Donnée	Fax	unidirectionnelle	~ 10 KB	< 30 s/page	N/A	$<10^{-6}$ BER
Audio	Conversation	bidirectionnelle	4 – 64 Kbits/s	De < 150 ms À < 400 ms[3]	< 1 ms	< 3% PLR
Audio	*Streaming* Audio	unidirectionnelle	16 – 128 Kbits/s[3]	< 10 s	<< 1 ms	< 1% PLR
Vidéo	Vidéoconférence	bidirectionnelle	16 – 384 Kbits/s[3]	De < 150 ms À < 400 ms	Synchro[4] < 80 ms	< 1% PLR
Vidéo	*Streaming* Vidéo	unidirectionnelle	16 – 384 Kbits/s	< 10 s	N/A	< 1% PLR

[1] unidirectionnel
[2] PLR : taux de paquets perdus ou *Packet Loss Ratio* - BER taux d'erreurs bits ou *Bit Error Rate*
[3] la valeur exacte dépend beaucoup du *codec* utilisé
[4] synchronisation entre audio et vidéo (*lips-synchro*)

Tableau 1 : Principales applications distribuées et leur paramètre de QdS

1.2.5 Services de niveau Session adaptés aux applications multimédias

Notons enfin que pour définir le paramétrage d'une connexion de bout en bout (classiquement de niveau Transport), de nombreux travaux ont été initiés au niveau Session. Notons en particulier les travaux réalisés autour des protocoles :

- SAP (Handley, 2000) et SDP (Handley, 1998). *Session Announcement Protocol* et *Session Description Protocol* sont deux protocoles qui permettent d'établir des sessions multimédias. Les paquets contiennent la description d'une session, incluant entre autres les médias et le format de codage des données audio/vidéo ainsi que le protocole de Transport ;
- SIP (Rosenberg, 2002). *Session Initialization Protocol* est un protocole qui permet aux participants d'une session de s'accorder sur les caractéristiques de la session : type de *codec*, …

Nous ne détaillerons pas davantage ces travaux car ils ne concernent pas directement le contexte de cette thèse.

1.3 Besoins en communication de groupe

Le média de communication auquel nous sommes le plus habitués, la voix, permet d'envoyer simultanément (au délai de propagation de l'onde sonore près) la même information à plusieurs personnes. Certaines applications (comme la visioconférence à plus de deux participants) se basent sur un schéma de communication de ce type où une information doit être envoyée simultanément à plusieurs récepteurs. De telles applications ont deux choix de fonctionnement : soit se baser sur un service de diffusion 1 vers n que l'on appelle *multicast,* soit répéter l'information à chacun des *n* récepteurs. On voit aisément que la deuxième solution implique une perte de ressources tant du point de vue de l'application que du système de communication qui doit véhiculer plusieurs fois la même information. Pour ces raisons, les applications qui souhaitent diffuser des informations ont besoin d'un service multicast, dont la QdS doit cependant être supérieure à celle offerte par la solution actuelle (UDP/IP multicast) qui n'offre aucune garantie sur le transfert des données.

On distingue deux types de besoins multicast, les besoins en communication de 1 émetteur vers *n* récepteurs (par exemple, une application de diffusion de télévision) et les besoins en communication de *n* émetteurs vers *m* récepteurs (par exemple, une application visioconférence où tous les participants ont la parole). Dans la plupart des cas, il n'est pas gênant de décomposer le second cas en *n* communications de 1 vers *m*.

1.4 Besoins en sécurité

Avec le succès de l'Internet auprès des entreprises et du public, les applications véhiculent de plus en plus de données confidentielles et sensibles (informations d'ordre privé, d'ordre financier,…). Par ailleurs, le nombre de tentatives d'écoute, de manipulation ou de création d'informations augmente en parallèle. De plus en plus d'applications réclament de la part du système de communication un transfert d'information sécurisé. (Moffet, 1994) définit cinq objectifs pour la sécurité dans un système distribué : (1) la confidentialité, qui garantit qu'une tierce personne ne peut lire les données de la communication, (2) l'intégrité des données, qui garantit que les données envoyées ne seront pas modifiées ni détruites, (3) la disponibilité des données, qui garantit que les données envoyées seront disponibles et ne seront pas perdues, (4) l'authentification, qui garantit l'identité d'un utilisateur, et (5) la non-répudiation, qui garantit qu'un utilisateur ne pourra pas nier avoir envoyé ou reçu des données.

2 Internet nouvelle génération

Afin de répondre aux besoins des applications à QdS, des études ont d'abord été menées sur les protocoles de Transport afin d'en augmenter les performances et les fonctionnalités. Dans un deuxième temps, des études ont été menées au niveau IP afin de fournir aux paquets véhiculés un service différent (et meilleur) que le *best effort* actuel, qui n'offre aucune garantie.

Dans un premier paragraphe (2.1), nous présentons tout d'abord les propositions issues de deux groupes de travail de l'IETF pour offrir des services IP améliorés : le groupe IntServ WG : *Integrated Services Working Group* et le groupe DiffServ WG : *Differentiated Services Working Group* ; nous présentons ensuite les nouveaux protocoles de Transport SCTP, DCCP, FPTP conçus pour étendre les fonctionnalités et/ou les services des protocoles TCP et UDP.

Dans le paragraphe (2.2), nous exposons trois des principales propositions d'architecture de communication intégrant en partie les évolutions présentées ci-dessus.

2.1 Nouvelles solutions pour améliorer les réponses aux besoins des applications

2.1.1 Couche IP : IntServ

Le groupe IntServ propose d'offrir des garanties de QdS par flux et a défini deux types de service en plus du *best effort* : le *Controlled Load* (CL) et le *Guaranteed Service* (GS). Le CL propose un service de bout en bout exprimable de façon qualitative en terme de bande passante : il assure que la transmission se fera comme sur un réseau peu chargé (pas de congestion). Le GS propose un service exprimable de façon quantitative en terme de bande passante et de délai de transit maximal : il garantit que tous les paquets d'un même flux arriveront (aux erreurs dues au médium physique près) en un temps borné défini par l'application qui utilise le service. Afin de réserver les ressources réseau (bande passante et mémoire tampon) nécessaires à l'obtention de ces services, l'approche IntServ nécessite l'utilisation d'un protocole de réservation de ressources : RSVP (*Resource reSerVation setup Protocol*) (Braden, 1997). Ce protocole propage la demande de réservation à tous les routeurs sur le chemin des données (de façon dynamique afin de s'adapter aux changements de route). Chaque routeur est en charge d'accepter ou non la réservation en tenant compte des ressources disponibles localement et de la caractérisation du trafic fournie avec la réservation. Une interface de programmation applicative (API) a été définie pour accéder à ces services, la RAPI (Braden, 1999).

2.1.2 Couche IP : DiffServ

L'approche développée par le WG DiffServ est expliquée plus en détail car elle constitue un élément important du contexte de nos travaux.

L'idée de base des solutions DiffServ est de fournir une QdS différenciée aux paquets traversant un réseau tout en repoussant (le plus possible) la complexité du traitement en bordure du réseau afin de ne pas surcharger le cœur du réseau. De plus, afin d'éviter le problème de passage à l'échelle inhérent aux solutions IntServ, le choix a été fait de traiter un nombre limité d'agrégats (paquets IP n'appartenant pas nécessairement à un même flux) plutôt que des flux individuels.

Dans les paragraphes suivants, nous présentons d'abord une notion importante pour les solutions DiffServ, la notion de domaine, puis nous décrirons les principes généraux de ces solutions, avant de présenter l'architecture DiffServ de fourniture de QdS au niveau IP.

2.1.2.1 La notion de domaine

Tel que nous l'avons vu dans l'introduction générale, l'Internet est constitué d'une interconnexion a priori anarchique de réseaux. Cependant, plusieurs de ces réseaux sont souvent rassemblés sous une même autorité administrative (par exemple dans les grandes entreprises, les centres de recherches, les universités, ...) ; (Blake, 1998) désigne par domaine, un ensemble de nœuds (hôtes et routeurs) administrés de façon homogène.

Dans un domaine, on distingue les nœuds internes et les nœuds frontières : les premiers ne sont entourés que de nœuds appartenant au domaine alors que les seconds sont connectés à des nœuds frontières d'autres domaines.

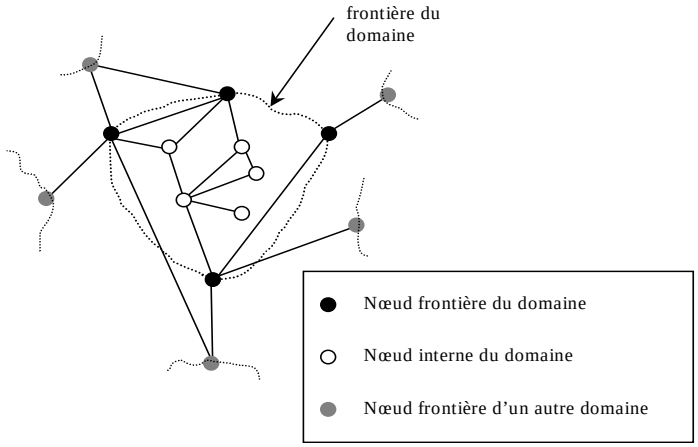


Figure 3 : Distinction des nœuds d'un domaine

2.1.2.2 SLA : Service Level Agreement

Pour un utilisateur (une personne ou une organisation qui loue les services d'un ISP pour accéder au réseau), l'utilisation d'une architecture à services différenciés implique la signature d'un contrat avec le fournisseur d'accès Internet : ce contrat s'appelle le *Service Level Agreement*[4] (SLA).

Contrairement à ce qui se passe avec RSVP, ce contrat est signé avant toute connexion au réseau, et non à l'établissement d'une quelconque session (établir une session RSVP revient en effet à

[4] Accord de niveau de service.

passer un contrat avec les routeurs intermédiaires, qui garantissent certaines propriétés du transport de données tant que le trafic respecte un certain profil).

Ce contrat contient les informations suivantes :

- le trafic que l'utilisateur peut injecter dans le réseau fournisseur (en termes de volume de données, de débit moyen, d'hôtes sources ou destinations, ...[5]),
- les actions entreprises par le réseau en cas de dépassement de trafic (rejet, surtaxe, ...),
- la QdS que le fournisseur s'engage à offrir au trafic généré ou reçu par l'utilisateur (ou les deux). Celle-ci peut s'exprimer notamment en termes de délai, de bande passante, de fiabilité ou de sécurité.

Pour le moment, seuls des contrats statiques, c'est-à-dire peu susceptibles de changer dans le temps, sont étudiés.

Après signature du SLA, l'utilisation des services DiffServ est transparente pour l'utilisateur, l'architecture DiffServ ayant été conçue pour fonctionner avec les applications déjà existantes.

Dans le paragraphe 3, nous reviendrons sur ces contrats en étudiant les problèmes spécifiques des communications traversant différents domaines autonomes pour lesquelles sont impliqués plusieurs SLA. Notre proposition d'architecture s'inscrit dans ce contexte multi domaines qui est défini par un ensemble de domaines contigus implémentant des services différenciés.

2.1.2.3 PHB : Per Hop Behavior

Du point de vue du réseau, l'implantation de l'architecture nécessite un découpage du réseau en domaines (au sens domaine Internet). Tous les nœuds (hôtes et routeurs) d'un domaine implémentent les mêmes classes de service et les mêmes comportements vis-à-vis des paquets des différentes classes (*Per Hop Behavior*[6] : PHB). Un comportement inclut le routage, les politiques de service des paquets (notamment la priorité de passage ou de rejet en cas de congestion) et éventuellement la mise en forme du trafic entrant dans le domaine. Les nœuds internes ne doivent pas conserver d'états en mémoire (contrairement à ce qui apparaît dans l'architecture IntServ), ils ne font que transmettre les paquets selon le comportement défini pour leur classe. Ce comportement est donc purement local et ne tient pas compte d'un état global du réseau. Toutefois, nous verrons dans les chapitres suivants qu'il est possible, sous le couvert de certaines hypothèses concernant le trafic global, de caractériser, au moins statistiquement, les services obtenus entre deux nœuds frontières. Les nœuds frontières se chargent de marquer les paquets selon le code réservé à chaque classe, comme nous allons le voir dans la partie suivante.

---

[5] Dans la documentation, les termes de caractérisation du contrat sont volontairement informels afin de rendre la spécification la plus ouverte possible.

[6] Littéralement : comportement par saut.

2.1.2.4 Architecture à services différenciés

L'architecture proposée par le groupe DiffServ (Blake, 1998) est basée sur un modèle simple dans lequel le trafic entrant dans un réseau est conditionné, puis assigné à une classe de comportement, au point d'entrée de ce réseau. Chaque classe est identifiée par un code unique : le *DS CodePoint* (DSCP). A l'intérieur du réseau, les paquets sont acheminés selon le comportement associé au code de la classe à laquelle ils appartiennent.

Examinons à présent les éléments clefs dans un environnement multi domaines du réseau à services différenciés.

Les services différenciés sont mis en œuvre grâce à un conditionnement du trafic entrant et un acheminement des paquets selon un comportement par nœud (PHB).

Le conditionnement du trafic intervient aux nœuds frontières d'un domaine (à l'entrée de celui-ci) afin d'assurer que le trafic entrant est conforme aux règles spécifiées dans le SLA et de le préparer aux traitements par PHB à l'intérieur du domaine. On distingue deux cas d'entrée d'un paquet dans un domaine : quand le paquet passe par un nœud frontière d'un domaine vers un nœud frontière d'un autre domaine et quand le paquet est généré par l'hôte source. En fait, on réunit les deux cas en considérant l'hôte source comme un domaine à un seul nœud (donc obligatoirement un nœud frontière).

Le conditionnement est réalisé par un *conditionneur de trafic*, voir Figure 4, pouvant contenir les éléments suivants :

- le classificateur sélectionne des paquets dans le trafic en se basant sur le contenu d'une partie de leur en-tête (cette classification peut être faite sur le DSCP seulement : classification par agrégation des comportements), ou sur n'importe quelle combinaison d'un ou plusieurs champs de l'en-tête du paquet (adresse source ou destination, type de protocole), et de l'en-tête de niveau Transport (TCP ou UDP) tels que les numéros de port source ou destination. Le rôle d'un classificateur est d'extraire certaines caractéristiques des paquets et de les transmettre aux autres éléments du conditionneur ;
- le mesureur détermine les paquets qui respectent le profil associé à leur classe et ceux qui sont hors profil. Selon qu'ils sont dans le profil ou hors profil, les paquets sont traités différemment (en fonction de ce qui est spécifié dans le SLA). Les actions peuvent être :
  - pour un paquet respectant le profil :
    - de le laisser passer sans autre conditionnement (cas où les domaines utilisent deux ensembles de comportements et de marquage identiques) ;
    - de marquer les paquets selon un nouveau DSCP (si le paquet n'était pas encore marqué ou si les deux domaines utilisent des marquages et des comportements différents) ;
  - pour les paquets hors profil :
    - le retardement de ceux-ci jusqu'à ce qu'ils respectent le profil (*shaping*) ;

  - le rejet des paquets ;
  - le marquage avec un DSCP spécial ;
  - le déclenchement d'une action de compte rendu.
- le rôle du marqueur est donc de marquer l'en-tête des paquets qui lui sont transmis avec le DSCP correspondant à leur classe s'ils sont dans le profil ou selon un DSCP spécifique sinon ;
- le rôle de l'écarteur est de retenir des paquets hors profil jusqu'à ce qu'ils soient dans le profil. Un écarteur possède une mémoire de taille finie : des paquets sont donc détruits s'il y a saturation de celle-ci.

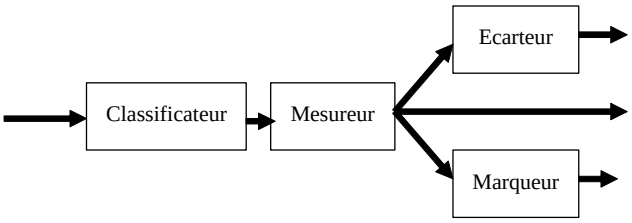


Figure 4 : Structure d'un conditionneur de trafic

Les solutions DiffServ permettent donc à un administrateur de réseau de proposer des services à ses utilisateurs de façon souple et sans problème de mise à l'échelle. Les propositions DiffServ sont actuellement considérées comme les plus adaptées pour la définition du futur Internet. Leurs limites ont principalement trait :

- au manque de finesse dans le paramétrage des services, dû à la nécessité de limiter le nombre d'agrégats dans le réseau ;
- à la difficulté d'un accord entre administrateurs des différents domaines. Ce point sera repris dans le paragraphe (3.1) concernant l'environnement multi domaines.

2.1.3 Couche Transport (étendue)

De nombreux travaux se sont attachés à proposer de nouveaux protocoles de Transport pour enrichir les fonctionnalités et/ou les services des protocoles UDP et TCP.

Les trois paragraphes suivants présentent trois de ces protocoles, SCTP et DCCP, actuellement développés à l'IETF, ainsi que le protocole FPTP, conçu au LAAS-CNRS et à l'ENSICA.

2.1.3.1 SCTP

Le protocole *Stream Control Transmission Protocol*, SCTP (Steward, 2000), est un protocole de Transport à fiabilité totale se déployant sur un service paquet de niveau Réseau sans connexion, offert par exemple par le protocole IP. Il est unicast et orienté session, une session étant définie comme une association établie entre deux hôtes. Dans le cas où un hôte dispose de plusieurs adresses IP, les adresses sont échangées lors de l'établissement de la session (on appelle *multi-homing* le fait que plusieurs adresses IP peuvent correspondre à une session). Un mécanisme de contrôle d'erreur est

implémenté dans SCTP et permet de détecter les pertes, la rupture de séquences, la duplication ou la corruption de paquets. Un schéma de retransmission est utilisé pour corriger ces erreurs. SCTP utilise le principe de *Selective ACKnowledgement* : SACK pour la confirmation de la réception des données. Les retransmissions sont faites après expiration d'un *timer* ou sur interprétation du SACK.

Au contraire de TCP, SCTP est orienté message (ce qui le rapproche par cet aspect de UDP). Chaque paquet contient un en-tête commun et une partie donnée (contenant soit des données utilisateurs soit des données de contrôle). En fait, bien qu'il soit orienté message, plusieurs données peuvent être contenues dans le même paquet, mais seront délivrées à l'application avec le format des messages initiaux.

SCTP offre un service de multiplexage/démultiplexage entre flux : une application multimédia peut être découpée en plusieurs flux pouvant avoir chacun des schémas de remise des données différents. C'est donc un protocole d'ordre total au sein d'un flux et n'offrant aucun garantie sur l'ordre inter flux. Ce qui permet au protocole de délivrer les données d'un flux même si des pertes ou des déséquencements sont détectés sur un autre flux. Le type de contrôle de flux et de congestion est négocié à l'établissement de la connexion. Ces mécanismes sont construits sur la base des algorithmes de TCP : le récepteur informe l'émetteur de sa taille de *buffer* et la taille de la fenêtre de congestion est contrôlée au cours de la connexion SCTP. Les mécanismes de *slow start*, *congestion avoidance*, *fast-recovery* et de *fast-retransmit* sont les mêmes que ceux de TCP mais ils utilisent les paquets SCTP comme unités d'acquittement.

SCTP peut intéresser les applications désirant un service de transport à ordre partiel. Cependant, le fait que SCTP offre un service totalement fiable entraîne une incompatibilité avec les applications multimédias ayant des contraintes en terme de débit, de délai ou de gigue. Une extension (Stewart, 2003) de SCTP permet d'offrir un service à fiabilité partielle temporisée. Une fiabilité partielle temporisée signifie que l'utilisateur peut spécifier une durée de vie à son message. Mais ce service n'est pas adapté aux applications à temps contraint présentant des données applicatives spécifiques, comme par exemple les données des images I, P et B d'un flux vidéo MPEG.

2.1.3.2 DCCP

Le protocole *Datagram Congestion Control Protocol*, DCCP (Kohler, 2002), offre un service de transport non fiable pour des flux *datagram* (donc type UDP) mais intégrant un mécanisme de contrôle de congestion, ce qui permet aux applications utilisant habituellement UDP de ne pas avoir à implémenter le leur. Le but de DCCP est d'offrir l'efficacité d'UDP à certaines applications tout en respectant les autres flux (TCP) du réseau. Les mécanismes de contrôle de congestion sont négociés pour les deux sens de la connexion entre les hôtes au moyen d'un identifiant appelé *Congestion Control IDentifier*, CCID. Plusieurs mécanismes sont disponibles, parmi lesquels un contrôle de congestion *TCP-like* utilisant une fenêtre de congestion et un algorithme TFRC *TCP-Friendly Rate Control* (Floyd, 2003).

DCCP peut être utilisé par toutes les applications présentant des contraintes temporelles et qui sont capables de s'adapter aux fluctuations de débit imposées par les mécanismes de contrôle de congestion.

2.1.3.3 FPTP

Le protocole *Fully Programmable Transport Protocol*, FPTP (Ernesto, 2003), offre un service à ordre partiel et fiabilité partielle complètement paramétrable : possibilité de synchronisation inter et intra flux, taux de pertes acceptables, nombre de pertes consécutives acceptables, … Il intègre deux types de connexion :

- une connexion de contrôle multimédia, basée sur des échanges de messages XML.
- une connexion mono média, relative au transport des données ayant des contraintes de QdS. Chacune de ces connexions est unidirectionnelle.

Pour pouvoir accéder à ces services, les applications multimédias disposent d'une API *socket* ressemblant à une API classique UDP/TCP mais ayant des primitives particulières pour pouvoir paramétrer la connexion.

Toutes les applications multimédias standard peuvent facilement utiliser ce protocole pour le transport de leurs données. De plus, les travaux récents concernant cette souche de Transport évoluée intègrent les nouveaux algorithmes de contrôle de congestion et deviennent compatible avec des mécanismes tel que TFRC.

Nous présentons dans le paragraphe 2.2 dans un premier temps la définition d'une architecture de communication de bout en bout dont le but est de fournir de la QdS puis des implémentations d'architecture intégrant l'une ou l'autre des solutions présentées.

2.2 Nouvelles propositions d'architecture de communication

Le paragraphe précédent a décrit les solutions de niveaux IP et Transport envisagées pour répondre aux besoins des nouvelles applications. Cependant, elles ne constituent pas une solution complète au problème posé et il est nécessaire de concevoir une architecture de communication de bout en bout visant à intégrer les solutions envisagées.

Ce paragraphe présente trois architectures de communication. Les deux premières, dont les travaux ont été initiés dans le cadre de projets de recherche européens, intègrent l'approche DiffServ. La troisième, dont la version la plus aboutie est décrite dans (Ernesto, 2003), s'appuie sur le protocole à ordre partiel et fiabilité partielle FPTP.

2.2.1 Aquila

L'architecture de communication du projet Aquila[7] (AQUILA, 2000) est basée sur les solutions DiffServ de fourniture de QdS au niveau IP. L'objectif de ce projet est de fournir un contrôle dynamique pour du trafic à QdS. Pour se faire, le projet Aquila définit une couche de contrôle des ressources (*Resource Control Layer* : RCL) au-dessus d'un domaine DiffServ. Le RCL gère les ressources du domaine au travers de trois modules :

- l'agent de contrôle des ressources (*Resource Control Agent* : RCA) contrôle et mesure les ressources disponibles dans le réseau. C'est l'autorité de plus haut niveau pour la gestion des ressources dans le réseau Aquila. Il distribue les ressources aux agents de contrôle d'admission sur la base des requêtes de ceux-ci et de l'historique des ressources utilisées ;
- l'agent de contrôle d'admission (*Admission Control Agent* : ACA) est présent dans tous les routeurs de bordure et réalise le contrôle de politique et le contrôle d'admission sur la base des ressources qu'il s'est vu allouées par le RCA. Chaque ACA est donc capable de réaliser le contrôle d'admission de façon autonome ;
- l'interface pour les applications (*End-user Application Toolkit* : EAT) est une interface graphique qui permet à l'utilisateur ou aux applications lancées par celui-ci de réclamer des ressources au système de communication afin de bénéficier de la QdS voulue ;

La signalisation dans un domaine est réalisée au moyen de protocoles existants (RSVP par exemple) ou sur des technologies telle que CORBA (*Common Object Request Broker Architecture*). De plus l'architecture Aquila intègre des propositions relatives à l'aspect multi domaines de la fourniture de QdS avec l'utilisation du protocole BGRP (*Border Gateway Reservation Protocol*) sur lequel nous reviendrons dans le paragraphe (3.3.2).

L'architecture Aquila s'adresse à tous les types d'applications et les paramètres de service qu'elle utilise sont le délai de bout en bout, le taux de pertes et le débit.

Le déploiement de cette architecture requiert un réseau supportant les services différenciés.

2.2.2 Tequila

L'architecture Tequilla[8] est basée sur un réseau supportant les services différenciés. Elle est axée sur la création et la gestion de nouveaux services, décrits par des SLS[9] (*Service Level Specification*). L'architecture Tequila se découpe en trois grandes parties, chacune responsable d'un ensemble de tâches :

---

[7] AQUILA : *Adaptive resource control for QoS Using an IP-based Layered Architecture*. Projet IST-1999-10077. Ce projet a commencé en janvier 2000 et s'est terminé en décembre 2002.

[8] TEQUILA : *Traffic Engineering for QUality of service in the Internet at LArge scale*. Projet IST débuté en janvier 2000 et se terminant en juin 2002.

[9] Le SLS est une partie du SLA décrit dans le paragraphe 2.1.2.2 ; il contient la spécification technique du service.

- la gestion de SLS s'occupe de la souscription de SLS, l'invocation de SLS et la prévision de trafic. La souscription de SLS permet à un utilisateur de s'enregistrer pour l'utilisation future d'un service. Cette fonction doit fournir l'authentification nécessaire afin de garantir l'identité des souscripteurs. L'invocation de SLS fait partie du plan de contrôle et gère chaque flux de façon dynamique. C'est cette fonction qui réalise le contrôle d'admission par flux. Enfin la partie prévision du trafic génère une matrice prévisionnelle du trafic qu'elle fournit à la fonction de dimensionnement du réseau de la partie ingénierie de trafic ;
- la partie ingénierie de trafic (*Traffic Engineering* : TE) proposée dans l'architecture Tequila peut être décomposée en trois fonctions : le dimensionnement du réseau (*Network Dimensioning* : ND), la gestion dynamique de routes (*Dynamic Route Management* : DRtM) et la gestion dynamique de ressources (*Dynamic Resources Management* : DRsM). La fonction ND se charge de configurer la distribution des ressources dans le réseau suivant les instructions du module de prévision du trafic. Le module ND fournit aussi des informations aux autres modules du TE. Cette fonction opère sur une échelle de temps de l'ordre du jour ou de la semaine. La fonction DRtM s'occupe de la gestion des routes de façon à satisfaire les contraintes de routage issues de la souscription des SLS. Ce module fournit donc un routage à QdS permettant de router les flux sur des liens permettant de garantir la QdS contractée (par exemple en évitant le passage de flux interactifs sur des liens à grande latence comme les liens satellites). Enfin la fonction DRsM est distribuée entre les routeurs du réseau et partage les ressources disponibles sur chaque routeur entre les différents PHB. Le module DRsM fournit des informations au module DRtM sur la condition des ressources locales et informe le module ND quand l'état des ressources l'empêche de fonctionner correctement ;
- la gestion de politiques (*Policy Management*) permet de définir des politiques de haut niveau qui sont ensuite traduites et transmises à chaque niveau de l'architecture. Elle comprend trois parties fonctionnelles, l'outil de gestion de politiques, le service de sauvegarde de politiques et le consommateur de politiques. L'outil de gestion de politiques permet à un administrateur de réseau de décrire, avec un langage de haut niveau, les politiques qui régissent l'accession aux services. Une fois décrites, les politiques sont enregistrées par un service de sauvegarde de politiques. Une fois enregistrées, leur activation est réalisée en les transmettant aux agents consommateurs de politiques présents à chaque niveau de l'architecture.

Cette architecture se distingue de l'architecture précédente dans le sens où elle ne fournit pas d'API à l'utilisateur : l'utilisation d'un service est régie par des contrats signés entre un utilisateur (réseau pair ou utilisateur final) et un fournisseur de services. La fourniture de QdS est réalisée par un mélange de mise en œuvre de PHB, de routage à QdS et de Traffic Engineering dynamique.

Par sa capacité à créer de nouveaux services, Tequila s'adresse à tous les types d'applications. Bien que le service offert par l'architecture du projet Tequila ne soit pas prédéfini, les exemples tirés de (Trimintzios, 2001) montrent une utilisation du délai, de la gigue, du taux de pertes et du débit comme paramètres de service. L'architecture Tequila permet la définition de ces paramètres de façon

quantitative ou qualitative (paramètre haut, moyen ou bas). Pour sa mise en œuvre, l'architecture Tequila nécessite de disposer d'un réseau supportant les services différenciés, et de pouvoir installer des fonctionnalités au cœur de celui-ci.

2.2.3 POC et XQoS

XQoS n'est pas le nom d'une architecture mais un langage de spécification d'une session multimédia défini dans (Exposito, 2002). XQoS vient de l'architecture POC (*Partial Order Connection*). Par la suite par simplification, nous appellerons XQoS l'architecture utilisant ce langage.

XQoS est basée sur l'utilisation du protocole FPTP paramétré par le biais d'une API classique ou par le biais d'une caractérisation de la session faite à partir d'un fichier XML. Dans les deux cas, l'application doit spécifier la QdS en termes de paramètres de fiabilité, d'ordre, de synchronisation et de débit. Le protocole FPTP est alors capable de mettre en œuvre soit les services des protocoles standard comme UDP et TCP, soit de nouveaux services pour satisfaire la requête de l'application.

La Figure 5 montre un exemple de spécification d'une session par une application à l'aide d'un fichier XML.

```
<flow id="video"
        sourceAddr="192.168.0.1" sourcePort="2010"
        destAddr="192.168.0.2" destPort="2010"
        order="partial" reliability="partial" priority="p2"
        bandwidth="342kbps">
        <definition name="http://dmi.ensica.fr/xqos/RTP_MPEG2.xqos">
          <param name="bandwidthImagesI" value="75kbps" />
          <param name="bandwidthImagesP" value="138kbps" />
        …
        …
          <param name="reliabilityImagesB" value="none" />
          <param name="repeatImagesB" value="2" />
        </definition>
</flow>
<flow id="images"
        sourceAddr="192.168.0.1" sourcePort="2020"
        destAddr="192.168.0.2" destPort="2020"
        order="partial" reliability="partial" priority="p3">
        <seq>
          <subflow id="layer0" reliability="full" order="none" />
          <subflow id="layer1" reliability="50%" order="none" />
        …
        </seq>
 </flow>
```

Figure 5 : Exemple de session XQoS.

Nous y retrouvons deux flux principaux relatifs aux transferts d'une vidéo et d'une image. Pour le flux vidéo de type MPEG2 défini par un couple adresse IP et numéro de port source et

destination, les images B, P et I n'auront pas la même QdS en termes de bande passante requise et de fiabilité. Pour le flux de l'image, l'utilisateur a pu préciser également pour chacune des couches composant cette image une QdS spécifique.

En dépit du paramétrage important autorisé par cette approche, l'architecture XQoS ne prend pas en compte les solutions de niveau IP (IntServ ou DiffServ) et se base donc implicitement sur un service IP de type *best effort*, ce qui ne permet pas d'offrir des garanties sur le respect de contraintes temporelles.

3 Protocoles de signalisation adaptés au multi domaines

3.1 Introduction

Ce paragraphe présente les différentes solutions proposées pour résoudre certains des problèmes spécifiques à la gestion de la QdS de bout en bout dans un environnement multi domaines. Ces problèmes sont causés par le fait que les communications doivent traverser plusieurs ISP, gérant chacun un domaine autonome (AS), et ayant des politiques différentes de gestion de la QdS.

Dans nos travaux, nous considérerons les trois problèmes suivants :

- (P1) la définition des services. Dans le cas du mono domaine, la définition des services offerts aux clients est précisée dans la partie SLS du SLA, contrat établi entre le client et l'ISP. Dans le cas du multi domaines, une communication implique la participation des différents acteurs rencontrés. Se posent alors d'abord comme problèmes la définition et la caractérisation homogène des services présents dans les différents AS ;
- (P2) la découverte des services IP disponibles au sein de chaque AS. Dans le cas du mono domaine, ce point ne présente pas de difficulté car les utilisateurs connaissent les (classes de) services IP qui leur sont proposés par l'ISP lors de l'établissement du SLA entre les deux parties. Dans un environnement multi domaines, la connaissance des services tout au long de la route est nécessaire pour pouvoir évaluer de bout en bout la QdS. Pour cela, il s'agit de définir les mécanismes et protocoles permettant de découvrir les caractéristiques des services rencontrés et de faire remonter cette connaissance aux services de communication d'extrémités ;
- (P3) la gestion des ressources entre domaines adjacents. Un problème nouveau associé à la gestion des ressources apparaît lorsque la communication traverse plusieurs domaines. En effet, obtenir des garanties de QdS de bout en bout dans un environnement multi domaines impose potentiellement d'envisager une réservation dynamique des ressources entre domaines voisins. Nous expliciterons davantage ce problème au Chapitre 2, en relation avec la façon dont sont dimensionnés les liens de bordure d'un domaine.

Notons enfin qu'il existe plusieurs autres problèmes liés au passage du multi domaines comme par exemple ceux concernant la tarification (ou *billing*),… mais ils sortent du cadre de cette thèse, et ne seront pas considérés par la suite.

La suite de ce paragraphe est structurée de la façon suivante.

Le paragraphe (3.2) présente plusieurs des protocoles de signalisation utilisables dans un contexte multi domaines. Le paragraphe (3.3) décrit les principales propositions d'architectures de signalisation basées sur ces protocoles en faisant ressortir les objectifs ciblés par chacune d'elles.

Deux approches différentes seront étudiées :

- une approche basée sur l'utilisation du protocole de routage inter domaine BGP, qui met en œuvre des mécanismes de réservation de ressources par l'intermédiaire d'un champ de l'en-tête des paquets BGP. Cette approche apparaît naturellement adaptée au contexte par le fait que BGP a été conçu pour être déployé dans un environnement multi domaines ;
- une approche basée sur l'utilisation du protocole COPS, qui centralise les mécanismes de gestion de la QdS sur des équipements particuliers (*Bandwidth Broker* - BB) de chacun des AS. Notons que ce protocole n'a pas été développé dans ce but initial.

Le choix d'exposer ces deux approches est justifié par le raisonnement suivant :

- la première est illustrative des approches que l'on peut qualifier de « diffusives », dans le sens où les équipements intermédiaires (routeurs, BB,…) voient passer des paquets de signalisation (type paquet BGP) et rajoutent des informations dans ces paquets jusqu'à l'équipement terminal qui retourne les informations à l'initiateur. A la différence de la deuxième approche, il n'y a pas d'ouverture d'une nouvelle connexion spécifique à la gestion de la QdS entre équipements intermédiaires sur réception d'un paquet de signalisation ;
- la deuxième approche est illustrative des approches que l'on peut qualifier de « chaînées », dans la mesure où les équipements intermédiaires sont les destinataires des paquets et répondent à l'initiateur de la connexion avant d'être à leur tour l'initiateur d'une nouvelle connexion avec l'équipement suivant.

Notons que notre proposition d'architecture de signalisation s'insère dans le cadre de cette deuxième approche.

3.2 Description des protocoles

3.2.1 BGPv4

L'intérêt de BGP (*Border Gateway Protocol*) (Rekhter, 1995) vis-à-vis du contexte ciblé est double. D'une part, BGP est le standard des protocoles de routage inter domaine ; d'autre part, en modifiant légèrement son fonctionnement, certaines architectures de signalisation, que nous étudierons par la suite, l'utilisent pour offrir un service de gestion de la QdS en environnement multi domaines.

3.2.1.1 Présentation de BGP

BGP est le protocole standard de l'Internet pour les interconnexions entre opérateurs. Il fait partie de la famille des EGP (*Exterior Gateway Protocol*), (Mills, 1984), dont il est aujourd'hui le seul représentant réellement déployé. BGP est nécessaire pour un opérateur ou bien pour la gestion d'un point d'échange entre opérateurs.

La différence essentielle entre les protocoles EGP, comme BGP, et les protocoles utilisés à l'intérieur des AS, les IGP (*Interior Gateway Protocol*), (Gross, 1992), n'est pas technique mais plutôt administrative. Les IGP ne sont utilisés qu'à l'intérieur d'une entité (entreprise, association,…), où des décisions (comme la suppression ou bien l'ajout d'une route) peuvent être prises par un service unique. Le but des IGP est de trouver la route la plus efficace, en faisant confiance aux autres routeurs. Au contraire, les EGP s'utilisent entre entités distinctes (et même souvent concurrentes). Il n'y a plus de possibilité de prendre une décision qui s'imposera à tous et bien souvent un administrateur ne connaît pas ce que vont faire ses pairs.

La définition d'un AS est administrative et constitue l'unité de routage pour BGP ; c'est une entité où le pouvoir de décision est centralisé. Elle communique avec d'autres entités, ayant leurs propres AS et qui n'ont pas de relations hiérarchiques. Les autres entités peuvent être des fournisseurs de connectivité, aussi bien que des clients, parfois des pairs mais, dans tous les cas, il est impossible de leur dicter leur politique de routage. Entre les autres opérateurs, du trafic va pouvoir s'échanger sur la base d'un contrat.

Le classement hiérarchique entre AS se fait en fonction de « l'achat de transit »[10] qu'ils effectuent. Si un opérateur est suffisamment grand et a une envergure internationale, on parle de *Tier-1*. Tous les autres opérateurs, tout en ayant des clients et en échangeant avec leurs pairs, sont obligés d'acheter du transit pour combler les limites de leur réseau. Par exemple, un opérateur européen va devoir acheter du transit à un *Tier-1* pour permettre à ses clients d'accéder au Japon ou au Brésil.

3.2.1.2 Déploiement

BGP utilise le protocole de transport TCP (port 179). Deux entités BGP échangent des messages (PDU BGP) pour ouvrir et maintenir les paramètres d'une session BGP. Le premier flot de données est la table de routage BGP. BGP ne nécessite pas de mise à jour périodique des tables de routage ; celles-ci sont envoyées lorsque la table de routage change. En revanche, un routeur BGP doit retenir la totalité des tables de routage courantes de tous ses pairs durant le temps de la connexion.

BGP est basé sur l'échange de plusieurs messages, parmi lesquels les messages UPDATE, utilisés pour envoyer les informations de routage entre les entités BGP. Nous détaillerons plus particulièrement ce message dans la suite du paragraphe.

[10] Quantité de bande passante achetée.

3.2.1.3 Format des PDU

Chaque message possède un en-tête dont la taille est fixe. Il peut y avoir ou non une portion de données à la suite de l'en-tête.

MARKER	
LENGTH	TYPE

Marker : ce champ de 16 octets permet d'authentifier les messages BGP entrants, de détecter une perte de synchronisation entre deux périphériques BGP. Si le type du message est OPEN ou si le message OPEN ne fournit pas d'information d'authentification, alors tous les bits de Marker doivent être à 1, sinon, la valeur de ce champ est calculée suivant le mécanisme d'authentification utilisé.

Length : ce champ de 2 octets indique la taille totale du message, en-tête compris. La taille d'un message est au minimum de 19 octets et au maximum de 4096 octets.

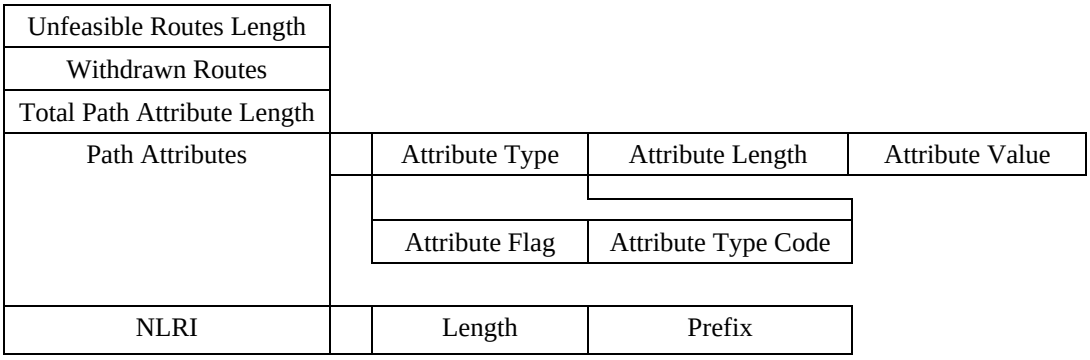
Type : ce champ de 1 octet définit le type de message envoyé :

Valeur du champ type	Nom	Description
1	OPEN	Ouverture de connexion
2	UPDATE	Mise à jour des tables
3	KEEPALIVE	Maintenance de la connexion
4	NOTIFICATION	Notification

Format du message UPDATE

Les messages UPDATE sont utilisés pour envoyer les informations de routage entre les entités BGP. Ces informations permettent de construire un graphe décrivant les relations entre les divers AS. Un message UPDATE permet de faire connaître une unique route utilisable et/ou d'annuler plusieurs routes à présent inutilisables.

Le message UPDATE contient toujours la partie en-tête et peut contenir les autres champs ci-dessous :



Unfeasible Routes Length : taille totale du champ Withdrawn Routes en octets. Une valeur de zéro indique qu'aucune route n'est devenue impraticable ; dans ce cas, le champ withdrawn routes n'apparaît pas dans le message UPDATE ;

Withdrawn Routes : champ de taille variable contenant la liste des préfixes d'adresses IP pour les routes devenues impraticables ;

Total Path Attribute Length : taille totale en octet du champ Path Attributes. Une valeur de zéro indique qu'il n'y a pas de champ Path Attributes dans le message UPDATE ;

Path Attributes : il s'agit d'un triplet de taille variable: Attribute Type, Attribute Length et Attribute Value. La partie Attribute Type du champ Path Attribute est composée de deux octets : l'octet Attribute Flag et l'octet Attribute Type Code.

Etudions plus en détail l'octet Attribute Flag. Le premier bit de poids fort définit si l'attribut est optionnel (mis à 1) ou déjà connu (mis à 0). Le terme exact pour un Attribut déjà connu est *Well-known*. Ces attributs sont reconnus par toutes les implémentations BGP. Certains de ces attributs sont dits Mandatory (obligatoire) et doivent être inclus dans tous les messages UPDATE. Les autres sont dits Discretionary (optionnels) et peuvent ou non être inclus dans les messages UPDATE. Tous les attributs *Well-known* doivent être propagés aux autres systèmes BGP. En plus des attributs *Well-known*, chaque chemin peut inclure un attribut optionnel.

NLRI est composé de deux informations :

- le champ Length qui indique la taille du champ Prefix. Une taille de zéro précise qu'un préfixe englobe toutes les adresses IP ;
- le champ Prefix qui contient les préfixes d'adresses IP suivis d'un nombre nécessaire de bits inutiles pour l'information et dont le seul but est de rendre la taille du champ Prefix multiple d'un octet (puisque le champ Length indique la taille en octet). Un message UPDATE peut au plus faire connaître une route, cette dernière pouvant être décrite par plusieurs attributs de chemin (Path Attributes). Tous les attributs de chemin sont appliqués aux destinations contenues dans le champ NLRI du message UPDATE.

Un message UPDATE peut distinguer plusieurs routes ; chacune de ces routes est identifiée par sa destination (préfixe IP) ce qui identifie sans ambiguïté cette route pour le système BGP.

3.2.1.4 Conclusion sur BGP

Les paragraphes précédents permettent de souligner un point important du déploiement des protocoles adaptés au multi domaines. En effet, BGP est déployé indépendamment de l'IGP choisi au niveau des domaines. La solution de routage inter domaine ne vient pas remplacer une solution retenue en mono domaine, mais vient la compléter tout en offrant un service de bout en bout.

Par la suite, nous verrons que ce principe de déploiement est retenu par les architectures de signalisation multi domaines qui ne remettent pas en cause les solutions développées en mono domaine mais les complètent.

Dans le cas où BGP est utilisé par des architectures de signalisation, il leur suffit d'implémenter de nouvelles fonctions dans les routeurs supportant BGP pour que les paquets BGP transmettent en plus des informations classiques de routages, les données de QdS qui les intéressent.

3.2.2 COPS

Le protocole COPS (*Common Open Policy Service*) n'a pas été conçu pour des besoins associés à un contexte multi domaines mais peut s'adapter à cet environnement. Notre proposition d'architecture de signalisation appartenant, d'un point de vue conceptuel, à la famille des architectures de signalisation multi domaines basées sur l'utilisation du protocole COPS, nous présentons dans ce paragraphe les principes de ce protocole.

3.2.2.1 Principes du protocole COPS

L'IETF définit le protocole COPS comme le protocole standard de transaction dans un type de réseau particulier : les réseaux à base de règles (*Policy Based Network* : PBN).

Réseaux à base de règles

La gestion de réseau à base de règles permet beaucoup plus qu'une automatisation de la configuration des équipements à partir d'une base de données, comme il est déjà possible de le faire dans certaines architectures.

Un serveur de règles interprète et combine les informations statiques enregistrées dans les bases de données circonstancielles pour prendre une décision et envoyer ses directives aux nœuds du réseau. La flexibilité ainsi obtenue permet d'appliquer des services personnalisés en temps réel supportant les nouvelles normes de QdS, les contraintes de sécurité ainsi que la politique de l'entreprise quant à l'utilisation du réseau.

La Figure 6 illustre l'architecture générale d'un réseau à base de règles. Le serveur de règles intégrant un PDP (*Policy Decision Point*) accède aux sources d'informations dont il dispose (serveur de temps, serveur d'annuaire,…) pour prendre une décision qu'il achemine ensuite vers le nœud du réseau intégrant le PEP (*Policy Enforcement Point*).

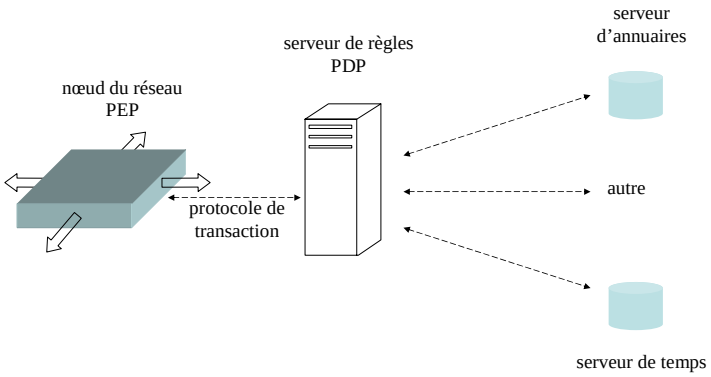


Figure 6 : Architecture générale d'un réseau à base de règles

Il existe deux modèles de gestion dans l'architecture de réseau à base de règles :

– le modèle *outsourcing* dans lequel le PEP fait appel au PDP pour répondre à un évènement qui doit être résolu sur la base de règles dont le PEP ne dispose pas. Ce cas de figure est typiquement associé aux protocoles de signalisation de bout en bout (type RSVP) ;

– le modèle *provisioning* qui suppose que le PDP prépare la configuration d'un équipement à partir de nouvelles règles d'utilisation du réseau (changement des règles de gestion, heure/date, expiration d'un quota,…). Le PDP charge ensuite les informations de configuration dans le PEP sans en avoir été sollicité. Ce modèle est surtout utilisé pour contrôler l'utilisation du réseau en l'absence de signalisation (contexte DiffServ par exemple).

Détaillons à présent les équipements de base de ces réseaux : les PDP et les PEP.

PDP

Le serveur de règles tient le rôle de PDP, il interagit avec les équipements du réseau dont il gère le comportement : routeurs, commutateurs, équipements VPN, pare-feux et autres PEP. Ceci suppose un grand nombre de fonctionnalités, illustrées par la Figure 7 et présentées dans la suite.

Interprétation des règles et prise de décision

Pour prendre sa décision, le serveur de règles interprète les données qu'il consulte dans les différentes bases : l'annuaire centralisé, pour les données statiques relatives à la politique d'utilisation du réseau mais aussi les données issues de divers équipements : serveurs RADIUS (*Remote Authentification Dial-in User Services*), DHCP, DNS, NTP. Il peut également recueillir des informations relatives à l'utilisation du réseau générées par des sondes de type RMON (*Remote MONitoring*). Les données topologiques du réseau et celles issues des SLA sont prises en compte.

A partir de cet ensemble de données, l'interpréteur du serveur de règles formule une décision en réponse (dans le modèle *outsourcing*) à la requête d'un PEP, qu'il lui communique sous forme de bail (ce qui caractérise la nature éphémère de la décision). Lorsque le bail se termine, le PEP doit en demander le renouvellement ou attendre qu'un nouvel événement se présente et émettre une autre requête vers le serveur.

Dans le cadre du modèle *provisioning*, le serveur de règles doit spontanément adapter le comportement des équipements du réseau aux évolutions qu'il peut constater, soit dans les règles à appliquer, soit dans les données issues du réseau.

Formulation de la réponse

Comme nous le verrons dans la suite, la réponse formulée par le serveur de règles est susceptible de s'adresser à de multiples équipements qui ne supportent pas forcément l'un des protocoles de transaction définis par l'IETF (COPS/Diameter). Dans ce cas, le serveur de règles doit générer des données de configuration bas niveau, spécifiques à chaque équipement. Elles sont ensuite acheminées vers les équipements par des moyens plus traditionnels : SNMP, RADIUS, une chaîne de commandes HTTP ou une ligne de commandes CLI (*Command-Line Interface*). L'utilisation d'un *proxy* est aussi une solution envisageable.

Gestion de classes de règles multiples

Le serveur de règles doit faire face à différentes classes de règles : sécurité, QdS, discrimination d'applications, authentification des utilisateurs et respect des règles inscrites dans l'annuaire centralisé.

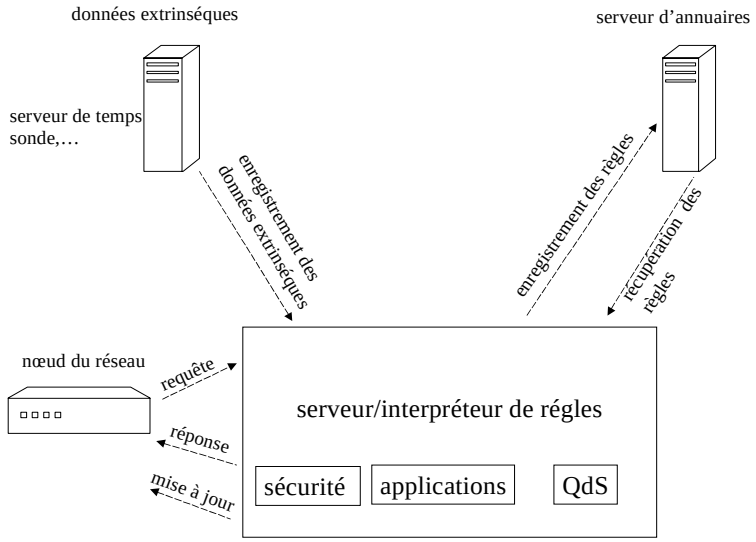


Figure 7 : Fonctionnalités du serveur de règles

PEP

Les décisions prises par le serveur de règles, dans le cadre du modèle "*Outsourcing*" ou "*Provisioning*", sont véhiculées vers les équipements du réseau chargés de les mettre en application. Ces équipements, qualifiés de PEP ou clients, mettent en œuvre les outils dont ils disposent pour faire respecter les décisions : le filtrage de paquets, la réservation de bande passante, la priorité de trafic, …

Comme nous l'avons remarqué précédemment, le serveur de règles doit pouvoir entrer en contact avec tout type de PEP, dont en particulier ceux qui, déployés avant les travaux de l'IETF, ne supportent pas de protocole de transaction ou utilisent un protocole de transaction légataire. Dans ces cas, l'utilisation d'un *proxy* doit faciliter leur intégration dans l'architecture du réseau à base de règles.

Le *proxy* doit remplir les tâches suivantes :

- maintenir à jour la liste des équipements dont il est responsable ;
- traduire un protocole légataire (SNMP, RADIUS) en protocole COPS/Diameter ;
- véhiculer les décisions du serveur de règles vers les PEP dépourvus de protocole de transaction ;
- surveiller l'état des équipements qui lui sont rattachés.

La Figure 8 résume l'ensemble des catégories de clients possibles.

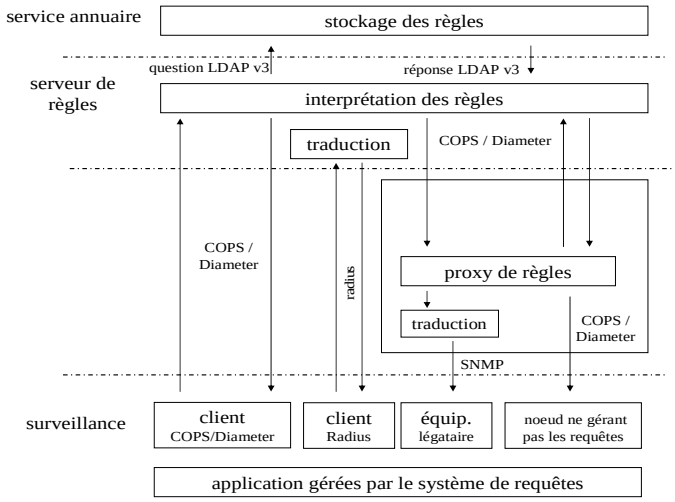


Figure 8 : Différentes catégories de clients

COPS

COPS est un protocole extensible et simple de question/réponse spécialement conçu pour l'échange d'informations entre le serveur de règles (PDP) et son client (PEP). Chaque message COPS renferme des objets d'informations dont le contenu est interprété en fonction du type de client COPS (par exemple le client RSVP). Chaque type de client est défini dans une RFC d'extension qui précise l'interprétation à donner aux objets[11].

Dans le modèle de base, illustré par la Figure 9, les communications consistent le plus souvent en un échange de requêtes et de décisions entre le client et le serveur. Le protocole est conçu pour véhiculer des objets auto identifiés qui contiennent les données nécessaires à l'identification des requêtes, à l'établissement du contexte, au référencement des configurations précédemment installées, au relais des décisions, au transfert des informations spécifiques à un type de client, à l'indication d'erreurs et à l'intégrité des messages.

Pour distinguer les différents clients, leur type est identifié dans tous les messages. Chaque type de client peut avoir des données spécifiques nécessitant différentes catégories de décisions. L'IETF en édite les directives d'usage dans des RFC d'extension : COPS-RSVP, COPS-PR (pour COPS PRovisioning).

Avant tout échange de données, le PEP doit ouvrir une connexion TCP (port 3288) vers son PDP responsable et s'identifier. Il peut alors générer sa requête à laquelle le PDP répondra par un message de décision. Lorsque le PEP a installé la nouvelle configuration, il doit en avertir le PDP. Le serveur de règles peut ensuite modifier ou mettre à jour les informations de configuration par de nouveaux messages de décision. A chaque décision, le PEP efface la configuration nommée et renvoie une confirmation au PDP.

[11] Notons que COPS n'est pas le seul protocole recommandé par l'IETF : voir par exemple Diameter basé sur RADIUS que nous ne décrirons pas car ne présentant pas, du point de vue conceptuel, de grande différence avec COPS.

Le contexte de chaque requête dépend de ce qui l'a déclenchée. COPS identifie quatre types d'événements susceptibles de déclencher une requête :

- l'arrivée d'un message (contrôle d'admission) ;
- l'allocation de ressources locales ;
- le relais d'un message ;
- une demande d'informations de configuration donnée.

Le PDP Local (LPDP), représenté sur la Figure 9 est optionnel. Il peut être utilisé par le PEP dans certaines circonstances pour appliquer une décision locale (perte de la connexion entre le PEP et le PDP, attente de décision trop longue, …) mais le PDP conserve dans tous les cas son autorité. Ce qui veut dire qu'une décision locale doit être transmise au PDP dès que possible par un objet de décision LPDP que le serveur de règles confirmera ou remplacera.

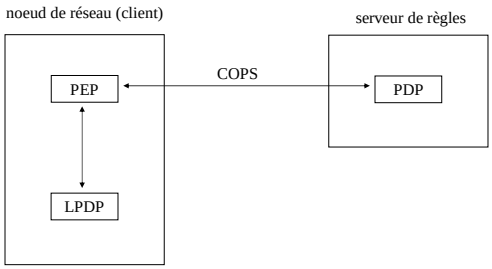


Figure 9 : Modèle de base du protocole COPS

3.2.2.2 Déploiement de COPS dans un contexte multi domaines

Plusieurs propositions d'architecture de signalisation pour la QdS en environnement multi domaines s'appuient sur le protocole COPS (ou sur des protocoles analogues offrant le même service de transaction). Son principe d'utilisation est alors le suivant :

- si comme nous l'avons vu dans le paragraphe précédent, le protocole COPS est utilisé à l'intérieur d'un domaine entre les PEP et les PDP, une autre utilisation consiste à l'employer entre des *Bandwidth Brokers* (BB) appartenant à des domaines différents. La gestion des ressources inter domaines est centralisée sur les BB et la gestion intra domaine est laissée libre et est indépendante de la solution retenue en inter domaines (utilisant ou non COPS) ;
- lors de l'ouverture d'une connexion, chaque BB est alors considéré tour à tour comme client PEP et serveur PDP et les requêtes de QdS sont alors transmises entre les BB via COPS.

La Figure 10 illustre ces principes. Les données échangées entre les PEP / PDP concernent le descriptif des services : type de services disponibles, disponibilités des services,… ; ceci correspond au *Service Level Specification* (SLS) d'un SLA classique passé entre un client et son fournisseur de service. Dans le cas où les BB s'occupent également de la réservation des ressources, nous retrouvons généralement, entre les routeurs de bordures (entrant : $R_{ingress}$ ou sortant R_{egress}) et les BB, des protocoles spécifiques à cette tâche comme ceux proposé par le groupe *Resource Allocation Protocol*,

RAP, basés sur les PDU de requête de service *Resource Allocation Request*, RAR, et de réponse *Resource Allocation Answer*, RAA.

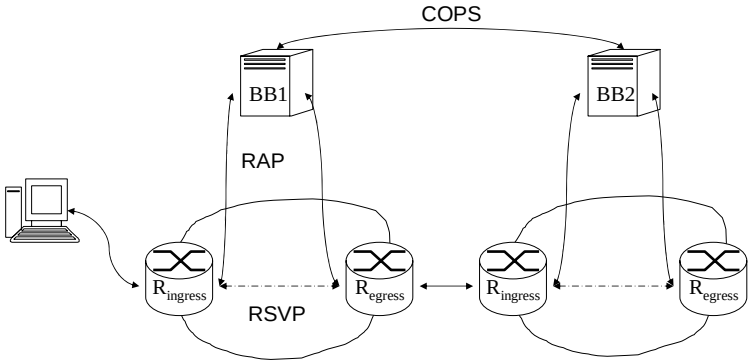


Figure 10 : Gestion par COPS de la QdS multi domaines

3.3 Architectures de signalisation en multi domaines

3.3.1 Introduction

Ce paragraphe présente les principales architectures de signalisation basées sur les protocoles vus précédemment (ou qui se déploient de façon analogue).

A la différence des architectures de communication, les travaux présentés dans ce paragraphe sont plus récents : il n'existe pas d'implémentations réelles de ces architectures mais uniquement des propositions de modèle de conception et des tests en simulation.

De plus, ces architectures se focalisent généralement chacune sur un aspect précis de la problématique multi domaines ; le choix des points traités s'expliquent souvent par l'origine des concepteurs des architectures : opérateurs des télécommunications qui ont une vue orientée accords/contrats ou équipes de projets positionnées en tant que concepteurs avec une vue orientée protocoles.

3.3.2 Description des architectures

Les critères de présentation des architectures décrites ici sont :

- les objectifs ciblés. Nous ferons un parallèle avec les problèmes soulevés dans le paragraphe (3.1), à savoir : (1) la définition homogène des services commune à tous les AS, (2) l'acquisition dynamique de la disponibilité et les caractéristiques des services présents dans les AS traversés, (3) la gestion des réservations entre domaines adjacents ;
- le principe de déploiement : de type « diffusif » ou « chaîné » ;
- les hypothèses (environnements IP ciblés, mécanismes sur lesquels repose l'architecture, …). Notons que ces architectures ne reposent pas sur les mêmes hypothèses car elles ne ciblent pas les mêmes objectifs et donc font abstraction de problèmes différents.

Les limites de ces architectures seront discutées de façon comparative au paragraphe 4.2.

Architecture Aquila

Objectifs

L'architecture de signalisation proposée dans le cadre du projet IST-Aquila (dont l'architecture de communication a été décrite au paragraphe (2.2.1)) a été conçue pour offrir un service d'acquisition de la disponibilité et des caractéristiques des services présents dans les AS traversés, ce qui donne un élément de réponse au deuxième problème soulevé dans le paragraphe (3.1).

Principe de déploiement

C'est une architecture de type «diffusif » basée sur le protocole *Border Gateway Reservation Protocol*, BGRP (Pan, 2000) (ou la dernière version BGRP+/BGRPP pour BGRP Plus (Salsano, 2001)). BGRP est déployé entre les équipements de bordure compatibles avec BGP. Ce protocole distingue deux messages :

- les messages *PROBE*, qui sont générés par le premier domaine et transmis au travers des routes BGP d'un routeur de bordure à un autre. Les mécanismes déclenchés par ce type de message permettent de contrôler la conformité de la requête avec les SLA, et de garder en mémoire la route pour que les messages *GRAF*T passent en retour par la même route ;
- les messages *GRAFT*, qui permettent d'indiquer la disponibilité du service.

La Figure 11 illustre le déploiement de l'architecture de signalisation Aquila.

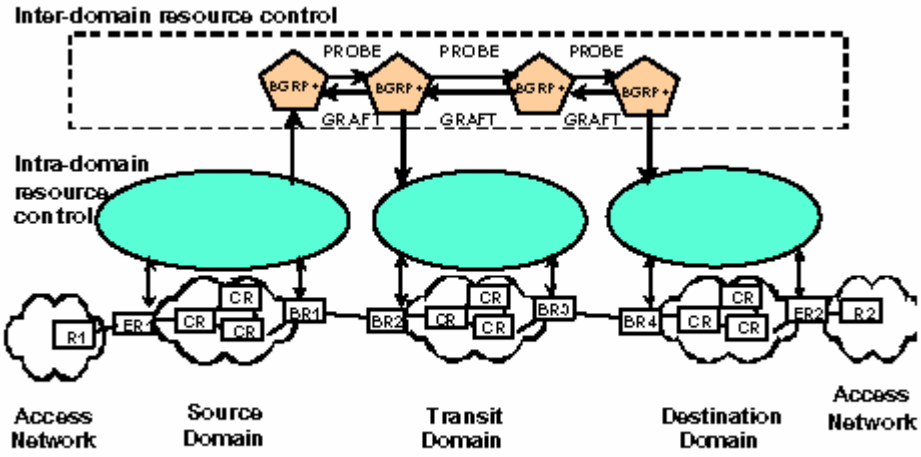


Figure 11 : Architecture de signalisation Aquila.

A titre indicatif, le Tableau 2 présente une partie des champs des messages *PROBE* et *GRAFT* ainsi que les données disponibles dans chaque routeur de bordure, appelé BR sur la Figure 11.

Message PROBE	Message GRAFT	Informations d'un BR
émetteur : numéro AS, IP du BR	émetteur : numéro AS, IP du BR	IP du prochain routeur BGRP
identifiant du message PROBE	identifiant du message GRAFT	IP du précédent routeur BGRP
identifiant de GWKS	destination : IP	débit réservé pour chacune des routes
destination	débit réservé	
débit demandé	…	…
chemin		
…		

Tableau 2 : Champs des messages BGRP et données présentes sur un BR

Hypothèses

Cette architecture est basée sur l'utilisation de services connus (type IntServ, DiffServ,…) dont les performances sont supposées être connues. Ces services sont identifiés dans le champ GWKS (*Globally Well Known Services*) des messages *PROBE*.

Architecture Tequila

Objectifs

Issue des travaux réalisés dans le cadre du projet du même nom (voir paragraphe 2.2.2), l'architecture TEQUILA (Christallo, 2002) a pour objectif d'acquérir des informations sur les performances (délai de transit, gigue, taux de perte,…) des liens inter AS, ce qui répond au deuxième problème évoqué dans le paragraphe (3.1). Elle ne propose pas de solution pour la gestion de la réservation des ressources.

Principes de déploiement

C'est une architecture de type «diffusif ». Par conception, elle n'impose pas une politique de gestion de la QdS particulière en intra domaine même si les travaux du projet TEQUILA sont basés sur l'approche DiffServ au niveau IP. Les informations relatives à un lien sont contenues dans un attribut libre des en-têtes des paquets BGPv4, appelé QOS_NLRI (Jacquenet, 1999), (Jacquenet, 2004b), présenté dans le Tableau 3 :

Nom du champ	Taille en octets
QoS Information Code	1
QoS Information Sub-code	1
QoS Information Value	2
QoS Information Origin	1
Address Family Identifier	2
Subsequent Address Family Identifier	1
Network Address of Next Hop	4
Network Layer Reachability Information	variable

Tableau 3 : Champs de l'attribut QOS_NLRI des paquets BGPv4

Les deux premiers champs sont, pour notre contexte, les plus intéressants. Ils permettent de mettre à jour des informations concernant les caractéristiques d'un lien en terme de : gigue, taux de

paquets perdus par unité de temps, délai de propagation (minimum, maximum, moyen), bande passante disponible, …

Hypothèses

La principale hypothèse sous-jacente à la proposition concerne la possibilité de mettre à jour l'attribut QOS_LNRI par les équipements concernés lorsqu'ils voient passer des paquets BGP. Pour cela, (Jacquenet, 2004a) propose un nouveau client type pour le protocole COPS, qui permettrait que lorsqu'un routeur client (PEP) traite un paquet concerné, son PDP lui indique de mettre à jour l'attribut QOS_NLRI.

Architecture NAIS

Objectifs

L'objectif de l'architecture NAIS, *Network Architecture for Inter domain Services*, (Füzesi, 2003), est de connaître à tout instant la quantité de trafic attribuée par lien inter AS pour chacune des classes de service supportées. Elle permet donc de répondre au deuxième problème évoqué dans le paragraphe (3.1) en proposant des mécanismes pour connaître les disponibilités des ressources.

Principes de déploiement

NAIS est une architecture de type «diffusif ». Elle repose sur un protocole de signalisation, qui, sur le modèle de BGP, permet une découverte des caractéristiques des services disponibles au sein des AS par l'échange de messages contenant la description des services (*Service Description* : SD). Ces messages se regroupent en deux types :

- nom du service, marquage correspondant et caractéristique du service : délai, pertes ;
- quantité de service disponible.

Des mécanismes périodiques de mise à jour permettent de connaître l'état d'utilisation des services.

Cette architecture concerne davantage les ISP que les utilisateurs, car elle ne permet pas la prise en compte d'une requête pour un flux particulier d'une application ; elle est utile à deux ISP lorsqu'ils souhaitent négocier des accords de passage sur leurs AS respectifs.

Le choix d'une technologie réseau n'est pas défini mais pour la première phase, qui consiste en un échange des services disponibles, il est nécessaire que les services soient standard : par exemple CL, GS d'IntServ, Premium, AS pour DiffServ.

Hypothèses

La mise en place de cette architecture repose sur les hypothèses suivantes :

- les opérateurs passent des accords pour des agrégats de flux et non pour des flux particuliers ;
- la mise en place d'un service nouveau peut prendre plusieurs semaines mais l'utilisation de ce service va s'étendre sur une période longue de plusieurs mois ;
- il n'y a qu'un petit nombre de services définis, ce qui limite le nombre d'accords inter opérateurs ;

- les contrats entre opérateurs concernent des trafics de quelques dizaines de Megabits/s mais le trafic effectif à un instant donné ne représente qu'une petite utilisation des réservations ;
- les trafics à QdS sont minoritaires.

Architecture 2-Tier

Objectifs

L'objectif de l'architecture Two-Tier - *Two-Tier Differentiated Services Architecture* (Reichmeyer, 1998), (Terzi, 1999a), (Terzi, 1999b) est de proposer le cadre d'une architecture fournissant de la QdS de bout en bout tout en ayant deux niveaux de mécanismes déployés simultanément : au niveau d'un AS et entre les AS. Ceci suppose un accord stable et bilatéral entre les administrateurs de domaines adjacents, mais n'implique pas une politique commune de gestion de la QdS intra domaine.

L'allocation de ressource se fait par flux en intra domaine alors qu'elle est faite par agrégat en inter domaines. Le point clé de la proposition est de vérifier que les ressources demandées par le routeur de bordure du premier AS puissent bien être obtenues dans les autres AS au regard des accords passés. L'architecture Two-Tier propose donc une solution au troisième problème posé dans le paragraphe (3.1).

Principes de déploiement

Two-Tier est une architecture de type « chaîné ». Elle a été définie dans le cadre des travaux réalisés par la communauté Internet-2 sur le Qbone[12].

Le problème de la réservation de ressource est décomposé en deux niveaux distincts et indépendants, à savoir :

- au niveau intra domaine, les choix de protocoles sont laissés libres : COPS, COPS + RSVP, … ;
- au niveau inter domaines, les mécanismes sont basés sur des messages RAR/RAA[13] entre les BB et les routeurs de bordure et sur le protocole COPS (ou bien un protocole analogue) entre les BB.

Hypothèses

Two-Tier impose de mettre en place des mécanismes particuliers dans les équipements de bordure en accord avec les décisions prises au niveau des BB pour assurer une continuité de service lors du passage d'un AS à l'autre. Ces mécanismes sont :

- du *shaping* sur les routeurs de sortie, pour que le trafic respecte le profil prévu dans le contrat ;
- du *policing* sur les routeurs d'entrée, pour surveiller le profil des trafics entrants.

[12] Voir le site http://qbone.internet2.edu/ pour plus de renseignements sur les activités de la communauté Internet2.

[13] Déjà introduits au paragraphe (3.2.2).

Projet Eurescom

Eurescom[14], Institut Européen pour la recherche et études stratégiques en télécommunications, a été fondé en 1991 par les principaux opérateurs de télécommunications. Il est basé en Allemagne.

Objectifs

Le projet Eurescom, par sa forte orientation opérateurs de télécommunications[15], a permis de dégager des points incontournables dans l'élaboration d'architectures de signalisation. Les travaux décrits dans (Eurescom, 1999), (Eurescom, 2000) montrent une grande hétérogénéité dans la définition des services et soulignent la nécessité de proposer une interface commune de description des services entre les opérateurs pour offrir des garanties de QdS de bout en bout. Eurescom propose donc une solution au premier des problèmes soulevés au paragraphe (3.1). Il propose également un modèle de transaction entre les ISP en préconisant l'utilisation d'un protocole standard comme COPS.

Principes de déploiement

Concernant la partie définition des services, (Eurescom, 2000) propose une spécification de SLS standard à échanger entre deux opérateurs pour la définition de leurs services. La Figure 12 représente un extrait d'une proposition de SLS (Eurescom, 2001).

Nous ne reprenons pas l'intégralité de cet extrait de description, mais nous pouvons y noter :

- la date de validité du contrat ;
- les destinations accessibles ;
- le nom du service avec un descriptif de la QdS : pertes, délais ;
- le profil de trafic accepté.

Concernant la partie architecture, Eurescom propose davantage un modèle d'architecture qu'une architecture de signalisation à proprement parler. (Ebata, 2000) définit un modèle de transaction des SLS entre les ISP (basé sur COPS). L'architecture instanciée en conséquence est de type « chaîné ».

Hypothèses

Le groupe Eurescom ne proposant pas une architecture de signalisation mais un modèle de conception, il n'y a pas vraiment d'hypothèse de déploiement. C'est à la charge des concepteurs d'architecture d'effectuer les choix afférents aux solutions retenues (IntServ / RSVP, DiffServ, …).

[14] Voir le site http://www.eurescom.de pour une description plus détaillée du groupe.

[15] Les partenaires des projets décrits ici sont : *British Telecommunications* plc, *eircom* plc, *Telefonica* S.A., *Telia* AB, *Telekom Austria* AG.

Identification Service Name: “P1008 Virtual Leased Line” [S] Version Number: 1 [S] Service Description: “This service …“ [S] Service PDB: “EF-PHB” [S] Seller Identifier: “VLL Plc” [S] Buyer Identifier: “Network Provider Plc” [B] **Validity Period** Start Date: “1 March 2001” [B] End Date: “1 March 2002” [B] **Traffic Identification** Identification Header Fields: DESTINATION_ADDRESS_FIELDS & PROTOCOL_PORT_NUMBER [S->B] Destination Address: “134.148.156.0” [B] Destination Address Mask: “255.255.255.0” [B] Protocol-Port Range Protocol: “TCP” [S -> B] Source Port Start: UNSPECIFIED Source Port End: UNSPECIFIED Destination Port Start: 1000 [B] Destination Port End: 1050 [B] **Traffic Profile** Traffic Profile Algorithm: Leaky_Bucket [S -> B]	MTU Size: 1500 [S ->B] Mean Rate: 64 KB/s [S -> B] Burst Size: 1.2 MB [S] Time Interval: 10 Sec [S] **QoS** Delay Descriptor Delay Priority: UNSPECIFIED Mean Delay: “50 ms” [S] Mean RTT: “105 ms” [S] Loss Descriptor Loss Priority: UNSPECIFIED Mean Loss: 10-9 [S] Throughput Mean Throughput: “64 KB/s” [S] **Service Reliability** Availability: 99% [S] Scheduled Maintenance Period Start Time of Day: “02:00” [S] End Time of Day: “03:00” [S] Day of Week: Sunday [S] Monitored Parameters Delay Measurement Frequency: “1 Minute” [S->B] Delay Reporting Frequency: ”30 Minutes” [S->B] Delay Threshold: “100 ms” [S -> B]

Figure 12 : Description d’un SLS échangé entre un ISP « vendeur » : S et un autre « acheteur » : B

BMP Architecture

Objectifs

L’objectif de la proposition BMPA - *Bandwidth Management Point Architecture* (Fernandez, 2001), (Mantar, 2001) (Okomus, 2001) et (Hwang, 2003) est de définir une architecture pour la réservation de ressources (intra domaine entre deux routeurs de bordure et inter domaine entre deux équipements intermédiaires appelés BMP). En cela, BMPA répond au troisième point évoqué au paragraphe (3.1).

Principes de déploiement

BMPA est une architecture de type « chaîné ». Son déploiement repose sur l’utilisation d’un protocole à définir sur le modèle du protocole SIBBS - *Simple Inter domain Bandwidth Broker Signaling* (Qbone, 2001). Le principe de déploiement est représenté sur la Figure 14 et la Figure 13.

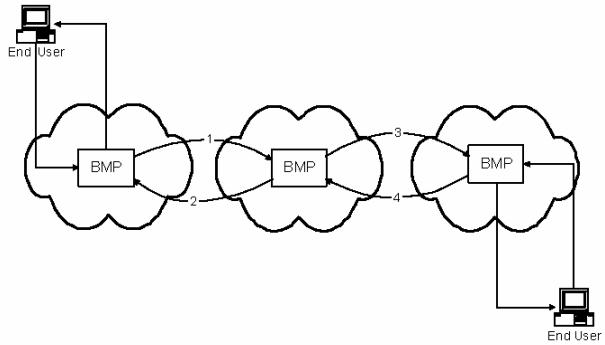


Figure 13 : Signalisation de bout en bout utilisant le protocole SIBBS

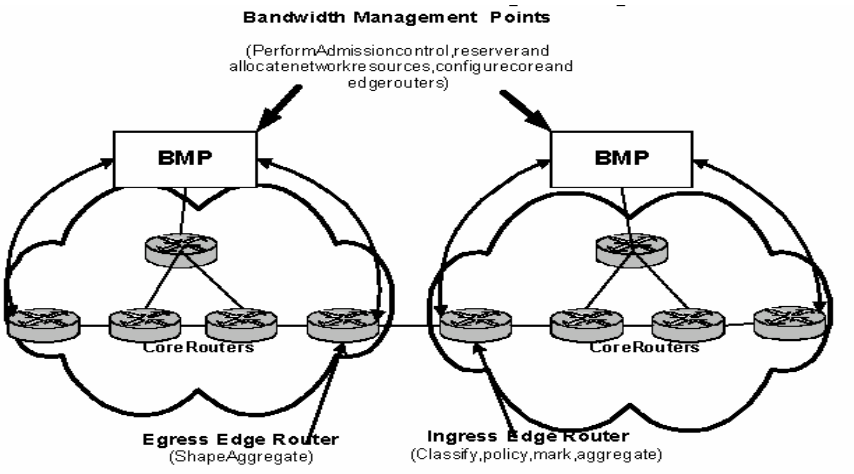


Figure 14 : BMP Architecture

Hypothèses

Ne proposant qu'un protocole de réservation de ressources, cette architecture doit s'appuyer sur d'autres mécanismes pour que les BMP connaissent les services disponibles dans chaque AS ainsi que l'état d'utilisation de ces services en bordure des AS.

4 Conclusion

Ce chapitre a présenté un état de l'art des propositions associées à la problématique ciblée dans nos travaux, à savoir :

- les applications distribuées dans l'Internet présentant des besoins en QdS, et plus spécifiquement les applications multimédias ;
- les propositions d'évolutions (en conséquence) de l'Internet vis-à-vis :
  - des protocoles de Transport ;
  - des modèles de services IP et des architectures de déploiement associées ;
  - des architectures de communication à QdS ;
  - des protocoles de signalisation pour le multi domaines ;
  - des architectures de signalisation pour le multi domaines.

La discussion qui suit est structurée en trois étapes au cours desquelles nous rappellerons les objectifs des propositions d'évolutions de l'Internet, puis nous en soulignerons les limites, pour enfin positionner nos travaux de thèse.

4.1 Objectifs des propositions

Le Tableau 4 présente d'une manière synthétique les principaux objectifs des propositions d'évolution de l'Internet (hors partie architecture de signalisation) vis-à-vis de la QdS. Les travaux relatifs à la couche Application sont également résumés.

		Objectif	Exemples
Plan de communication	Application	Définir les besoins des applications	ITU G10-10, HSTPN, SIP, SAP
	Transport	Offrir plus de fonctionnalités de bout en bout	SCTP, DCCP, FPTP
	IP	Offrir des services évolués	IntServ, DiffServ
	Architecture	Intégrer une partie des 3 couches précédentes	Aquila, Tequila, XQoS
Plan de signalisation m-d[16]	Services	Donner un standard de définition des services	SLS inter domaines
	Protocole	Définir un modèle de transaction pour échanger les caractéristiques des services	COPS, SIBBS
		Définir un modèle de protocole de réservation	SIBBS, BGRP

Tableau 4 : Rappel des objectifs des propositions

Concernant les architectures de signalisation, le Tableau 5 en présente un comparatif suivant l'intégration ou non :

- d'un modèle de définition des services ;
- d'un modèle de protocole de transaction entre AS pour la découverte des services ;
- de mécanismes et protocoles de réservation des ressources.

	NAIS	Tequila	Eurescom	BMP-A	Aquila	2-Tier
Proposition de définition des services	NON	NON	OUI	NON	NON	NON
Proposition d'un modèle de protocole(s) de transaction	OUI	OUI	OUI/NON	NON	OUI	OUI
Proposition de mécanismes et protocoles de réservations	NON	NON	NON	OUI	NON	OUI

Tableau 5 : Comparatif des différentes architectures de signalisation

4.2 Limites des propositions

Au niveau Application, les travaux initiés pour caractériser les besoins des applications restent insuffisants. En effet, l'effort de l'ITU pour proposer des paramètres standard n'inclut aucune sémantique de garantie pour ces paramètres.

Au niveau des architectures de communication, aucune des propositions n'intègre les solutions proposées aux niveaux IP et Transport, alors que chacun de ces niveaux a un impact sur la QdS fournie de bout en bout.

Sur le plan de la signalisation multi domaines, nous pouvons faire ressortir deux limites principales :

[16] multi domaines

- aucune des propositions ne prend en compte le triple problème lié à la gestion des services dans le contexte multi domaines ;
- aucune de ces propositions n'est intégrée dans un système global de communication. Cela signifie pour l'utilisateur final un accroissement de complexité, dans le sens où il devra gérer indépendamment les deux plans, celui de la communication et celui de la signalisation.

4.3 Notre proposition

Notre proposition de système de communication cible les différents points suivants :

- sur le plan communication, notre proposition intègre les nouveaux services IP et Transport (offerts respectivement par l'approche DiffServ et le protocole FPTP) de façon à garantir une QdS de bout en bout selon plusieurs sémantiques possibles ;
- sur le plan de la signalisation multi domaines, nous adressons sur quatre points :
  - la proposition d'une définition des services via une description statistique des services d'un AS, qui vient compléter les propositions actuelles ;
  - la proposition d'un protocole de découverte et de rapatriement des caractéristiques des services des différents AS impliqués au cours d'une communication sur un équipement unique ;
  - la proposition d'un protocole de réservation des ressources multi domaines ;
  - la proposition d'une architecture de ces protocoles intégrée à l'architecture de communication issue du premier point.

Les hypothèses sur lesquelles nous basons nos travaux sont les mêmes que celles présentées dans (Füezesi, 2003), à savoir :

- il n'y a qu'un petit nombre de services définis ;
- les trafics à QdS sont minoritaires ;
- les sessions à QdS ont une durée de plusieurs minutes à quelques heures.

Les deux premières vont dans le sens d'une limitation de la charge des équipements de type BB en terme de quantité d'information à mémoriser.

La dernière autorise que l'établissement de la connexion durant les négociations inter domaines puisse être de l'ordre de la seconde.

Chapitre 2 Environnements IP ciblés et architecture de communication

Ce chapitre a trois objectifs distincts. Le premier est de présenter les modèles d'environnement DiffServ, successivement mono et multi domaines, considérés dans cette thèse, ainsi que les hypothèses sur lesquelles nous nous appuierons par la suite pour soutenir notre proposition d'architecture. Le deuxième objectif est de présenter l'architecture de communication dans laquelle s'insère notre proposition. Nous nous focaliserons en particulier sur son interface de programmation (API). Le dernier objectif est de présenter les principales expérimentations attenantes dont les conclusions sont importantes pour la suite de nos travaux.

Le chapitre est organisé de la façon suivante. Dans un premier paragraphe, nous présentons les modèles de l'Internet que nous avons retenus, successivement de type DiffServ mono domaine puis multi domaines. Dans un second paragraphe, nous présentons l'architecture de communication en détaillant son principe de fonctionnement et son déploiement. Dans le troisième paragraphe, nous décrivons son API (*Application Programming Interface*). Dans le quatrième paragraphe, nous présentons les campagnes d'expérimentation réalisées dans le cadre d'un projet national pour valider cette architecture. Le dernier paragraphe conclut ce chapitre en ouvrant sur les limitations liées à cette architecture, introduisant ainsi les chapitres suivants.

1 Environnements IP ciblés

Ce paragraphe présente les modèles de l'Internet que nous considérons successivement, à savoir un Internet offrant des services différenciés sur le modèle du WG DiffServ dans un contexte tout d'abord mono domaine puis multi domaines.

1.1 Environnement DiffServ mono domaine ciblé

Le modèle d'environnement Internet ciblé pour le contexte mono domaine a été initialement défini dans le cadre du projet RNRT @IRS[17]. Sa mise en œuvre a conduit à la constitution d'une plate-forme nationale, le @IRSBone, interconnectant plusieurs sites locaux (LAAS, LIP6, INRIA Sophia, …). Le lecteur pourra se référer à (Garcia, 2002), (Chassot, 2002) pour une description plus exhaustive

[17] Le projet @IRS, Architecture Intégrée de Réseaux et Services, est un projet du Réseau National de la Recherche en Télécommunications – décembre 1998 - février 2001 visant à développer et à analyser de nouveaux mécanismes et protocoles pour l'Internet (IPv6, QdS DiffServ, multicast, mobilité,…) dans un environnement de réseaux hétérogènes (ATM, Ethernet, sans fil, …). Les partenaires de ce projet étaient Aérospatiale, CNRS-LAAS, CNET, INPG-LSR, INRIA Sophia Antipolis, RENATER, 6 WIND, LIP6, LSIIT. Le projet a été prolongé jusqu'en février 2003 dans le cadre d'une seconde phase (projet @IRS++).

de la plate-forme. Dans la suite de ce paragraphe, nous présentons le modèle retenu au travers du @IRSbone.

Le @IRSbone distingue au niveau IP trois classes de services que nous détaillerons par la suite :

- la classe (des paquets marqués) EF (*Expedited Forwarding*) qui donne accès au service GS (*Guaranteed Service*)[18] ;
- la classe AF (*Assured Forwarding*) qui donne accès au service AS (*Assured Service*) ;
- la classe DE (*Discard Eligibility*) qui donne accès au service BE (*Best Effort*).

Comme tout domaine DiffServ, le @IRSbone distingue deux éléments du réseau particulièrement importants :

- les routeurs de cœur (Rc) ou nœuds internes à un domaine ;
- les routeurs de bordure (Rb) ou nœuds frontaliers.

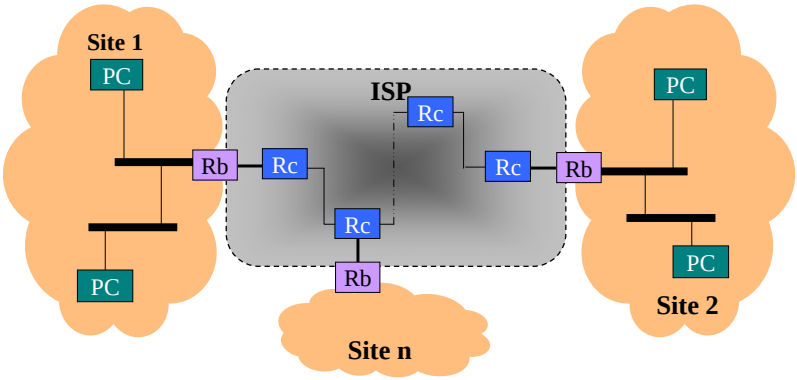


Figure 15 : Modèle d'environnement Internet pour le mono domaine

1.1.1 Fonctions d'un routeur de bordure

Les services IP peuvent être activés par des requêtes envoyées au routeur d'extrémité. Du point de vue de ce routeur, la disponibilité de chaque service (AS ou GS) est fonction :

- du contrat de trafic (TCA : *Traffic Contract Agreement*) négocié de façon statique avec l'ISP ;
- de l'utilisation courante du service.

Indépendamment du service choisi, le contrôle de trafic appliqué par le routeur de bordure est effectué par flux. Par conséquent, pour respecter le TCA local, il est nécessaire de mettre en œuvre un contrôle d'admission pour les services AS et GS. Dans le contexte d'un environnement mono domaine, ces mécanismes de contrôle d'admission sont implantés dans les routeurs de bordure.

Les autres fonctions du routeur sont les suivantes :

- un conditionnement de trafic par flux, comportant :

[18] Cette dénomination peut porter à confusion avec le service proposé par IntServ. Cependant, cette appellation a été décidée dans le cadre du projet @IRS et a été conservée par la suite.

- une classification par flux (MF : *Multi-Field*) à partir de plusieurs champs du paquet IP (identificateur de flux et adresse source) ;
- un marquage DiffServ du paquet en fonction de la QdS requise pour le flux considéré ;
- un contrôle de la conformité du trafic au profil annoncé par l'application, avec marquage « hors profil » des paquets dits « opportunistes » (dépassant le profil autorisé) pour la classe AS, et rejet des paquets GS en cas de non-conformité du trafic GS ;

– un ordonnancement des paquets couplant priorité stricte (PQ[19]) pour les paquets EF et priorité pondérée (WFQ[20]) entre les paquets AF et DE. Ce type d'ordonnancement est préconisé par le groupe DiffServ.

Pour réaliser les fonctions précédemment décrites, cet équipement dispose d'un métreur (*meter*), d'un espaceur (*shaper*), d'un marqueur (*marker*) et d'un destructeur (*dropper*) de paquets.

1.1.2 Fonctions d'un routeur de cœur

Indépendamment du service à mettre en œuvre, le contrôle de trafic par les routeurs de cœur est appliqué par classe (les routeurs de bordure ayant effectué le marquage), que les paquets appartiennent ou non à un même flux. Plus précisément, les fonctions mises en œuvre par les routeurs de cœur sont les suivantes :

– une classification de type BA (*Behavior Aggregate*) : les paquets de même marque sont rangés dans une même file ; il y a donc trois files ;
– un ordonnancement des trois files précédentes couplant des mécanismes de priorité stricte : PQ et de priorité pondérée : WFQ ;
– un contrôle de congestion (uniquement en AS), destiné à détruire les paquets opportunistes en cas de saturation de l'ISP ; ce contrôle inclut en particulier le mécanisme de rejet sélectif de type PBS (*Partial Buffer Sharing*) : voir (Hamma, 1997) pour plus de détails sur ce mécanisme.

Comparativement à d'autres approches, telles que (Bianchi, 2002) et (Benameur, 2002), le contrôle d'admission des flux à servir en AS ou GS est basé sur l'hypothèse que le TCA établi avec tous les sites locaux est tel que la quantité de trafic correspondant aux différentes classes ne dépasse pas à chaque instant la quantité totale admissible pour la classe considérée dans les routeurs de cœur. Sous cette hypothèse, le contrôle d'admission peut être appliqué uniquement en bordure de domaine. Nous reviendrons sur cette hypothèse dans le cadre du multi domaines.

---

[19] PQ : Priority Queuing

[20] WFQ : Weighted Fair Queuing

Les Figure 16 et Figure 17 illustrent les fonctionnalités de ces deux éléments :

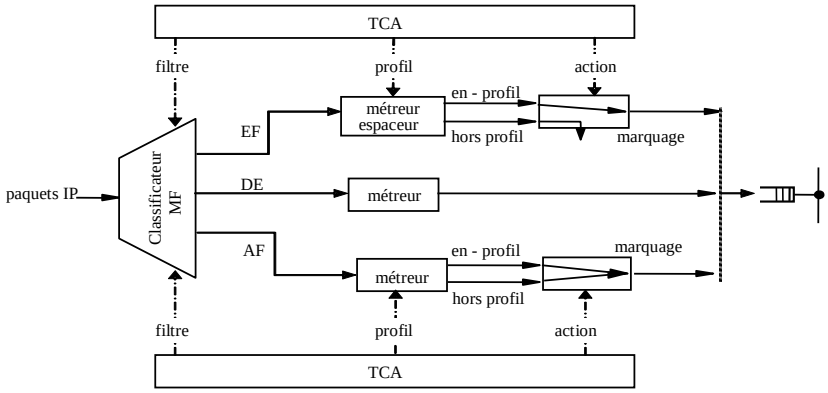


Figure 16 : Interface d'entrée des routeurs de bordure

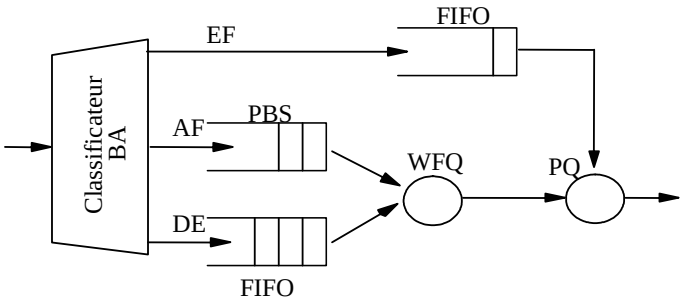


Figure 17 : Interfaces de sortie des routeurs de bordure et de coeur

1.2 Environnement DiffServ multi domaines ciblé

L'environnement Internet ciblé pour le contexte multi domaines est illustré Figure 18.

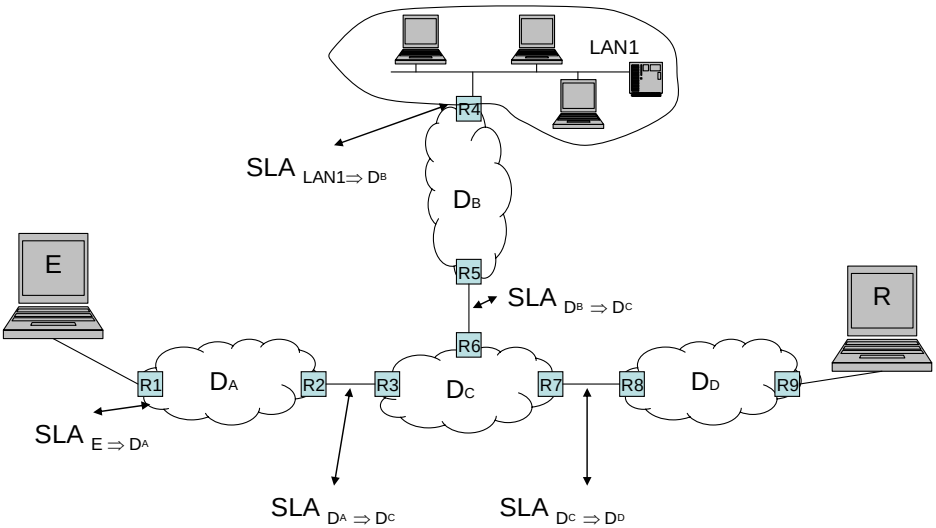


Figure 18 : Environnement multi domaines

Comme il a été introduit au Chapitre 1, le contexte du multi domaines fait apparaître des problèmes nouveaux dans la gestion de la QdS, pour lesquels les solutions envisagées en mono domaine ne suffisent plus, notamment pour ce qui concerne le contrôle d'admission (envisagé en bordure uniquement dans le cas mono domaine), en raison du problème que pose le dimensionnement d'un domaine au travers duquel des agrégats de différents domaines transitent, dans notre exemple le domaine D_C.

1.2.1 Hypothèse relative au dimensionnement d'un domaine de transit

Considérons un contrat (SLA : *Service Level Agreement*) établi entre l'opérateur gérant le domaine D_A et un client E (Figure 18). Pour chacune des classes de services IP, ce contrat fixe la quantité de trafic et le modèle de trafic (débit moyen, *Token bucket*, …) que E est autorisé à introduire en entrée de D_A via le routeur de bordure R1.

Ce contrat précise que le service est garanti uniquement entre un point d'entrée du domaine et un point de sortie, soit dans notre exemple entre R1 et R2 si l'on suppose que E souhaite émettre du trafic à destination de la machine R.

Pour garantir une QdS de bout en bout, il est donc nécessaire de mettre en place des mécanismes spécifiques, qui vont prendre en compte le fait que la communication doit traverser plusieurs domaines (ici, D_A, D_C, D_D).

Pour cela, l'opérateur du domaine D_A a également passé un contrat avec l'opérateur du domaine D_C. Ce contrat précise la quantité de trafic que D_A est autorisé à émettre sur D_C à travers le routeur de bordure R3. Il est établi de manière quasi statique, dans le sens où le contrat s'étend sur une période assez longue (plusieurs mois). Comme dans le cas mono domaine et des contrats entre un opérateur et un client, les contrats entre opérateurs précisent le modèle de trafic utilisé pour vérifier la conformité du trafic et les mesures prises en cas de dépassement. Dans notre exemple, il est nécessaire qu'il y ait également un contrat établi entre les domaines D_C et D_D.

Le problème de dimensionnement des liens du domaine D_C vient du fait qu'il peut y avoir une concentration de trafic issu de plusieurs domaines clients et devant passer par le même routeur de sortie. Dans notre exemple, sur le routeur R7 du domaine D_C, le trafic peut provenir du domaine D_A et du domaine D_B.

Dans ce cas, deux hypothèses peuvent être envisagées au niveau de D_C.

- La première hypothèse consiste à ce que l'opérateur administrant D_C dimensionne ses liens avec les opérateurs adjacents (et établit un contrat en conséquence) de telle que sorte qu'à tout instant, tout le trafic le traversant puisse sortir vers un opérateur quelconque voisin. Étudions les conséquences de ce choix sur notre exemple, la Figure 19 illustrant les contrats établis par D_C en considérant que le modèle de description du trafic ne comporte que le débit moyen.

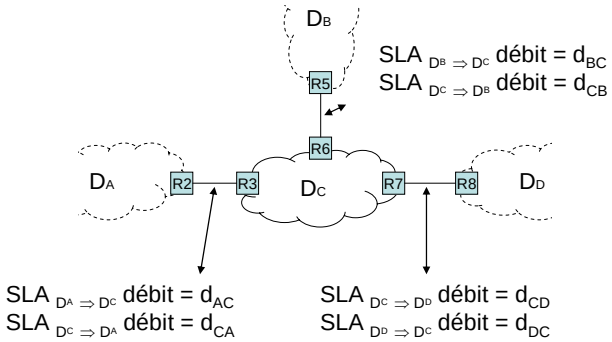


Figure 19 : Exemple de contrat

Si l'opérateur de D_C veut pouvoir respecter tous ses contrats à chaque instant, il doit donc configurer ses liens de sortie (soit R3, R6, R7) de telle sorte que :

$$d_{CA} = d_{BC} + d_{DC} \; ; \; d_{CB} = d_{AC} + d_{DC} \; ; \; d_{CD} = d_{AC} + d_{BC}$$

Dans le cas général, pour un opérateur d'un domaine D_{A0} ayant *n* domaines adjacents $D_{A1}..D_{An}$, nous avons donc :

$$d_{A_0A_i} = \sum_{j \in \{\{1,\ldots,n\}-\{i\}\}} d_{A_jA_0} \quad \text{pour tout } i \in \{1,\ldots,n\}$$

On conçoit aisément que cette solution est très pénalisante car elle demande à chaque opérateur de surdimensionner ses liens entre domaines adjacents. En contrepartie, une fois les contrats établis, il n'y a pas besoin de mécanismes complexes de signalisation pour chaque flux ; la gestion de la QdS de bout en bout reposant sur l'établissement de ces contrats, il suffit seulement de vérifier sur le tout premier routeur de bordure que les flux respectent leur profil de trafic et de sanctionner le trafic excédentaire.

– La deuxième hypothèse est que l'offre d'un opérateur ne porte que sur l'accès à son domaine, mais pas sur l'ensemble des domaines rencontrés. Ceci signifie que les accords entre opérateurs se basent sur une estimation du maximum de trafic, mais ne prennent pas en compte la relation précédente. Dans notre exemple, cela signifie que pour le client E, le débit accordé à son flux est garanti uniquement jusqu'au routeur R2. Comme dans le cas mono domaine, l'opérateur n'a à configurer que ses équipements internes en fonction de ses clients mais pas en fonction du trafic des autres opérateurs. Dans le cas où une communication passe par plusieurs opérateurs, il convient alors de mettre en place un protocole de signalisation pour vérifier la disponibilité des ressources dans chaque domaine, puis (si possible) de les réserver pour garantir de bout en bout la QdS. Cette approche est bien évidemment moins coûteuse pour les opérateurs car elle ne les oblige pas à surdimensionner les liens de transit, mais en contrepartie repose sur une architecture de signalisation indispensable pour vérifier les disponibilités et faire les réservations dans les domaines traversés.

C'est sous cette seconde hypothèse que s'inscrit notre proposition d'architecture de signalisation.

1.2.2 Fonctions des routeurs inter domaines

Quelle que soit l'hypothèse retenue, des mécanismes de *policing* et de *shaping* sont indispensables sur les routeurs situés entre deux domaines adjacents afin que le trafic émis par un domaine client respecte bien le contrat (sinon il risque d'être pénalisé).

La Figure 20 distingue deux cas :

- celui d'un routeur situé entre un client émetteur (réseaux locaux ou machine isolé) et un opérateur ;
- celui d'un routeur situé entre deux opérateurs, ce routeur pouvant être considéré soit comme un routeur de sortie (*egress*), soit comme un routeur d'entrée (*ingress*).

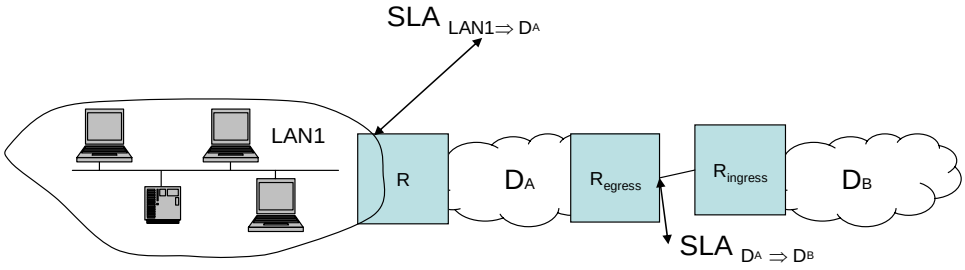


Figure 20 : Routeurs inter domaines

Dans le premier cas, sur l'interface d'entrée du routeur R, il y aura un *policing* par flux et sur l'interface de sortie un *shaping* par agrégat.

Dans le second cas, nous retrouverons du *shaping* par agrégat sur l'interface de sortie de R_{egress}, et du *policing* par agrégat sur l'interface d'entrée de $R_{ingress}$.

Notons qu'il n'y a pas de contrainte particulière sur l'implémentation des routeurs de bordure du côté du récepteur.

2 Architecture de communication

L'architecture de communication ici décrite a été conçue dans le cadre des travaux de (Chassot, 2002), (Garcia, 2001 et 2002) pour fournir aux applications des garanties de QdS pour le transfert de chacun de leur flux (par exemple, pour le flux audio et pour le flux vidéo dans le cas d'une application de visioconférence). Différents services et protocoles sont disponibles dans cette architecture :

- les services de niveau IP permettent de prendre en compte des critères temporels et de fiabilité ;
- les services de niveau Transport permettent, de façon complémentaire, de considérer des contraintes de synchronisations inter/intra–flux et des contraintes de fiabilité partielle ;
- une API permet aux programmeurs d'application de formuler des requêtes de QdS par le biais de primitives et de paramètres mis à leur disposition en faisant abstraction de la complexité des protocoles de communication sous-jacents.

Le choix d'un couple protocole de Transport / classe de service IP en adéquation avec la requête d'une application est réalisé (si la requête est recevable) de façon automatique par le système de communication. Les mécanismes conçus pour réaliser ce choix seront détaillés au Chapitre 3.

2.1 Contexte et objectifs de l'architecture

Cette architecture a été conçue dans le cadre du projet @IRS dont l'un des objectifs était de déployer une architecture de communication prenant en compte de façon disjointe les aspects QdS, mobilité et multicast. Nous ne nous intéressons ici qu'à l'aspect QdS du projet.

2.2 Proposition d'architecture au niveau des hôtes

L'hypothèse sous-jacente à la définition de l'architecture (que nous appellerons @IRS dans la suite) rejoint celle de plusieurs autres dédiées au transport de flux multimédias (Nahrstedt, 1994), (Campbell, 1994), (Chassot, 1996), à savoir que le trafic applicatif peut être décomposé en plusieurs flux ayant chacun des besoins spécifiques en terme de QdS.

Sur ces bases, l'architecture @IRS (illustrée Figure 21) définit la notion de *canal de bout en bout à QdS*, un tel canal étant une association de niveau Transport (donc un couple d'adresses IP et de numéros de ports source et destination) à laquelle est retenue une certaine QdS dont les paramètres seront explicités dans le paragraphe décrivant l'API.

La requête de QdS est formulée par l'application lorsqu'elle réclame la création d'un canal. Pour cela, l'application doit fournir le profil de trafic qu'elle compte générer via le canal ainsi qu'un certain nombre de paramètres de QdS.

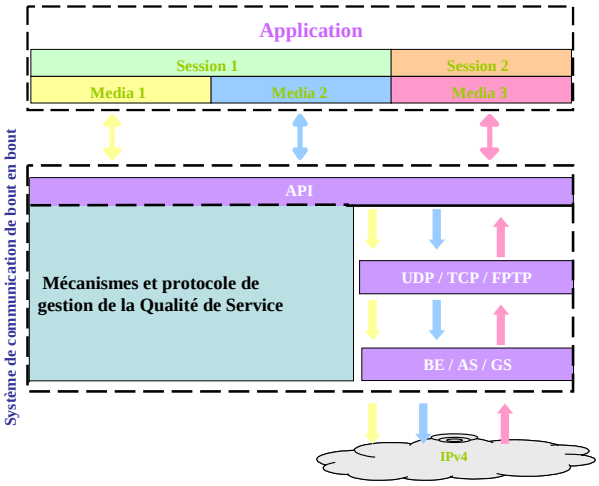


Figure 21 : Architecture @IRS

2.2.1 Notion de session

Telle que définie jusqu'ici, l'architecture @IRS permet de fournir une QdS par canal ; cependant, nous avons vu dans le chapitre 1 que les besoins en QdS des applications ne se limitaient pas à la seule QdS à fournir à chaque média, mais que la dépendance entre les données pouvait entraîner des besoins en synchronisation (ordre partiel) entre les différents canaux. Ce point ne pouvant être réglé au travers des canaux, un objet de plus haut niveau doit être défini. L'architecture @IRS introduit pour chaque application un contexte de communication appelé *session* qui est défini comme l'ensemble des canaux établis par l'application plus un ordre partiel entre les canaux.

2.2.2 Services de niveau Transport

De bout en bout, le système de communication gère trois protocoles de niveau Transport, offrant les garanties de QdS suivantes :

- une garantie totale sur l'ordre et la fiabilité offerte par le protocole TCP ;
- une garantie de fiabilité et d'ordre partiels intra flux et/ou inter flux offerte par le protocole FPTP issu des travaux de (Exposito, 2003) et qui a été présenté dans le Chapitre 1 ;
- aucune garantie d'ordre ni de fiabilité offerte par le protocole UDP.

2.2.3 Services de niveau IP

Si la spécification de l'architecture au niveau des hôtes ne nécessite pas de choix particulier quant au système de fourniture de QdS à utiliser, sa mise en œuvre en est indissociable. En effet ce choix influe sur les services que peut fournir l'architecture et sur les mécanismes mis en œuvre pour obtenir la QdS. Deux propositions de systèmes de fourniture de QdS IP auraient pu être envisagées lors de la définition de l'architecture @IRS, les solutions IntServ et les solutions DiffServ.

Le choix des solutions DiffServ a été motivé par deux aspects :

- les limites sur le déploiement à grande échelle des solutions basées sur IntServ pour fournir la QdS ;
- la possibilité de fournir une QdS par flux applicatif sur les hôtes d'extrémité tout en offrant une QdS par agrégat en cœur de réseau.

La fourniture d'une QdS par flux à partir des propositions DiffServ est adressée dans le paragraphe suivant et sera validée par les mesures réalisées sur la plate-forme @IRS qui ont été étendues lors de simulations (présentées dans le chapitre 3).

Trois services ont été définis au niveau IP :

- le service BE (*Best Effort*) n'offre aucune garantie de QdS ;
- le service garanti GS (*Guaranteed Service*), analogue au service *Premium* (Nichols, 1999), a été retenu pour les flux applicatifs à fortes contraintes de temps et de fiabilité. Les applications ciblées par ce service sont celles ne tolérant pas (ou difficilement) des variations de délai ou de débit ;

– le service assuré AS (*Assured Service*) a été retenu pour répondre au besoin des flux réactifs n'ayant pas de trop fortes contraintes de délai, mais requérant une bande passante minimale. Un flux servi en AS dispose d'une bande passante minimale garantie pour la partie de son trafic respectant la caractérisation de trafic formulée pour le flux considéré. La partie de son trafic excédant la caractérisation est véhiculée en AS tant qu'aucune congestion n'intervient sur le chemin emprunté par le flux.

3 Interface de programmation d'application (API)

3.1 Paramètres de QdS

Au regard de (Campbell, 1994) et (Gopalakrishma, 1995), la définition des paramètres de QdS varie d'une proposition à l'autre s'adaptant souvent aux types des applications ciblées. L'objectif est ici de proposer une API ayant des paramètres et des sémantiques de garantie suffisamment génériques pour que toutes les applications puissent utiliser la même API. Pour cela, les paramètres de QdS retenus sont les suivants :

– un ordre partiel intra flux permettant d'exprimer des contraintes de synchronisation intra flux,
– une fiabilité partielle et un délai de transit, exprimés en termes de :
  - pourcentage minimal des paquets émis que l'application souhaite recevoir : τ_r,
  - pourcentage minimal des paquets émis que l'application souhaite recevoir dans l'intervalle de temps]0,b] : (τ_d,]0,b]),
  - nombre maximal de pertes consécutives admissibles.
– un ordre partiel inter flux, permettant à l'application d'exprimer des contraintes de synchronisation logique entre des flux différents.

La motivation du choix de ces paramètres réside dans le fait qu'ils permettent de prendre en compte n'importe quel type d'application.

A titre d'exemple, une requête (d'une application de type DIS ayant des contraintes temporelles fortes) telle que : « tous les paquets applicatifs doivent être reçus, 90 % d'entre eux devant l'être avec un délai de transit inférieur à 50 ms » sera ainsi paramétrée de la manière suivante :

$\tau_r = 1$, $\tau_d = 0.9$ et b = 50ms.

Le débit est considéré comme l'expression d'une caractéristique du trafic (au travers d'un modèle tel que, par exemple, le *Token Bucket*), plutôt que l'expression d'un paramètre de QdS. Il apparaît donc dans les paramètres de service de l'API relatifs à la caractérisation du trafic.

Notons que du point de vue des programmeurs, fournir une valeur à ces paramètres peut s'avérer difficile. Nous présenterons dans le Chapitre 5 les mécanismes d'un module d'adaptation permettant de faire correspondre aux paramètres de l'API des paramètres compréhensibles par les programmeurs (tels que, pour une visioconférence, le nombre d'images par seconde, la taille des

images, …) et les sémantiques de garantie (telles que fluidité optimale, dégradations passagères acceptées, …) pour chaque famille d'application.

3.2 Sémantiques de garantie

Comparativement aux protocoles TCP et UDP qui offrent respectivement une garantie totale et aucune garantie sur l'ordre et la fiabilité, nous introduisons d'autres sémantiques de garantie comme le proposait déjà au niveau de la spécification (Dantine, 1992). Ainsi, les paramètres de QdS sont couplés avec l'une des trois sémantiques suivantes :

- une garantie absolue, notée A,
- une garantie en moyenne avec notification en cas de dégradation, notée N,
- une garantie en moyenne, notée M.

3.2.1 La garantie absolue : A

La garantie A signifie que la valeur du paramètre de QdS sur laquelle elle s'applique est obtenue « de façon certaine », par exemple par un mécanisme de réservation de ressources au niveau IP ou par un mécanisme de retransmissions « autant de fois que nécessaire » au niveau Transport. Aucune incertitude n'apparaît donc dans le respect des paramètres requis par l'application si le système de communication a accepté la requête.

Un exemple d'applications concernées par la sémantique A est celui des applications de simulation interactive distribuée. Pour ces applications, un flux de données événementielles (tir, détonation, …) peut par exemple être servi avec ce type de garantie tant sur τ_r que sur τ_d, car une altération de la fiabilité ou du délai de transit associé au transfert d'une de ces données est extrêmement préjudiciable à la cohérence de la simulation.

3.2.2 La garantie en moyenne : M

La garantie M signifie que la valeur du paramètre de QdS sur laquelle elle s'applique peut être obtenue, non plus mécaniquement, mais au regard de critères « statistiques » caractérisant le système de communication, en termes de délai et de fiabilité. Une incertitude apparaît donc dans la garantie requise par l'application, même si le système de communication a répondu positivement à la requête. La précision de ces critères est donnée dans le paragraphe (3.2.2).

Un exemple d'applications concernées par la sémantique M est celui des applications de visioconférence, qui peuvent réclamer une telle sémantique tant sur τ_r que sur τ_d (au moins pour le flux vidéo) car l'utilisateur final est susceptible de tolérer quelques altérations de la vidéo.

3.2.3 La garantie en moyenne avec notification : N

Comme pour la garantie M, la garantie N signifie que la valeur du paramètre de QdS sur laquelle elle s'applique peut être obtenue au regard de critères statistiques caractérisant le système de communication en terme de délai et de fiabilité. Comme précédemment, une incertitude apparaît donc dans la garantie requise par l'application, mais à la différence de la sémantique M, toute dégradation de qualité doit être notifiée à l'utilisateur.

Un exemple d'applications concernées par la sémantique N est celui des applications dites « adaptatives », conçues pour s'adapter automatiquement aux fluctuations du réseau.

3.3 Primitives

Au travers d'une session, l'application peut établir un ou plusieurs canaux de communication de bout en bout entre un émetteur et un récepteur, chacun étant (1) unicast, (2) dédié au transfert unidirectionnel d'un seul flux de données, et (3) offrant une QdS spécifique au flux véhiculé.

Pour cela, elle dispose d'une API unique et générique, offrant les primitives de service suivantes :

- openSession(String sessionName).
- int openChannel(String sessionName , String remoteAddress ,
  int dataPort , String role , qosClass QoS).
- sendData(String sessionName , int flowNumber , byte[] Data).
- byte[] receiveData(String sessionName , int flowNumber).
- closeChannel(String sessionName , int flowNumber).
- closeSession(String sessionName).

Nous décrivons ci-après chacune de ces primitives en en précisant les paramètres.

<u>Ouverture / Ferneture d'une session</u>

La primitive *openSession()* permet à l'application de réclamer la création d'une session et de lui associer un nom (s*essionName*). Au travers de cette session, elle peut alors établir autant de canaux que nécessaire. Par exemple, dans une session de vidéoconférence, nous retrouverions deux canaux, l'un véhiculant les données vidéo, l'autre véhiculant les données audio.

La primitive *closeSession()* permet à l'application de fermer une session. Tous les canaux de la session sont alors automatiquement fermés.

<u>Ouverture / Fermeture d'un canal</u>

La primitive *int openChanne()* permet à l'application de créer un canal à QdS spécifique (*QoS)*, appartenant à la session *sessionName*.

Le champ *Dataport* est utilisé en émission et en réception pour l'échange des données.

Côté entité Émettrice, l'application précise :

– *role* : sender,
– *remoteAddress* : l'adresse IP du récepteur.

Côté entité Réceptrice, l'application précise :

– *role* : receiver,
– *remoteAddress* : l'adresse IP de l'émetteur.

Dans les deux cas, l'entité applicative (émettrice ou réceptrice) se voit retourner un numéro de canal libre (champ *flowNumber* des primitives d'émission et de réception) qui sera ensuite utilisé pour le transfert des données.

La primitive *closeChannel()* permet de fermer le canal ayant pour numéro *flowNumber* et appartenant à la session *sessionName*.

Envoi / Réception d'une donnée

La primitive s*endData()* (resp. *byte[] receiveData()*) permet à l'application d'envoyer une donnée (resp. de recevoir une donnée) appartenant au flux véhiculé sur le canal *flowNumber* de la session *sessionName*.

Note : L'appel à la primitive *openChannel()* entraîne l'activation de deux protocoles : le protocole de création d'un canal et le protocole de gestion de la QdS. Nous décrirons ces protocoles au Chapitre 4.

3.4 Règles d'enchaînement des primitives

Du côté de l'entité émettrice (resp. réceptrice), l'application doit faire appel aux primitives dans l'ordre suivant :

	openSession()	
openChannel()	openChannel()	…
sendData() (receiveData)	sendData() (receiveData)	…
closeChannel()	closeChannel()	…
	closeSession()	

4 Mesures de performances

Nous présentons dans ce paragraphe les principales mesures de performances réalisées sur la plate-forme @IRS. Le lecteur pourra se référer à (Chassot, 2002), (Garcia, 2001 et 2002) pour plus de détails sur ces expérimentations[21].

Ces mesures avaient deux objectifs :

[21] L'auteur tient à préciser que si les mesures ici présentées sont extraites de (Garcia, 2002), il a cependant activement contribué à leur réalisation durant son stage de DEA et son début de thèse.

– mesurer les performances du système déployé de fourniture de QdS DiffServ ;
– valider l'utilisation de DiffServ pour fournir aux applications des garanties de QdS par canal de communication.

4.1 Plate-forme et outils de mesure

Dans ce paragraphe, nous présentons tout d'abord la plate-forme expérimentale utilisée pour ces mesures (4.1.1) et les outils utilisés lors de ces mesures (4.1.2).

4.1.1 Plate-forme expérimentale

Les mesures présentées ci-après ont été réalisées entre le LAAS-CNRS et le LIP6[22] dans l'environnement IPv6 illustré sur la Figure 22 représentant une partie de la plate-forme @IRS.

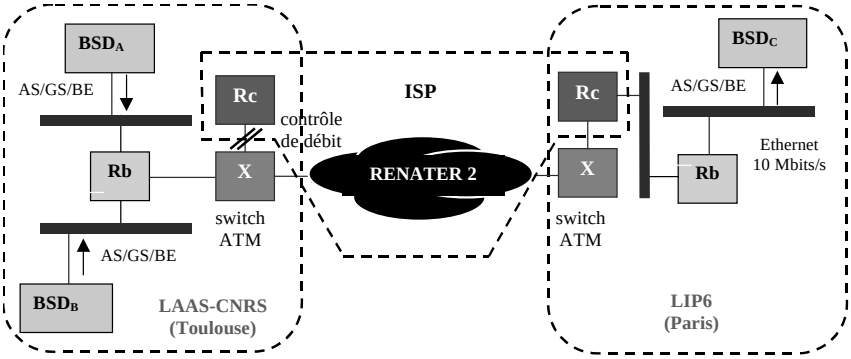


Figure 22. Configuration de la plate-forme d'expérimentation

La bande passante du lien (notée BPL par la suite) connectant les sites à l'ISP (VP ATM CBR) était telle que le débit maximal disponible au niveau UDP est de 107 Koctets/s pour des paquets de 1 Koctets (1024 octets).

Les routeurs de cœur et de bordure ont été configurés sous les hypothèses suivantes :

– la quantité maximale QM_{GS} de trafic GS que peut introduire le routeur de bordure (en moyenne) a été fixée à 20 Koctets/s, soit environ 20% de BPL ;
– la quantité maximale QM_{AS} de trafic AS que peut introduire le routeur de bordure (en moyenne) a été fixée à 40 Koctets /s, soit environ 40% de BPL ;
– les poids du WFQ appliqué entre paquets AF et DE sont de 0,5 et 0,5 ;
– le débit maximum autorisé par les routeurs de cœur est de 100 Koctets/s.

[22] Pour cette plate forme dans le cadre du projet @IRS, le LIP6 s'est essentiellement concentré sur les mécanismes de QdS déployés sur les routeurs FreeBSD et le LAAS-CNRS sur la spécification et l'évaluation des performances des services.

4.1.2 Outils utilisés

Pour réaliser ces mesures, nous avons utilisé un générateur de trafic développé au LAAS et au LIP6, nommé Debit6. Debit6 est utilisé pour :

- envoyer/recevoir des flux UDP/IPv6 sous Free BDS ou Windows NT ;
- émettre un trafic respectant un profil défini par un *token bucket* ;
- mesurer la bande passante et les pertes pour un exercice donné ;
- mesurer le délai de transit pour chaque paquet envoyé.

Le trafic émis par Debit6 est caractérisé par :

- la taille L (en octets) des paquets UDP émis ;
- le nombre N_B de paquets par rafale ;
- Le délai ΔT_B (en millisecondes) entre les rafales ;
- le nombre N_T de paquets émis au total ;
- le flow ID marquant les paquets.

Le débit moyen émit est donné par : $D=\frac{L*N_B}{\Delta T_B}$

Pour les mesures décrites ci-après, le trafic a été généré sous forme de rafales de 1 paquet UDP de 1024 octets. Le temps inter rafales a été utilisé pour faire varier le débit moyen en émission.

4.2 Spécifications des mesures

Deux séries de mesures ont été définies :

- la première série met en jeu un seul flux UDP à débit variable servi en AS ou GS, en concurrence avec un trafic BE. Ces mesures étaient destinées à valider la protection d'un *agrégat* AS ou GS par rapport à du trafic BE ;
- la deuxième série de mesures met en jeu plusieurs flux UDP servis en AS ou GS et à débit variable, en concurrence entre eux et avec un trafic BE. Ces mesures visaient à caractériser la QdS fournie à un *flux applicatif* (donc à un canal de bout en bout) et à déterminer l'impact du nombre de flux et de la charge du réseau sur celle ci.

4.2.1 Scénarios associés à la validation de la protection d'un agrégat

Cette série de mesures vise à valider la protection offerte à des flux AS et GS uniques, confrontés isolément ou ensemble à un trafic BE.

Trois scénarios ont été définis pour cette série de mesures :

- les deux premiers mettent en concurrence un unique flux AS (ou GS) et un flux BE. Le débit du flux servi en AS (ou GS) est fixé à 50% puis 100% du trafic maximal acceptable pour le service considéré (AS ou GS). Le trafic BE, nul pour le cas de référence, est augmenté à chaque mesure de 25% jusqu'à atteindre 100% de la bande passante totale disponible. Les Tableaux 6 et 7

résument ces deux scénarios en précisant si le routeur de bordure est en état de congestion ou non (la somme des trafics entrants étant alors supérieure à la bande passante disponible en sortie) ;

– le troisième scénario (Tableau 8) vise à évaluer l'impact du trafic AS (respectivement GS) sur la QdS GS (respectivement AS) lorsque le réseau est saturé ; pour cela, on se propose d'émettre un flux AS et un flux GS en parallèle, chacun à un débit de 50% du débit maximal autorisé pour le service associé, en présence d'un flux BE émis à 100% de la bande passante disponible.

<table>
<tr><td colspan="2" rowspan="2">Scénario 1</td><td colspan="5">BE (% de BPL)</td></tr>
<tr><td>0</td><td>25</td><td>50</td><td>75</td><td>100</td></tr>
<tr><td rowspan="2">AS (% du trafic AS max.)</td><td>50</td><td colspan="4">pas de congestion</td><td>congestion</td></tr>
<tr><td>100</td><td colspan="3">pas de congestion</td><td colspan="2">congestion</td></tr>
</table>

Tableau 6 : Scénario 1 - *Impact du trafic BE sur la QdS AS*

<table>
<tr><td colspan="2" rowspan="2">Scénario 2</td><td colspan="5">BE (% de BPL)</td></tr>
<tr><td>0</td><td>25</td><td>50</td><td>75</td><td>100</td></tr>
<tr><td rowspan="2">GS (%du trafic GS max.)</td><td>50</td><td colspan="4">pas de congestion</td><td>congestion</td></tr>
<tr><td>100</td><td colspan="4">pas de congestion</td><td>congestion</td></tr>
</table>

Tableau 7 : Scénario 2 - *Impact du trafic BE sur la QdS GS*

<table>
<tr><td colspan="2" rowspan="2">Scénario 3</td><td>BE (% de BPL)</td></tr>
<tr><td>100</td></tr>
<tr><td>AS (%du trafic AS max.)</td><td>50</td><td rowspan="2">congestion</td></tr>
<tr><td>GS (%du trafic GS max.)</td><td>50</td></tr>
</table>

Tableau 8 : Scénario 3 - *Impact du trafic AS sur la QdS GS*

4.2.2 Scénarios associés à la validation de la QdS offerte à un flux

Le but de la deuxième série de mesures est de valider l'hypothèse qu'il est possible de fournir une QdS par flux applicatif avec un système de fourniture de QdS de type DiffServ. En d'autres termes, il s'agit d'évaluer l'impact (que l'on espère faible) du nombre et de la charge des flux AS (ou GS) sur la QdS AS (ou GS).

Pour cela, nous avons défini trois scénarios :

– les deux premiers mettent en concurrence plusieurs flux utilisant le même service (AS ou GS) avec un ou plusieurs flux BE dont la somme des débits en émission correspond à la bande passante disponible en sortie du routeur de bordure. Les détails de ces deux scénarios sont donnés dans les Tableaux 9 et 10.

– le troisième scénario met en jeu un ou deux flux AS et un ou deux flux GS en concurrence avec deux flux BE. Il s'agit donc ici d'évaluer l'impact du nombre de flux AS (respectivement GS) sur la QdS GS (respectivement AS). Les détails de ce scénario sont donnés dans le Tableau 11.

Scénario 4	% de MQAS	Débit du trafic BE (% de BPL)
AS (BSDA)	50	100
AS1 (BSDA) AS2 (BSDB)	23 27	50 (BSDA) + 50 (BSDB)
AS11 (BSDA) AS12 (BSDA) AS21 (BSDB) AS22 (BSDB)	11 14 13 12	50 (BSDA) + 50 (BSDB)

Tableau 9 : Scénario 4 - *Impact des flux AS sur la QdS AS*

Scénario 5	% de MQGS	Débit du trafic BE (% de BPL)
GS (BSDA)	50	100
GS1 (BSDA) GS2 (BSDB)	23 27	50 (BSDA) + 50 (BSDB)
GS11 (BSDA) GS12 (BSDA) GS21 (BSDB) GS22 (BSDB)	11 14 13 12	50 (BSDA) + 50 (BSDB)

Tableau 10 : Scénario 5 - *Impact des flux GS sur la QdS GS*

Scénario 6	% de MQ (AS ou GS)	Débit du trafic BE (% de BPL)
AS (BSDA) GS (BSDB)	100 100	100 (50% BSDA - 50% BSDB)
AS1 (BSDA) AS2 (BSDB) GS1 (BSDA) GS2 (BSDB)	50 50 50 50	100 (50% BSDA - 50% BSDB)

Tableau 11 : Scénario 6 - *Impact des flux AS (GS) sur la QdS GS (AS)*

4.3 Résultats et analyse

Dans ce paragraphe, nous présentons les résultats et l'analyse des mesures effectuées pour chacun des scénarios précédents.

<u>Scénario 1 : impact du trafic BE sur la QdS AS</u>

Les résultats du scénario 1 sont présentés dans les Tableaux 12 et 13.

Dans les deux cas, ils sont conformes à ce que l'on pouvait attendre :

– quand le réseau n'est pas en état de congestion, on obtient une QdS caractérisée par :
  - un délai moyen presque constant (moins de 1ms d'écart) ;

- un débit en réception quasi égal à celui en émission ;
- un taux de pertes nul (ce qui indique que les différences entre débits sont dues à des délai dans les files d'attente du réseau) ;
- un délai maximum non borné ;

– quand le réseau est en état de congestion, on observe une légère augmentation du délai de transit ce qui est correct vis-à-vis du service attendu.

		BE (% de la bande passante)				
AS – 50%		0%	25%	50%	75%	100%
Délai de Transit (s)	- min	0.019	0.019	0.019	0.018	0.018
	- max	0.025	0.034	0.026	0.033	0.035
	- moyen	0.020	0.021	0.020	0.022	0.024
Débit (octets/s)	- émission	20483	20484	20484	20483	20483
	- réception	20483	20483	20483	20483	20483
Taux de pertes (%)		0	0	0	0	0

Tableau 12 : Résultats du scénario 1 (AS 50%)

		BE (% de la bande passante)				
AS – 100%		0%	25%	50%	75%	100%
Délai de Transit (s)	- min	0.019	0.019	0.019	0.019	0.019
	- max	0.045	0.032	0.030	0.033	0.036
	- moyen	0.020	0.021	0.021	0.024	0.024
Débit (octets/s)	- émission	40765	40963	40964	40963	40964
	- réception	40761	40963	40963	40963	40963
Taux de pertes (%)		0	0	0	0	0

Tableau 13 : Résultats du scénario 1 (AS 100%)

Scénario 2 : impact du trafic BE sur la QdS GS

Les résultats du scénario 2 sont présentés dans les Tableaux 14 et 15.

Ils sont similaires à ceux obtenus pour le scénario 1, avec un délai maximum légèrement inférieur pour les paquets GS à ce qu'il ne l'était pour les paquets AS.

Ces résultats sont corrects puisque l'on compare AS et BE d'une part et GS et BE d'autre part ; dans les deux cas, le flux qui bénéficie de QdS passe dans le réseau sans rencontrer de problème. Nous verrons par la suite que des flux AS et GS en concurrence ne bénéficient pas de la même QdS.

		BE (% de la bande passante)				
GS – 50%		0%	25%	50%	75%	100%
Délai de Transit(s)	- min	0.019	0.019	0.019	0.019	0.019
	- max	0.029	0.029	0.024	0.032	0.031
	- moyen	0.020	0.021	0.019	0.023	0.024
Débit (octets/s)	- émission	10243	10244	10243	10244	10244
	- réception	10243	10243	10243	10243	10243
Taux de pertes (%)		0	0	0	0	0

Tableau 14 : Résultats du scénario 2 (GS 50%)

		BE (% de la bande passante)				
GS – 100%		0%	25%	50%	75%	100%
Délai de Transit (s)	- min	0.018	0.019	0.019	0.019	0.019
	- max	0.024	0.029	0.031	0.032	0.033
	- moyen	0.019	0.020	0.022	0.023	0.025
Débit (octets/s)	- émission	18289	18289	18290	18289	18289
	- réception	18289	18289	18289	18289	18289
Taux de pertes (%)			0.0	0.0	0.0	0.0

Tableau 15 : Résultats du scénario 2 (GS 100%)

Scénario 3 : impact du trafic AS (GS) sur la QdS GS (AS)

Les résultats du scénario 3 sont présentés (Figure 23) sous la forme de la fonction de répartition du délai de transit des paquets AS et GS ; les paquets BE n'apparaissent pas sur la figure car l'échelle du graphique ne le permet pas : le délai de transit moyen qu'il subit est de plusieurs secondes.

La différence de traitement entre les trafics AS et GS apparaît ici nettement : on constate en effet que le délai de transit maximum est bien plus important pour AS que pour GS.

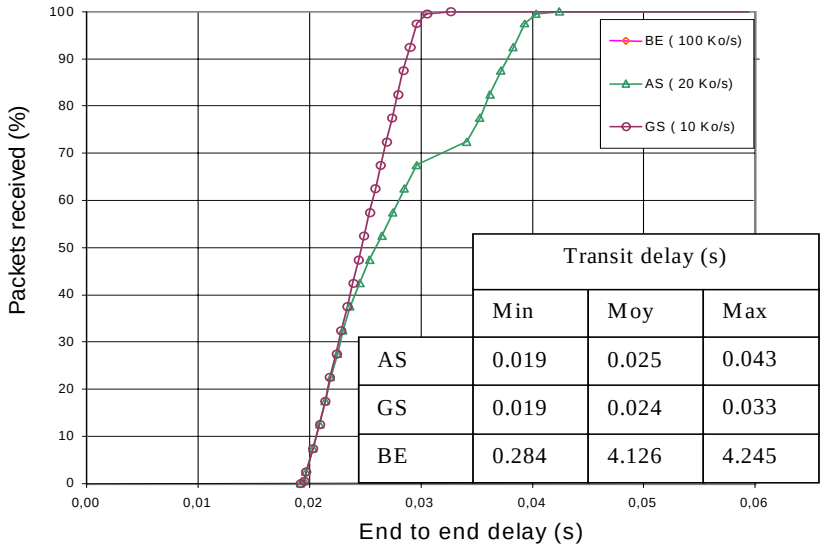


Figure 23 : Résultats du scénario 3

Le résultat des mesures des scénarios suivants (scénarios 4, 5 et 6) est donné au moyen :

- de la fonction de répartition du délai de transit ;
- du tableau indiquant le taux de pertes et les valeurs minimale, maximale et moyenne du délai de transit pour chaque flux.

Scénario 4 : impact des flux AS sur la QdS AS

Nous étudions à présent l'impact du nombre et de la charge des flux AS sur la QdS AS offerte à chaque flux.

Les résultats exposés dans le Tableau 16 montrent que cet impact est faible. En effet, le tableau indique une variation inférieure à 5 ms sur la valeur moyenne du délai de transit. Ce résultat

est conforté par la Figure 24 (fonction de répartition du délai de transit) qui met en évidence que le délai de transit est quasiment inchangé pour 90 % des paquets.

Note : la courbe nommée *reference flow* a été obtenue pour un flux AS seul dans un réseau sans autre trafic.

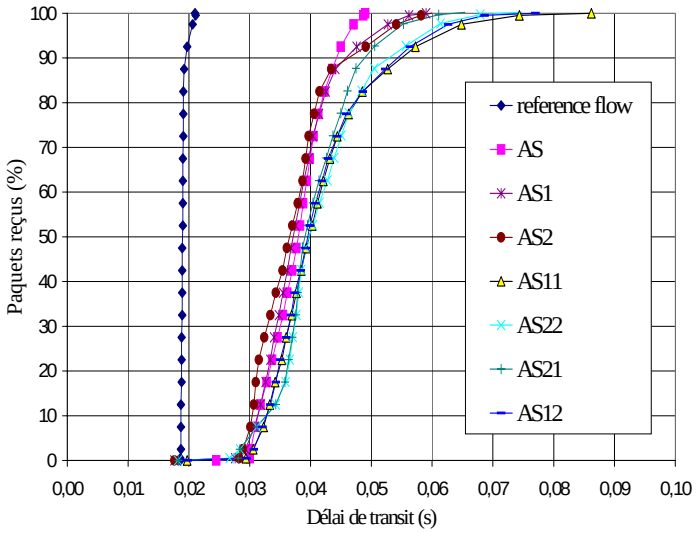


Figure 24 : Résultats du scénario 4 (1/2)

On peut cependant remarquer que 10 % des paquets (pour les flux AS1, 2, 11, 12, 21, 22) ont un délai de transit nettement plus important que celui observé pour le flux AS seul. Une première explication a été d'associer ce résultat à l'asynchronisme du système d'exploitation des PC (Free BSD) ; ce phénomène ne se répétant pas pour les mesures en GS (voir résultats du scénario 5), cette explication n'a finalement pas été retenue. A l'heure actuelle, aucune explication plausible n'a été formulée.

Remarquons enfin que le taux de perte demeure nul.

Délai (ms)	AS	AS1	AS2	AS11	AS12	AS21	AS22
- min	25	18	18	20	18	18	20
- moyen	38	38	37	42	42	40	41
- max	49	59	63	86	75	65	77
% de perte	0	0	0	0	0	0	0

Tableau 16 : Résultats du scénario 4 (2/2)

Scénario 5 : impact des flux GS sur la QdS GS

De même que pour le scénario précédent, l'impact du nombre de flux GS sur la QdS GS est faible. En effet, le Tableau 17 indique une variation inférieure à 8 ms sur la valeur moyenne du délai de transit. Ce résultat est conforté par la Figure 25 pour 100 % des paquets.

Remarquons là encore que le taux de perte est inchangé (nul).

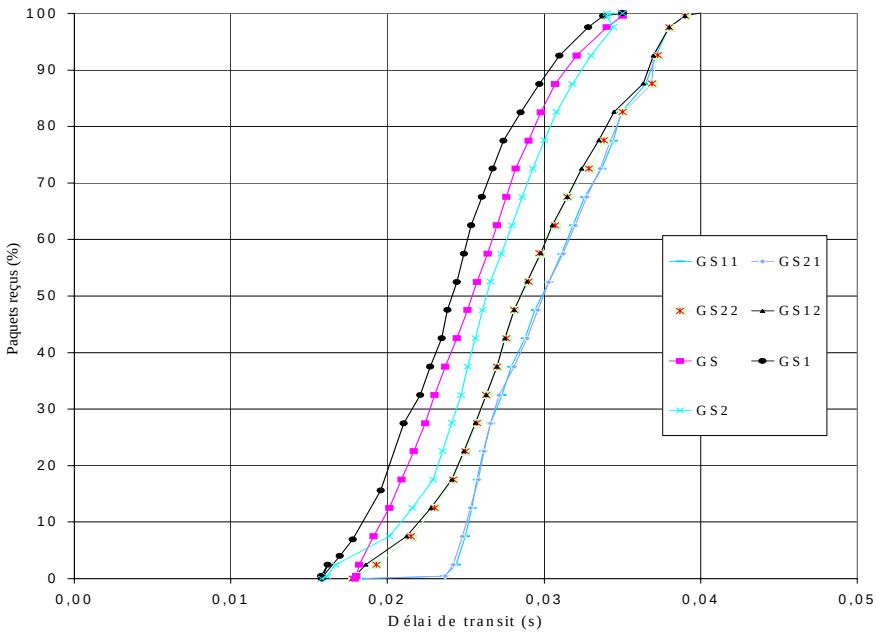


Figure 25 : Résultats du scénario 5 (1/2)

Délai (ms)	GS	GS1	GS2	GS11	GS12	GS21	GS22
- min	19	16	16	18	18	18	18
- moyen	25	26	26	32	31	33	31
- max	33	33	35	33	34	37	38
% de perte	0	0	0	0	0	0	0

Tableau 17 : Résultats du scénario 5 (2/2)

Notons que la différence maximale sur le délai de transit (20 ms) est acceptable et explicable. En effet, en conservant à l'esprit que :

- le délai de transit GS est le plus petit possible avec une gigue éventuelle correspondant à la présence d'un paquet en file d'attente,
- l'émission d'un paquet BE ne peut pas être interrompue,
- tous les paquets ont une taille de 1 Koctets et sont émis avec un débit maximal de 100 Koctets/s,

il en résulte un écart de 20 ms entre le meilleur et le pire des cas.

Scénario 6 : impact des flux AS (GS) sur la QdS GS (AS)

Le scénario 6 vise à évaluer l'impact du nombre et de la charge des flux AS (respectivement GS) sur la QdS GS (respectivement AS).

Cet impact est presque nul. En effet, le Tableau 18 indique une variation inférieure à 6 ms pour AS et 2 ms pour GS sur la valeur moyenne du délai de transit. Ce résultat est conforté par la Figure 26 pour 100 % des paquets.

Là encore, le taux de perte est inchangé (nul).

Notons que le délai de transit est quasiment le même que celui observé pour les flux AS et GS seuls (Tableaux 17 et 18 et Figures 25 et 26).

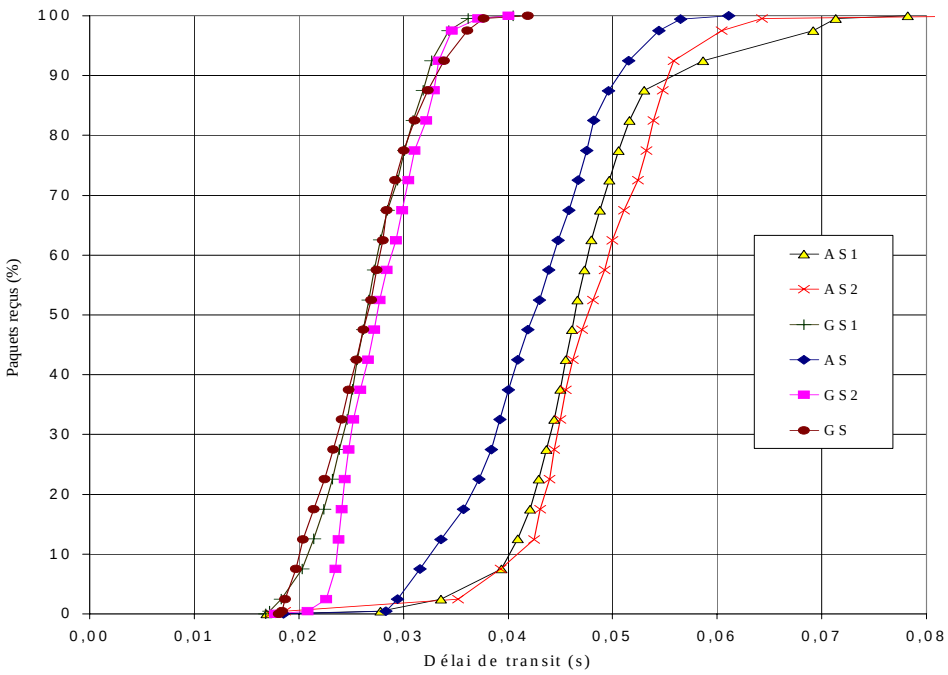


Figure 26 : Résultats du scénario 6 (1/2)

Délai (ms)	AS	GS	AS1	AS2	GS1	GS2
- min	19	18	17	17	17	18
- moyen	42	26	47	48	27	28
- max	61	42	78	88	41	40
% de perte	0	0	0	0	0	0

Tableau 18 : Résultats du scénario 6 (2/2)

4.4 Conclusion sur les mesures de performance

Une des hypothèses fortes faite lors de la définition de l'architecture est que le système de fourniture de QdS est capable d'offrir une QdS par flux (c'est-à-dire par canal de bout en bout). Les mesures réalisées permettent de valider cette hypothèse, dans les conditions expérimentales du projet.

En effet, la QdS offerte aux flux AS et GS est peu sensible à la charge du réseau et quasiment insensible au nombre de flux émis. Ce constat permet d'affirmer que, sous l'hypothèse qu'il y ait un contrôle d'admission et un *policing* par flux, un système de fourniture de QdS de type DiffServ permet de fournir une QdS reproductible pour chacun de ces flux.

Analysons plus en détail cette conclusion.

La première série de mesures (visant à évaluer la protection d'un agrégat AS et GS) permet d'établir deux conclusions :

- une architecture à services différenciés au niveau IP peut être facilement déployée dans un environnement DiffServ tel que celui de @IRS ;
- en présence de trafic BE dont la charge croît jusqu'à saturer le réseau, la QdS d'un unique flux UDP servi en AS (resp. GS) est conforme à celle attendue.

La deuxième série de mesures (visant à évaluer la protection des flux AS et GS lorsqu'il y en a plusieurs dans le réseau) permet d'apporter les conclusions supplémentaires suivantes.

Pour des flux (AS ou GS) respectant leur profil de trafic :

- l'impact du nombre de flux GS sur les QdS AS ou GS est faible ;
- l'impact du nombre de flux AS sur la QdS GS est faible ;
- l'impact du nombre de flux AS sur la QdS AS est faible mais :
  - si ce résultat est vrai pour 90% des paquets, 10% subissent un délai nettement accru. Ce résultat est cependant acceptable au regard de la spécification du service AS ;
  - en outre, il est particulièrement important pour la caractérisation d'un service de type AS dans une plate-forme DiffServ telle que celle du projet @IRS. En effet, un impact trop important aurait rendu impossible ou très difficile une telle caractérisation ;
  - notons que l'impact des paquets AS « OUT » sur le délai de transit des paquets « IN » n'est pas présenté dans ce mémoire mais une étude a été menée au LIP6, voir (Lochin, 2004) pour évaluer un tel impact.

5 Conclusion

Ce chapitre a tout d'abord présenté les deux modèles d'environnement de l'Internet considérés pour nos travaux, successivement mono puis multi domaines. Dans le deuxième paragraphe, nous avons présenté l'architecture de communication dans laquelle s'insère notre contribution. Le troisième paragraphe a décrit l'API mise à disposition des utilisateurs en présentant les paramètres et les primitives de service ainsi que l'enchaînement de ces dernières. Enfin, nous avons exposé les expérimentations menées pour valider les choix de conception de l'architecture de communication.

Ce chapitre permet de faire ressortir trois besoins forts :

- le premier concerne la gestion des services Transport et IP. Pour satisfaire une requête de QdS formulée pour un canal donné, le système de communication doit faire le choix d'un protocole de Transport adéquat (et le configurer si besoin pour le protocole FPTP) ainsi que d'une classe de service DiffServ pour le marquage des paquets IP. Pour pouvoir effectuer ce choix, nous proposons dans le Chapitre 3 un modèle des performances des services IP en environnement mono puis multi domaines. Ce modèle est alors intégré dans un mécanisme de gestion automatique de la QdS mettant en adéquation les requêtes des utilisateurs et les performances des services ;
- le deuxième besoin concerne la partie signalisation. En effet, comme nous l'avons exposé dans ce chapitre, dans le cas le plus général les communications s'étendent sur différents domaines administrés de façon indépendante. L'hypothèse retenue quant au dimensionnement des routeurs de bordure des différents domaines conduit au besoin d'une architecture de signalisation intégrée au système, de manière à pouvoir garantir une certaine QdS par canal de communication. Le Chapitre 4 présentera notre proposition d'architecture de signalisation multi domaines ;

– le dernier besoin concerne la mise en œuvre de mécanismes facilitant l'utilisation de notre système, sous deux angles distincts : (1) faciliter la sélection et la configuration de la QdS, en proposant des paramètres plus adaptés à des familles d'application et (2) faciliter l'intégration de notre API dans des applications déjà existantes. Dans le Chapitre 5, nous exposerons les principes d'un module d'adaptation accessible par une interface graphique qui permet à l'utilisateur de configurer simplement ses canaux. Nous présenterons également les mécanismes conçus pour permettre aux applications déjà existantes d'accéder à notre API avec un minimum de modification de leur code.

Chapitre 3 Caractérisation de la QdS des services de bout en bout et mécanismes de sélection automatique

Introduction

Ce chapitre a deux objectifs majeurs. Le premier est de présenter les modèles de caractérisation de la QdS (exprimée en termes de délai et de fiabilité) des services de bout en bout (couplant niveaux IP et Transport) offerts par notre architecture de communication dans un environnement mono ou multi domaines. Le deuxième objectif est de présenter l'algorithme, qui, sur la base des modèles précédents, sélectionne automatiquement les services de niveaux IP et Transport permettant de satisfaire la QdS requise par une application pour un canal de communication, dans un environnement mono ou multi domaines.

Pour cela, le chapitre est structuré de la façon suivante.

Le paragraphe (1) présente les travaux relatifs à la caractérisation des services IP dans un environnement mono domaine. Dans une première partie (1.1), nous présentons les résultats d'une campagne de simulation destinée à conforter les mesures présentées au Chapitre 2. Dans une deuxième partie (1.2), nous présentons le modèle analytique retenu pour caractériser les services IP.

Le paragraphe (2) présente les travaux relatifs à la caractérisation des services IP dans un environnement multi domaines. Dans une première partie (2.1), nous présentons une étude mathématique conduisant à l'élaboration d'un modèle générique. Dans une deuxième partie (2.2), nous appliquons les résultats de cette étude au modèle analytique retenu en (1.2). Dans une troisième partie (2.3), nous validons l'étude précédente par une campagne de mesures menée dans un environnement réseau émulant (via le logiciel *Dummynet*) le comportement d'un environnement multi domaines.

Le paragraphe (3) présente alors les travaux relatifs à la caractérisation de la QdS des services de bout en bout couplant niveaux IP et Transport offerts par notre architecture de communication dans un environnement multi domaines.

Enfin, le paragraphe (4) présente l'algorithme de sélection automatique des services de niveaux IP et Transport permettant de satisfaire la QdS requise par une application pour un canal de communication, dans un environnement multi domaines.

1 Caractérisation des services IP en environnement mono domaine

Ce paragraphe présente les travaux relatifs à la caractérisation des services IP dans un environnement mono domaine. Dans une première partie (1.1), nous présentons les résultats d'une campagne de simulation destinée à conforter les mesures présentées au Chapitre 2. Dans une deuxième partie (1.2), nous présentons le modèle analytique retenu pour caractériser les services IP.

1.1 Validation en simulation des mesures @IRS

1.1.1 Introduction

L'objectif de la campagne de simulations présentée ci-après est de conforter le résultat des mesures effectuées sur la plate-forme @IRS, en étendant les scénarios de mesure décrits au chapitre 2. Lors des tests effectués sur la plate-forme @IRS (représentée schématiquement sur le Figure 27), les équipements étaient limités à deux émetteurs (noté E1 et E2) et un récepteur (noté R), les émetteurs ne représentant chacun qu'au maximum trois sources de trafic. Les paquets véhiculés par les différents flux ne traversaient donc qu'un nombre limité d'équipements intermédiaires, en l'occurrence deux routeurs de bordure (notés Rb) et deux routeurs de cœur (notés Rc), uniquement chargés par le trafic issu de E1 et E2.

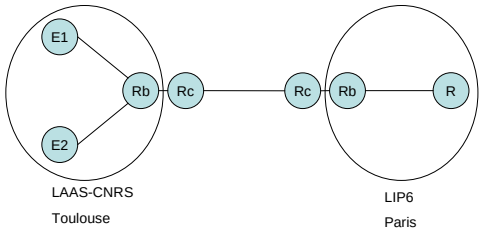


Figure 27 : Topologie de la plate-forme @IRS

Appelons « site local », le regroupement d'un routeur de bordure et de l'ensemble des émetteurs (ou des récepteurs) qui lui sont directement attachés, voir Figure 28.

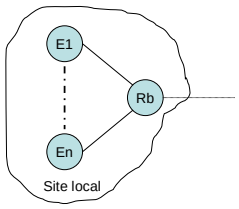


Figure 28 : « Site local »

Dans le même esprit que les tests menés sur le @IRSBone, l'objectif de cette campagne n'est pas de trouver le « meilleur » paramétrage des plates-formes (taille des files d'attente, valeurs des

poids du WFQ, capacités des liens, …), mais de conforter les résultats issus des expérimentations dans une configuration étendue de test en augmentant :

– le nombre d'émetteurs par site local, voir Figure 29 ;

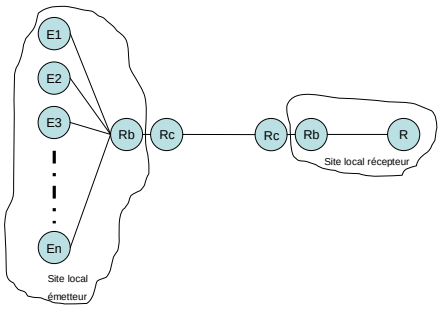


Figure 29 : Augmentation du nombre de sources de trafic dans un site local

– le nombre de sites locaux convergeant sur un routeur de cœur, voir Figure 30 ;

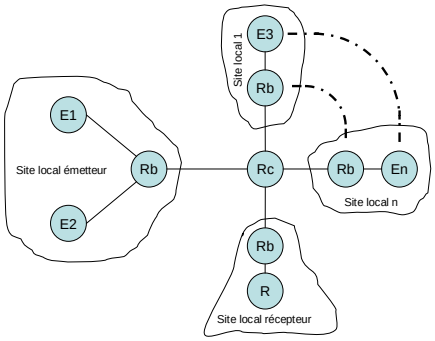


Figure 30 : Augmentation du nombre de sites locaux sur un routeur de cœur

– le nombre de routeurs de cœur, chacun étant chargé par un site local, voir Figure 31.

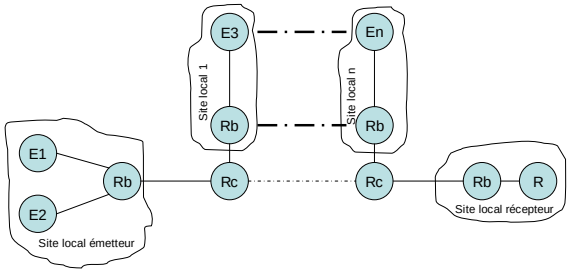


Figure 31 : Augmentation du nombre de routeurs de cœur

1.1.2 Modélisation de l'environnement et paramètres des simulations

1.1.2.1 Modélisation de l'environnement

Des simplifications de l'environnement réel ont été réalisées. Elles concernent les fonctionnalités des interfaces d'entrée des routeurs de bordure et les fonctionnalités des interfaces de sortie de tous les routeurs (bordure et cœur).

Interfaces d'entrée des routeurs de bordure

Dans nos simulations, il n'y pas de différence entre les routeurs de bordure et les routeurs de cœur car les fonctionnalités implémentées dans les équipements se résument aux mécanismes d'ordonnancement. Il n'y a pas de mécanismes liés au contrôle d'admission ou à la mise en forme du trafic de type *Token Bucket*, …. Ce choix a été fait car l'objectif des simulations est simplement d'évaluer les interactions pouvant apparaître lorsque plusieurs flux à QdS sont confrontés en prenant pour hypothèse que chaque site local respecte le profil du trafic lié à la QdS considérée. Dans la suite de ce paragraphe, lorsque la dénomination routeur de cœur ou routeur de bordure apparaît, elle fait seulement appel à une différence de position dans la topologie et non à une différence de fonctionnalité.

Interfaces de sortie des routeurs

Seules les fonctionnalités des interfaces de sortie des routeurs sont conservées, à savoir : le classificateur et les mécanismes d'ordonnancement et de gestion de files d'attente (voir Figure 32).

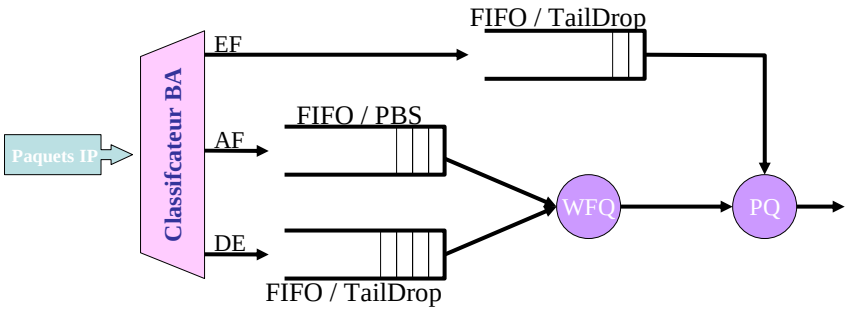


Figure 32 : Configuration des interfaces de sorties des routeurs @IRS

La caractérisation du service GS ne représentant pas de difficulté (voir les conclusions du paragraphe (4.4) du Chapitre 2) et les flux GS n'ayant que peu d'impact sur les performances du service AS, seuls les mécanismes liés à l'ordonnanceur WFQ ont été conservés. La classification se fait uniquement sur l'identifiant de flux (noté Flux_Id).

Le mécanisme de politique de gestion de file d'attente PBS (*Partial Buffer Sharing*) n'intervient que lorsqu'il y a des congestions dans le routeur (en fait, les paquets AS hors profil seront marqués en bordure OUT et seront prioritairement détruits en cas de congestion). Les campagnes de simulations se sont déroulées sous l'hypothèse qu'un contrôle d'admission veillait à ce qu'aucune congestion n'intervienne. Cette fonctionnalité n'a donc pas été testée en simulation.

Le module ns-2 lié à l'ordonnanceur est représenté par la Figure 33 :

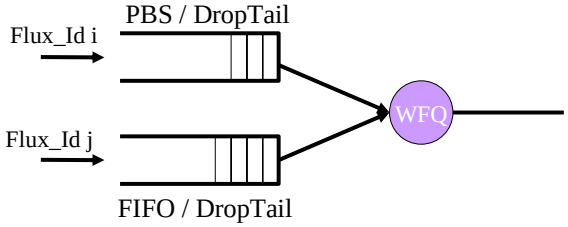


Figure 33 : Ordonnanceur simplifié implémenté en ns-2

Note sur les modules supplémentaires implémentés dans la distribution standard de ns-2

La distribution standard de ns-2 n'offre pas les fonctionnalités d'un ordonnanceur de type WFQ ou, si elles sont disponibles, ne sont pas accessibles simplement. En effet, certaines implémentations[23] couplent les mécanismes d'ordonnancement des flux avec le *policing* qui est associé à chacun des flux (rejoignant les traitements communément trouvés dans les routeurs de bordure de type DiffServ). Un court récapitulatif de ce type d'ordonnanceur (WFQ, SFQ, WFQ+, WF2Q) peut être trouvé sur (Semaria, 2001). Comme il a été écrit précédemment, l'objectif des simulations est de confirmer les résultats liés aux performances des paquets de la classe du service AS déduits des campagnes expérimentales, les fonctions de *policing* ne représentant dès lors qu'une complexité supplémentaire et un risque de perturber les mesures de performances ainsi que de rendre plus difficile l'interprétation des résultats. Les modules ajoutés concernent donc essentiellement l'ordonnanceur WFQ et le gestionnaire de file d'attente PBS, même si quelques simulations basiques à l'aide d'un PQ ont permis de retrouver les résultats expérimentaux obtenus avec les paquets de la classe du service GS. Deux modules d'origines différentes ont été successivement intégrés dans le simulateur ns-2 pour comparer les résultats obtenus Nous détaillons successivement ces deux versions.

Mécanismes de WFQ issus du LIP6

Ce premier module est issu des travaux réalisés dans le cadre du LIP6 (LIP6, 2002). Il propose une implémentation simplifiée des mécanismes du WFQ en permettant de s'affranchir du *policing*. Le module contient 3 fichiers :

- n-wfq.tcl ;
- wfq.h ;
- wfq.cc.

Le premier fichier contient l'interface entre l'utilisateur et le cœur du simulateur. Les fichiers suivants permettent d'implémenter dans le cœur du simulateur le comportement lié à l'ordonnanceur. Le détail d'ajout de ce module n'est pas donné dans ce mémoire ; pour tout complément d'information

[23] En fait, la distribution standard de ns-2 propose une implémentation particulière du WFQ, le WF2Q. Le détail de ce mécanisme peut être trouvé dans les travaux liés aux modules DiffServ NS de Nortel.

se référer à (Auriol, 2003). L'utilisation de ce module est relativement simple ; il permet de créer autant de files d'attente sur un lien que de comportements souhaités. Un exemple est donné Figure 34 :

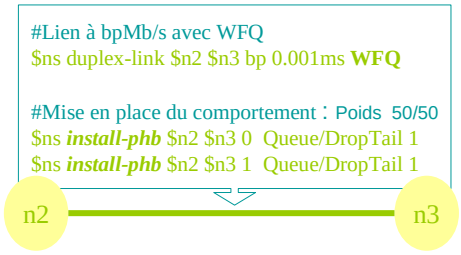


Figure 34 : Exemple d'utilisation du module WFQ du lip6

Sur cette figure, nous pouvons voir la déclaration de l'ordonnanceur *WFQ* sur le lien entre les nœuds *n2* et *n3* ainsi que la mise en place de deux files distinctes liées aux numéros de flux *0* et *1* gérées individuellement par un mécanisme de *DropTail* (implémentation du *Tail Drop*) et ayant chacune un poids égal à *1*.

Nous avons aussi intégré un module permettant de reprendre les fonctionnalités du PBS mais comme il a été annoncé au début de ce paragraphe, l'objectif étant de faire une évaluation des performances du service AS, les sources de trafic ont donc été dimensionnées pour qu'aucun des paquets ne dépasse le profil de trafic admissible sur les routeurs de bordure. Il n'y a alors aucun paquet marqué OUT. Nous n'avons donc pas effectué de tests utilisant le module PBS.

Autres mécanismes de WFQ

Une deuxième version de WFQ issue des travaux de (ULG, 2002) a été intégrée afin de vérifier les résultats obtenus avec la première. Tout comme la version du LIP6, cette version permet une utilisation simplifiée et découplée de tout autre mécanisme (comme le *policing*). Les détails d'utilisation de ce module peuvent être trouvés sur (ULG, 2002) et les fichiers sur (POLITO, 2002). Un exemple est donné Figure 35 :

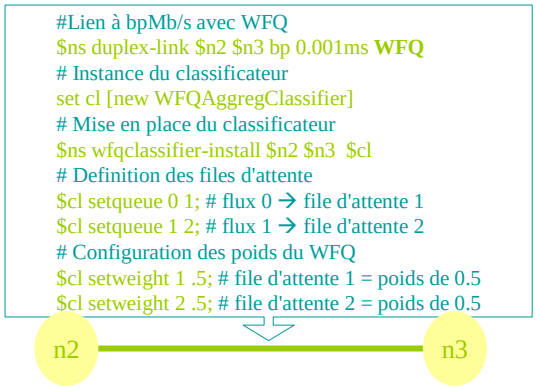


Figure 35 : Exemple d'utilisation de la deuxième version de WFQ

Sur cette figure, nous pouvons voir la déclaration de l'ordonnanceur *WFQ* sur le lien entre les nœuds *n2* et *n3*. Une étape intermédiaire avant d'avoir accès à la déclaration des files d'attente consiste à déclarer un objet *WFQAggregClassifier* qui permet d'appliquer un traitement différent suivant le numéro de flux. La configuration des files d'attente se fait ensuite en deux étapes : déclaration des files et configuration des poids de WFQ de chaque file. On notera que, n'ayant pas accès au type des files, celles-ci seront par défaut de type *DropTail*.

Dans la suite de ce paragraphe, seuls les résultats obtenus avec cette version d'implémentation sont présentés.

1.1.2.2 Paramètres des simulations

Les paramètres intervenant lors des simulations ns-2 sont :

- la taille des paquets,
- le modèle de source de trafic,
- la durée de la simulation,
- le dimensionnement des liens,
- le rapport entre le temps de propagation sur le lien et le temps d'émission sur le lien.

Nous détaillons ces différents paramètres dans les paragraphes suivants.

Taille des paquets

Lors des expérimentations sur la plate-forme @IRS, la taille des paquets était constante tout au long des tests et égale à 1024 Octets. Pour les tests en simulation, cette taille de paquet, notée p, a été conservée.

Sources de trafic

Lors des expérimentations sur la plate-forme @IRS, les modèles de source de trafic étaient de type CBR (Constant Bit Rate). Le logiciel *Débit6* (présenté dans le Chapitre 24.1.2) permettait d'émettre des paquets de taille donnée à intervalle de temps régulier.

Sous ns-2, le module CBR permet de générer un flux régulier de paquets. Le principal paramètre de cet agent est le temps inter paquets. Soit t le temps inter paquets en seconde, p la taille d'un paquet en octet, nous avons donc un débit d défini par l'Équation 1 :

$$d = \frac{p \times 8}{t} \text{ en bits/seconde}$$

Équation 1 : Calcul du débit

La débit des sources se règle en modifiant le temps inter paquets, et non en intervenant sur la taille des paquets (ce qui est plus difficile en simulation).

Dimensionnement des liens

Lors des expérimentations sur la plate-forme @IRS, le VPN fourni par Renater autorisait un débit maximum de 816Kb/s en cœur de réseau, et les sites locaux étaient équipés d'Ethernet 100Mb/s. Nous verrons ultérieurement que le dimensionnement retenu des liens (caractérisés par la BPL) varie en fonction de la topologie ciblée.

Durée des simulations

Lors des expérimentations sur la plate-forme @IRS, la durée des tests était fixée arbitrairement à 300 secondes. Dans un premier temps, cette durée a été conservée pour les simulations puis a été ramenée à 30 secondes et enfin à 10 secondes suivant les topologies pour diminuer le temps de simulation.

Nous considérons que nous pouvons diminuer le temps de simulation par la justification suivante. La durée de simulation est liée à la durée de remplissage des files d'attente, ce qui correspond à la phase de transition. Il est nécessaire que la durée de simulation soit supérieure à la durée de la phase de transition. Ne considérant que des sources dont le comportement est constant, dès que les files d'attentes sont pleines, le régime permanent est établi. Cette durée minimale de simulation se calcule de la façon suivante. Soit :

- une source de trafic de type CBR caractérisée par un temps inter paquet t,
- une file d'attente caractérisée par une taille de *buffer* de b paquets de taille p octets,
- un lien caractérisé par sa BPL de z bits/s,

Le temps de saturation, noté t_s, de la file d'attente est défini par l'Équation 2 :

$$t_s = \frac{b \times p \times 8}{\frac{p \times 8}{t} - z} \text{ en seconde}$$

Équation 2 : Temps de saturation

D'où la durée minimale, notée d_m, de la simulation est définie par l'Équation 3 :

$$d_m >> t_s$$

Équation 3 : Durée minimale de la simulation

Dans le cas des expérimentations réalisées sur la plate-forme @IRS, les données étaient :

- b = 20 paquets de taille p = 1024 Octets,
- t = 0.010 seconde pour le cas où les paquets BE saturent le lien à z = 816000 bits/seconde,

soit t_s = 0.4 seconde ce qui vérifiait bien $t << d_m$, lors des expériences d'une durée de 300 secondes.

Rapport temps de propagation / temps d'émission

Une grandeur particulièrement importante pour nos mesures ns-2 est le rapport entre le temps de propagation[24] des paquets sur les liens (noté t_p) et le temps de service d'un paquet sur le lien, appelé

[24] Egalement appelé temps « cuivre ».

aussi temps d'émission (noté t_e). Le temps d'émission est fonction de la capacité (réservée au flux considéré) du lien sur lequel va être émis le paquet et de la taille des paquets. D'un autre côté, le temps de propagation sur le lien est facilement paramétrable grâce au module *duplex-link* fourni dans la distribution standard de ns-2 (voir Figure 34). En effet un argument de la fonction qui permet d'installer un lien entre deux nœuds est le délai nominal de propagation sur le médium. Le rapport entre les deux temps, noté *r*, est alors défini par l'Équation 4 :

$$r = \frac{t_p}{t_e} \; avec \; t_e = \frac{p}{d}$$

Équation 4 : Rapport temps de propagation / temps d'émission

Ce rapport permet d'évaluer le phénomène prépondérant dans la mesure du délai de bout en bout. Ainsi, si $r > 1$, alors le temps de propagation aura une grande influence sur le délai de bout en bout alors que le temps d'émission aura une influence réduite. Or dans un environnement réel, les fluctuations dans la distribution du délai de bout en bout des paquets sont essentiellement dues au temps passé dans les routeurs ; en effet très peu de variations sur le délai peuvent intervenir lorsque les paquets transitent sur le lien. Il faut donc s'attacher dans les simulations à ce que $r << 1$ pour reproduire un comportement correct.

Pour avoir un ordre de grandeur de ce rapport, considérons l'exemple suivant.

Dans un environnement réel, le temps de propagation, noté t_p, est fonction de la longueur du médium, notée *l*, et de la vitesse de propagation sur le médium, notée *c* (en km/seconde).

Il est défini par l'Équation 5 :

$$t_p = \frac{l}{c} \text{ en seconde}$$

Équation 5 : Temps de propagation

Dans les expérimentations effectuées sur la plate-forme @IRS, nous pouvons estimer ce rapport. En effet, en estimant :

- la distance entre Toulouse et Paris à 800 km,
- une vitesse sur le médium de 200000 km/s,

on en déduit :

t_p = 800/2000000 = 0.004 seconde

De même, on peut estimer t_e à :

$\frac{p}{d}$ = 1024*8/850000 = 0.009 seconde

Ce qui conduit à un rapport :

r = 0.444

L'effet prépondérant est donc le temps passé dans les routeurs et non le temps de propagation sur le lien.

1.1.3 Scénarios de simulation

Dans toutes les configurations décrites dans ce paragraphe, les tests ont été réalisés avec un ordonnanceur WFQ dont les poids sont de 0.5 et 0.5.

Les scénarios sont basés sur le même principe que ceux des campagnes expérimentales : un flux AS sert de flux de référence ; un (ou plusieurs) flux BE sature(nt) les liens afin de congestionner le réseau et d'autres flux AS sont ajoutés, mais toujours en accord avec le maximum autorisé par le contrôle d'admission.

Les flux AS sont distingués par un identifiant de flux égal à 1 alors que les flux BE ont un identifiant égal à 0. La durée des simulations sera donnée lors de la description des topologies. Les files d'attente liées aux paquets de la classe BE ont (sauf indication particulière) une profondeur de 20 paquets ; celles liées aux paquets de la classe AS ont une profondeur de 2 paquets.

1.1.3.1 Scénario 1 : Augmentation du nombre de sources par site local

Description de la topologie

L'objectif de ces mesures est d'étendre le nombre de flux concurrents arrivant sur un même routeur de bordure. La topologie retenue est représentée Figure 36.

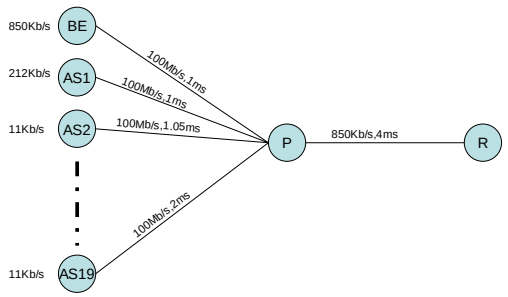


Figure 36 : Augmentation du nombre de sources d'un site local

Les brins entre les 20 sources (notées BE, AS1 à AS19) et l'équipement représentant le routeur de bordure (noté P) sont des liens à 100 Megabits/s ayant un temps de propagation (t_p) compris entre 1 ms et 2 ms. Cette configuration correspond à celle que nous pouvons retrouver sur un site local avec des sources de trafic de débit assez faible et des liaisons de type Ethernet 100 Megabits/s (les temps de propagation sont relativement élevés pour des connexions de type LAN mais sont en accord avec la relation de l'équation 4, $r = 0.1$).

Le brin entre le nœud P et le récepteur (noté R) est un lien à 850000 bits/s ayant un temps de propagation de 4 ms (voir l'équation 5) ce qui correspond à celle de la plate-forme @IRS.

La durée des simulations est la même que celle des tests expérimentaux, soit 300 secondes.

Les 20 sources émettent le trafic suivant :

- 1 source génère un trafic BE : sa charge est telle que le lien entre le nœud P et le récepteur est congestionné ;
- 19 sources génèrent un trafic (noté flux AS1 à flux AS19) et se partagent la bande passante attribuée au trafic à QdS ; le lancement de ces sources est différé et s'étale sur une durée de 10 secondes pour éviter tout risque de synchronisation entre flux.

Résultats et interprétation

Les résultats suivants ont été obtenus, voir Figure 37.

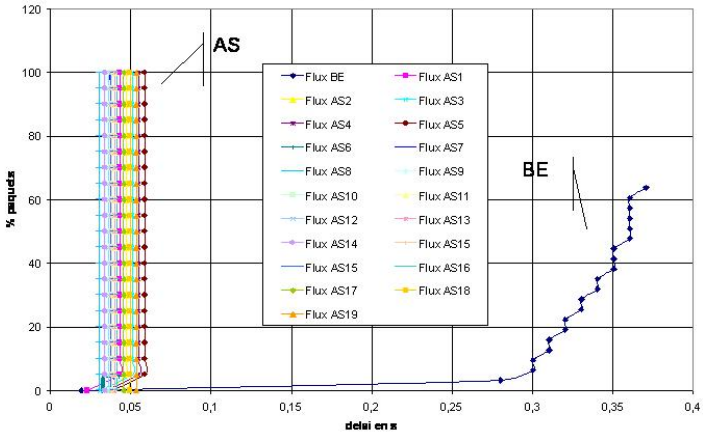


Figure 37 : Résultats : augmentation du nombre des sources

Sur cette figure, nous remarquons que :

- les performances des flux AS sont nettement meilleures que celles des flux BE, tant au niveau des pertes (il n'y a pas de perte de paquets AS) qu'au niveau délai ;
- les performances des flux AS sont similaires.

Nous retrouvons donc les résultats obtenus sur la plate-forme @IRS dans une topologie simplifiée. Pour une charge totale donnée, nous pouvons conclure que le nombre de sources émettant du trafic AS n'a pas d'influence sur les performances des flux.

Note : le décalage dans le temps des courbes représentatives des paquets AS provient de la configuration des brins d'accès. En effet, dans un premier temps, des simulations ont été faites avec des brins d'accès identiques, ce qui provoquait une synchronisation des arrivées de paquets sur le nœud P ; or, ce synchronisme ne correspond pas à ce que nous pourrions observer dans des conditions réelles. Toutefois, ces premières simulations ont montré l'importance du dimensionnement des files d'attente ; en effet :

- si la file d'attente des paquets AS est de profondeur égale au nombre de flux concurrents, alors il n'y a pas de perte au niveau du routeur mais les paquets subissent des variations très importantes dans leur délai de bout en bout ;

– si la file d'attente des paquets AS est d'une profondeur réduite, alors on observe un grand nombre de pertes concentrées uniquement sur certains flux (ce qui présente une limite des simulations : en effet, dans la réalité, la probabilité est faible que ce soit toujours les mêmes flux qui soient dégradés).

1.1.3.2 Scénario 2 : Convergence du trafic sur un routeur de cœur

Description de la topologie

L'objectif de cette campagne est de déceler l'impact que pourrait avoir une concentration de flux représentant deux différents agrégats (BE et AS) sur un routeur de cœur. Cette configuration permet de simuler un routeur de cœur possédant plusieurs interfaces. Dans un environnement réel, ce nombre est en fait limité pour des raisons matérielles.

La topologie incluant le dimensionnement des différents liens est représentée sur la Figure 38.

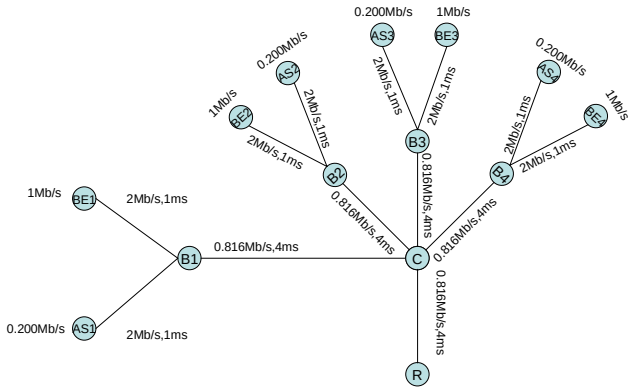


Figure 38 : Topologie de la 2ème campagne de simulation

Le flux de référence (noté AS1) traverse un routeur de cœur (noté C), celui-ci étant progressivement chargé selon les scénarios décrits dans le Tableau 19 :

Scénario 1	AS1
Scénario 2	AS1 + BE1
Scénario 3	AS1 + BE1 + BE2
Scénario 4	AS1 + BE1 + BE2 + BE3 + BE4
Scénario 5	AS1 + AS2
Scénario 6	AS1 + AS2 + BE1
Scénario 7	AS1 + AS2 + BE1 + BE2
Scénario 8	AS1 + AS2 + AS3 + AS4 + BE1 + BE2 + BE3 + BE4

Tableau 19: Scénarios de la 2ème campagne de simulation

Les sources actives démarrent simultanément pour une durée de simulation réduite à 30 secondes (pour éviter de générer des fichiers de traces trop volumineux), ce qui situe la durée des tests très au-dessus du temps minimal de simulation donné par l'Équation 2 ; en d'autres termes, les files d'attente BE sont très rapidement saturées.

Résultats et interprétation

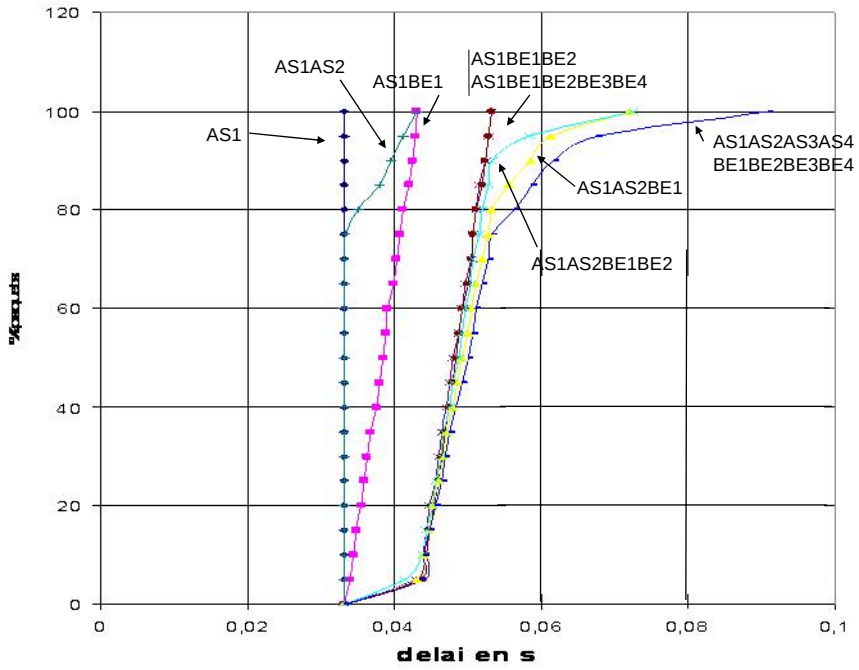


Figure 39 : Résultats : convergence des flux sur un routeur de cœur

Dans cette configuration, le trafic supplémentaire a un impact sur les performances du flux de référence dés le scénario 2. Ceci peut s'expliquer par la mise en concurrence simultanée (voir l'impact du synchronisme relevé sur la 1ère topologie) des paquets sur un nœud. Cependant, la comparaison des résultats obtenus avec les scénarios 4, 6, 7, 8 montre que l'impact peut se majorer ; en effet, les dégradations des performances sont quasiment identiques pour 80 % des paquets qu'il y ait un site supplémentaire (scénario 4) ou trois sites supplémentaires (scénario 8). La dernière partie étirée de la courbe représentative du scénario 8 s'explique si on regarde les schémas de la Figure 40 et de la Figure 41 représentant le cas le plus défavorable de la durée séparant 2 paquets consécutifs appartenant au flux de référence AS1.

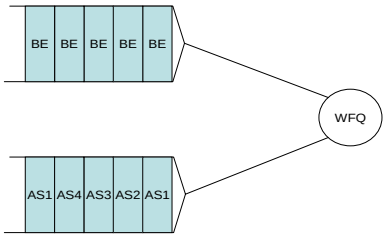


Figure 40 : État des files d'attente du routeur de coeur

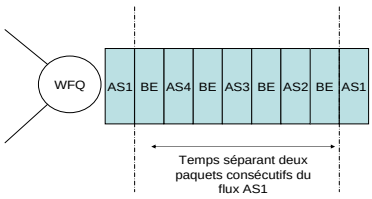


Figure 41 : Disposition des paquets en sortie du routeur de coeur

D'après les poids du WFQ (50/50), un paquet BE s'intercale entre deux paquets AS (quel que soit le flux d'appartenance du paquet AS) ; dans le pire cas, la différence de délai entre un paquet AS1 ne rencontrant aucun autre paquet (cas du scénario 1) et un paquet AS1 devant attendre que la file

d'attente AS se vide totalement, est de 7 × t_e (voir Figure 41), t_e désignant le temps d'émission donné dans l'Équation 4. On s'aperçoit que la différence observée est inférieure à ce temps, ce qui peut s'expliquer par le fait que ce cas pire ne s'est jamais produit au cours de la simulation.

1.1.3.3 Scénario 3 : Augmentation du nombre de routeurs de cœur traversés

Description de la topologie

Pour ce scénario, un soin particulier a été apporté sur le dimensionnement des liens en termes de capacité et de temps de propagation. En effet, l'objectif de cette topologie est d'étudier les performances d'un flux AS traversant un réseau de transit composé de plusieurs routeurs de cœur reliés par des liens de grande capacité, eux-mêmes chargés par du trafic issu des sources d'autres domaines.

Chaque routeur de cœur est donc chargé par un trafic agrégé important issu de routeurs de bordure (B2 à B4) comprenant des flux BE et du flux à QdS (ayant des débits élevés pour simuler l'agrégation de flux). Les liaisons entre routeurs de bordure et routeurs de cœur ont des caractéristiques, en termes de bande passante et de délai, correspondant à la simulation d'un réseau de cœur. Notre site de référence (équipements indexés « 1 ») a des caractéristiques semblables aux autres scénarios. La configuration des liens est représentée dans la Figure 42.

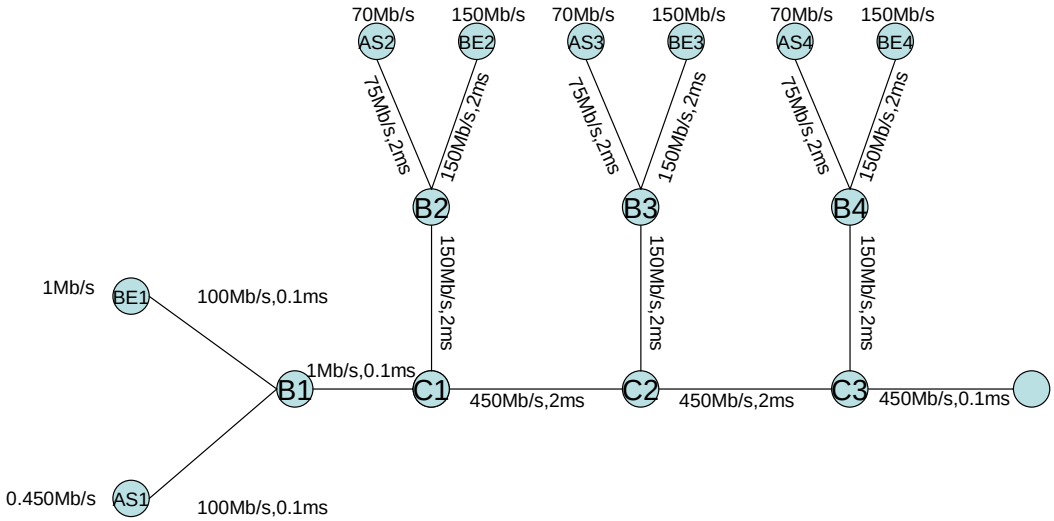


Figure 42 : Topologie de la 3ème campagne de simulation

Le dimensionnement des liens a été étudié afin de se rapprocher au plus près d'une configuration réelle. Ainsi :

- les nœuds AS1 et BE1 représentent une plate-forme locale accédant au cœur du réseau par B1 ;
- le reste de la topologie représente :
  - d'une part des sources de flux (agrégés) ayant des débits importants sortant des routeurs de bordure (B2, B3, B4) qui représentent des passerelles inter domaines), et venant s'ajouter aux flux du site local ;

- d'autre part, la concentration de ces flux sur les équipements de cœur (C1, C2, C3).

Le flux à QdS de référence (AS1) traversera l'ensemble des routeurs de cœur, ceux-ci étant progressivement chargés selon les scénarios suivants (Tableau 20).

Scénario 1	AS1
Scénario 2	AS1 + BE1
Scénario 3	AS1 + BE1 + BE2
Scénario 4	AS1 + BE1 + BE2 + BE3 + BE4
Scénario 5	AS1 + AS2
Scénario 6	AS1 + AS2 + BE1
Scénario 7	AS1 + AS2 + BE1 + BE2
Scénario 8	AS1 + AS2 + AS3 + AS4 + BE1 + BE2 + BE3 + BE4

Tableau 20: Scénarios de la 3ème campagne de simulation

Les sources actives démarrent simultanément pour une durée de simulation de 10 secondes. La durée des simulations a été diminuée pour des raisons de ressources systèmes. En effet, le trafic généré représente des traces ns-2 de résultat extrêmement volumineuses (le dernier scénario représente plus de 400 Moctets de trace). La durée de simulation du seul Scénario 8 a été tronquée après plus 36 heures de calcul sur un PC type Pentium 4 à 1.6GHz ayant 512MB de RAM. Cependant l'Équation 2 montre que ces conditions de simulation d'une durée réduite sont acceptables vis-à-vis du temps de saturation des files d'attente ; en effet :

– pour BE1 : b = 20 paquets de taille p = 1024 Octets, d = 1 Mbits/seconde et z = 0.5 Mbits/seconde d'où t_s = 0.04 seconde ;
– pour les autres flux BE dans le scénario 8 : b = 20 paquets de taille p = 1024 Octets, d = (75+75+75+0.5) Mbits/seconde et z = 225 Mbits/secondes d'où t_s = 0.16 seconde.

Résultats et interprétation

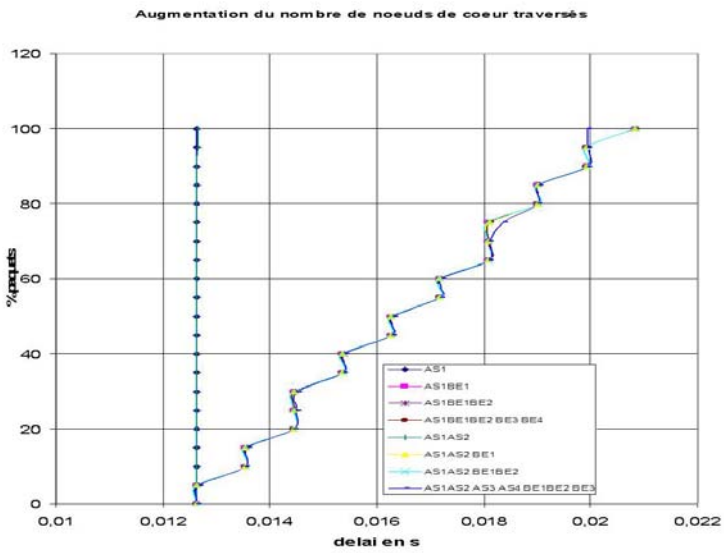


Figure 43 : Résultats : augmentation du nombre de routeurs de cœur traversés

Les courbes montrent qu'à partir du moment où un flux BE congestionne le premier routeur (ici B1), le reste de trafic n'a que peu d'influence sur le flux de référence.

La dernière partie de la courbe du scénario 8 (pour les délais de transit supérieurs à 0.02 seconde) se détache des autres courbes car la simulation a été arrêtée avant la fin du scénario.

1.1.4 Conclusion sur les simulations

Les campagnes de simulation présentées dans ce paragraphe ont étendu les scénarios retenus lors des tests expérimentaux menés sur la plate-forme @IRS, afin d'étudier une mise à l'échelle des hypothèses qui avaient alors été faites.

Le choix des ces topologies a été fait pour pouvoir représenter un grand nombre de cas. En effet, une configuration quelconque peut se décomposer suivant les trois topologies retenues. La généralisation, représentée dans la Figure 44, montre comment il est possible de décomposer un problème plus complexe en éléments dont on connaît l'impact sur un flux de référence.

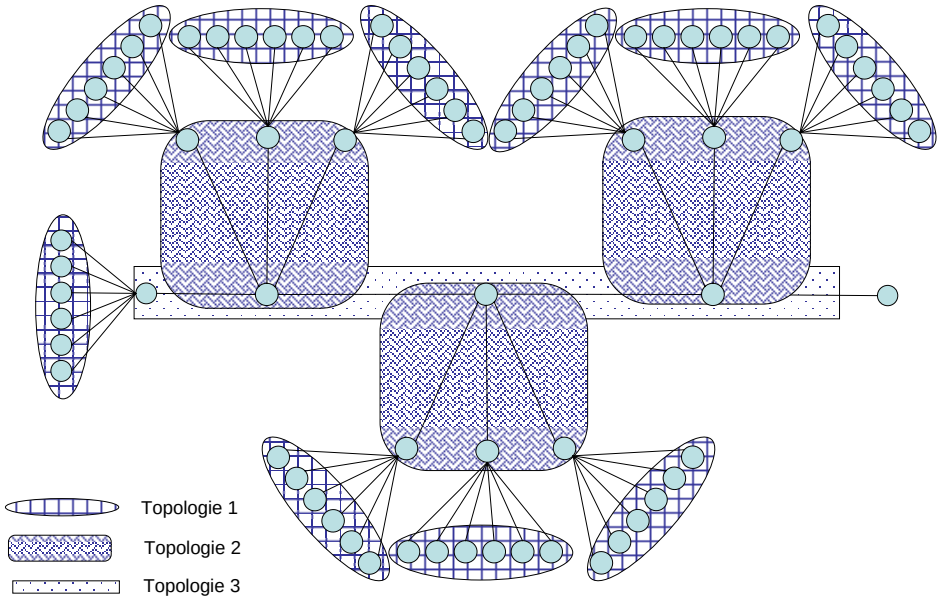


Figure 44 : Topologie générale

Dans un contexte aussi large que possible, cette campagne de simulation (précédée par la campagne expérimentale du projet @IRS) permet de poser l'hypothèse suivante :

Pour un état de charge de la route (par exemple un état de congestion) et une configuration de plate-forme donnés (poids du WFQ, taille des files d'attente,…), les performances entre deux routeurs de bordure d'un flux à QdS suivant cette route ne dépendra pas (ou très peu) de la quantité (nombre et charge) de flux de cette même classe, quelle que soit la topologie rencontrée.

La deuxième topologie a montré qu'il est possible de majorer l'influence de la convergence de flux agrégés sur un routeur de cœur.

1.2 Modèle analytique retenu

1.2.1 Étude théorique : modèle de caractérisation

Dans les paragraphes précédents, nous avons vu qu'il était possible de caractériser les services des différentes classes IP sous les conditions énoncées dans l'encadré ci-dessus. L'objet de ce paragraphe est d'exposer les différentes solutions permettant de modéliser analytiquement la fonction de répartition du délai de bout en bout qui caractérise les performances des services IP. Cette modélisation est indispensable pour limiter la quantité de données à mémoriser. A défaut, le système de communication serait obligé de conserver l'ensemble des points de la fonction de répartition pour toutes les routes, ce qui représenterait une masse de données trop importantes.

1.2.1.1 Différents modèles de caractérisation des performances des services

Pour chacun des modèles ci-dessous, nous présentons tout d'abord l'expression du modèle, les paramètres à conserver, puis le mode opératoire pour obtenir ces données et enfin comment il est possible d'utiliser le modèle.

Modèle trigonométrique

Expression

Ce modèle est basé sur la fonction hyperbolique : $x \rightarrow tanh(x)$ (choisie pour la forme générale d'une telle fonction) et est défini par la fonction suivante :

$$f(x) = a_0 + a_1.tanh(a_2.x - a_3) \text{ pour } x \in [x_{min}, x_{max}]$$

dans laquelle :

- x représente le délai de bout en bout ;
- x_{min} (resp. x_{max}) représente le plus petit (resp. grand) délai observable.

Paramètres

Les 4 coefficients a_i ainsi que x_{min} et x_{max} sont les paramètres du modèle de la fonction de répartition.

Mode opératoire

Les paramètres sont calculés, à partir des points expérimentaux, par un programme Matlab permettant de trouver la meilleure approximation de la courbe passant par les points expérimentaux en fonction des critères de convergence souhaités.

Utilisation

A partir d'un délai donné t, il est facile de connaître directement, en calculant $f(t)$, le pourcentage de paquets qui seront reçus.

Modèle affine par morceaux

Expression

Ce modèle est basé sur le découpage de la courbe initiale en N segments de droite, d'équation :

$$\forall i \in [1,N]\ y = a_i.(x-c_i)\ \ \forall x \in [x_{min} + (i-1).h,\ x_{min} + i.h]$$

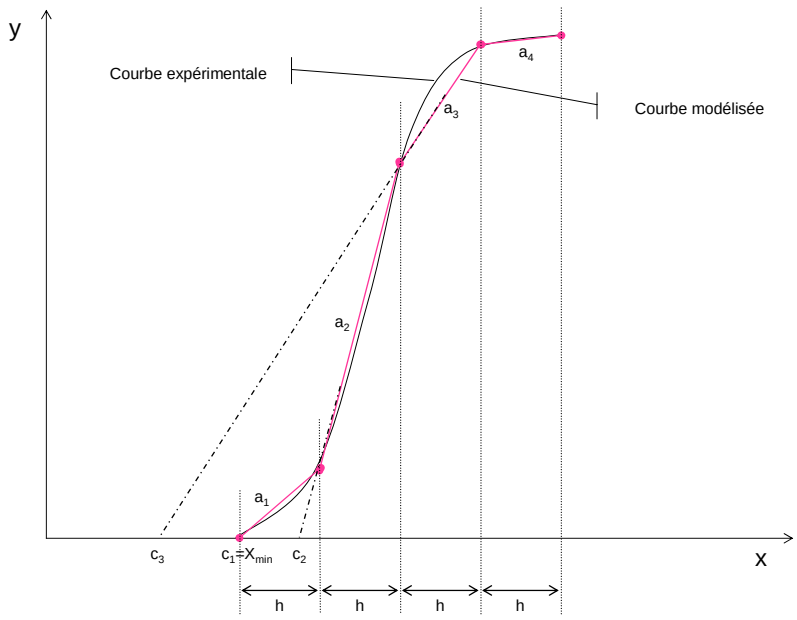


Figure 45 : Courbe modélisée par un ensemble de segments de droite

<u>Paramètres</u>

Les N paramètres a_i ainsi que c_i, h et x_{min} définissent le modèle.

<u>Mode opératoire</u>

Les paramètres a_i et c_i sont obtenus en discrétisant la courbe et en résolvant le système linéaire suivant :

$$\begin{cases} Y_i = a_i(X_i - c_i) \\ Y_{i+1} = a_i(X_{i+1} - c_i) \end{cases}$$

avec X_i et Y_i les $(N+1)$ valeurs des points expérimentaux et les conditions initiales suivantes : $\begin{cases} Y_0 = 0 \\ X_0 = X_{min} \end{cases}$

<u>Utilisation</u>

A partir d'un délai t donné, on recherche i tel que $t \in [x_{min} + (i-1).h,\ x_{min} + i.h]$ puis on obtient le pourcentage en calculant $a_i.(t-c_i)$.

Modèle polynomial

Expression

Ce modèle est basé sur une approximation de la courbe expérimentale par un polynôme de degré N, défini par :

$$P(x) = \sum_{i=0}^{N} a_i . x^i \quad \text{pour } x \in [x_{min}, x_{max}]$$

Paramètres

Les $N+1$ coefficients a_i ainsi que x_{min} et x_{max} sont les paramètres du modèle.

Mode opératoire

La valeur des coefficients est donnée par la procédure Matlab suivante :

– *[P,S] = polyfit*[25]*(X,Y,N)* avec :
  - *X* : valeurs expérimentales du délai → *X ;*
  - *Y* : valeurs expérimentales du pourcentage → *Y ;*
  - *N* : degré du polynôme ;
  - *P* : vecteur contenant les coefficients du polynôme (méthode des moindres carrés) ;
  - *S* : structure relative à l'erreur.

Utilisation

– Pour obtenir un pourcentage de paquets reçus à partir d'un délai *t*, il suffit de calculer la fonction Matlab *polyval(P,t)*.

La Figure 46 illustre les courbes expérimentale et polynomiale pour un polynôme de degré 5.

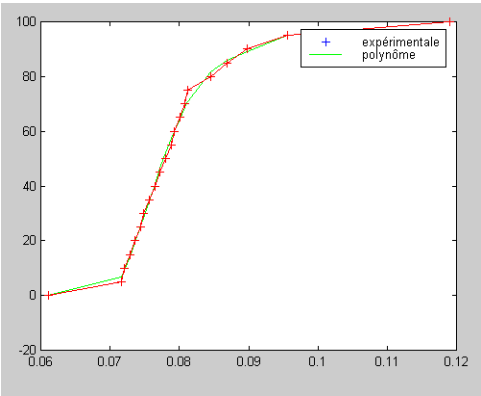


Figure 46 : Courbes expérimentale et polynomiale

[25] Fonction MatLab

Modélisation discrète (« en escalier »)

Expression

Cette solution consiste à ne choisir qu'un nombre limité de points N parmi les points expérimentaux.

$$\forall i \in [1,N]\ y_i = x\ \ \forall x \in [x_i, x_{i+1}]$$

La finesse de la modélisation dépend donc du nombre de points choisis. En fait, cette modélisation consiste à considérer la fonction de répartition comme une fonction en « escalier ». La figure suivante montre cette modélisation pour 5 points.

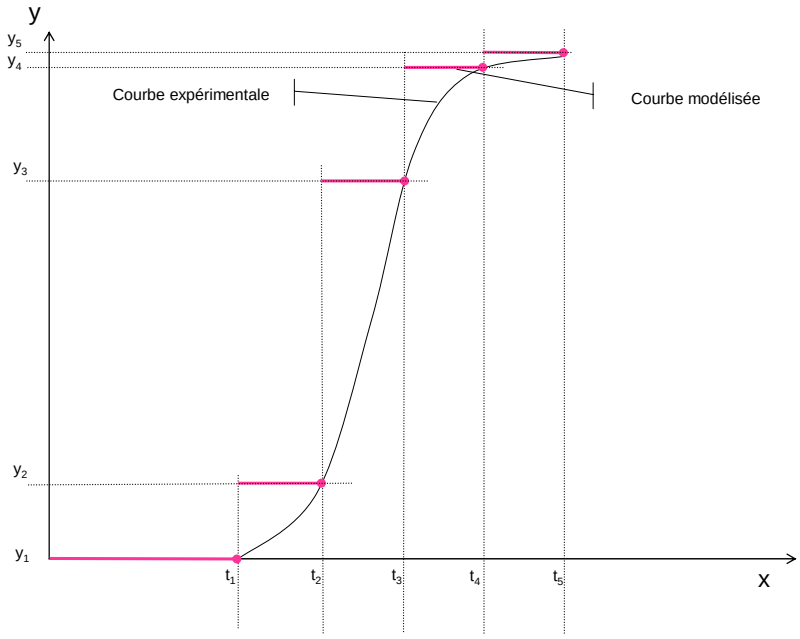


Figure 47 : Courbes expérimentale et modélisée en escalier

Paramètres

Les couples (x_i,y_i) sont les paramètres du modèle.

Mode opératoire

Cette modélisation est extrêmement simple à mettre en œuvre puisqu'il suffit de choisir des points parmi les points expérimentaux.

Utilisation

A partir d'un délai t donné, le pourcentage de paquets reçus y_i est défini tel que $t \geq x_i$.

1.2.1.2 Modèle retenu

Pour la suite de nos travaux, nous avons choisi le modèle discret (« en escalier ») pour sa simplicité : d'une part pour sa mise en œuvre afin de déduire à partir de données expérimentales les paramètres définissant le modèle, et d'autre part pour l'extension de la modélisation à un environnement multi domaines qui est présentée dans le paragraphe suivant.

Plusieurs tests de visioconférence sur une plate-forme d'émulation ont permis de montrer qu'un nombre réduit de points était tout à fait suffisant pour la perception humaine.

2 Caractérisation des services IP en environnement multi domaines

Ce paragraphe présente les travaux relatifs à la caractérisation des services IP dans un environnement multi domaines. Dans une première partie (2.1), nous présentons une étude mathématique conduisant à l'élaboration d'un modèle générique. Dans une deuxième partie (2.2), nous appliquons les résultats de cette étude au modèle analytique retenu en (1.2). Dans une troisième partie (2.3), nous validons l'étude précédente par une campagne de mesures menée dans un environnement réseau émulant (via le logiciel *Dummynet*) le comportement d'un environnement multi domaines.

2.1 Étude mathématique

L'objectif de cette étude est de définir comment caractériser le service résultant d'une communication traversant plusieurs domaines IP, caractérisés chacun (sous les conditions énoncées en conclusion du paragraphe (1.1.4)) par une fonction de répartition des délais de transit des paquets entre deux routeurs de bordure.

A titre d'exemple, la Figure 48 illustre deux domaines et les fonctions de répartition caractérisant le service de R1 à R2 et de R3 à R4.

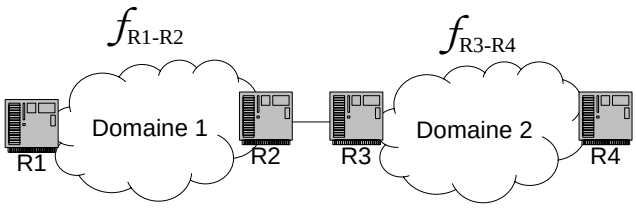


Figure 48 : Caractérisation multi domaines

2.1.1 Modèle mathématique sous-jacent à l'étude

Nous rappelons ici les propriétés mathématiques des fonctions de répartition nécessaires à notre étude.

<u>Variable aléatoire</u>

Dans un espace probabilisé fini Ω, on peut associer, à toute éventualité e_i, un nombre réel x_i ; on définit ainsi une application X de Ω dans $\Re$, appelée variable aléatoire.

Loi de probabilité et fonction de répartition

La loi de probabilité d'une variable aléatoire discrète est l'ensemble des couples (x_i,p_i) où p_i = Prob$\{X = x_i\}$, probabilité de réalisation de l'événement $\{X = x_i\}$. Cette loi de probabilité peut s'exprimer sous la forme de densité de probabilité, de fonction de répartition, … Dans un problème statistique, on cherche généralement la probabilité qu'une variable aléatoire soit inférieure ou supérieure à une valeur donnée, ou encore comprise entre deux valeurs données. On définit alors sa fonction de répartition F définie sur [0;1] par $F_X(x)$ = Prob$\{X \leq x\}$. C'est une fonction cumulative : si les x_i sont rangés par valeurs croissantes, alors :

pour tout $x < x_0$, $F_X(x) = 0$

$F_X(x_0)$ = Prob$\{X = x_0\}$

$F_X(x_1)$ = Prob$\{X = x_0\}$ + Prob$\{X = x_1\}$

$F_X(x_k)$ = Prob$\{X = x_0\}$ + ... + Prob$\{X = x_k\}$

$F_X(x_n) = 1$

pour tout $x > x_n$, $F_X(x) = 1$.

Dans le cas où X est une variable continue, alors $F_X(x)$ est définie par :

$F_X(x)$ = Prob$\{X<x\}$ et pour $x = x_0$, Prob$\{X=x_0\} = 0$.

Note : La dérivée de la fonction de répartition constitue la densité de probabilité.

Produit de convolution de lois de probabilités

On appelle produit de convolution, noté $*$, de deux lois de probabilités indépendantes P^X et P^Y sur $\Re$, la loi P^X*P^Y sur $\Re$ définie par :

$$\left(P^X * P^Y\right)(E) = \int_E \left(\int_E dP^Y(x-y) \right) dP^X(x)$$

Propriétés fondamentales

Si X et Y sont deux variables aléatoires réelles indépendantes, de lois P^X et P^Y, alors la loi de $X+Y$ est P^X*P^Y.

Le produit de convolution est commutatif et associatif.

2.1.2 Application au délai de transit des paquets d'un flux donné

La valeur des délais de transit des paquets d'un flux donné peut être considérée comme une variable aléatoire prenant ses valeurs dans un intervalle $[t_{min},\infty[$, t_{min} désignant le plus petit délai observable et ∞ correspondant au délai d'un paquet perdu (qui n'arrive donc jamais). La fonction de répartition des délais de transit des paquets d'un flux $F(t)$ représente alors le taux de paquets reçus ayant un délai de transit inférieur à t. Dans le cas où les paquets d'une communication traversent deux domaines caractérisés par des fonctions de répartition des délais F_X et F_Y, en considérant que les deux variables aléatoires X et Y (représentant les délais de transit dans le premier domaine et le second

domaine) sont indépendantes en probabilité, la fonction de répartition F_Z des délais de transit de bout en bout des paquets, avec $Z = X+Y$, est définie, d'après le paragraphe (2.1.1), par :

$$F_Z(t) = \frac{d}{dt}\left(F_X(t) * F_Y(t)\right)$$

La généralisation à la traversée de n domaines se fait simplement à partir du résultat obtenu avec $n–1$ domaines grâce à la propriété d'association du produit de convolution. Nous avons alors la relation de récurrence suivante :

$$F_{Z_n}(t) = \frac{d}{dt}\left(F_{X_n}(t) * F_{Z_{n-1}}(t)\right) \ \forall n \geq 2 \ et \ F_{Z_1}(t) = F_{X_1}(t)$$

avec $F_{X_n}(t)$ la fonction de répartition des délais de transit des paquets traversant le domaine n.

2.2 Application au modèle analytique retenu et validation

2.2.1 Application au modèle analytique retenu

Tel qu'exposé dans le paragraphe (1.2.1.2), le modèle analytique retenu pour caractériser les services IP d'un domaine DiffServ est une fonction en « escalier ». La figure suivante rappelle cette modélisation pour 5 points.

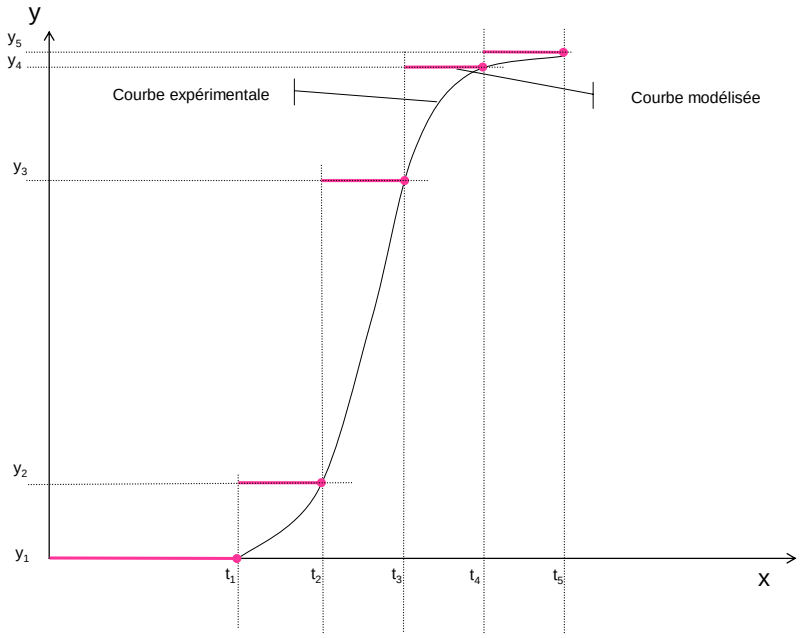


Figure 49 : Courbes expérimentale et en escalier

Ce modèle est extrêmement simple à mettre en œuvre puisqu'il suffit de choisir des points parmi les points expérimentaux. Il offre également une simplicité de calcul de la convolution de deux fonctions de répartition ainsi modélisées.

En effet, lorsque les lois de probabilité P^X et P^Y sont définies en seulement n points par :

$P^X(x_i) = F_X(x_i) - F_X(x_{i-1})\ \forall\ i \in \{2,..,n\}$ et $P^X(x_1) = 0$

$P^Y(y_j) = F_Y(y_j) - F_Y(y_{j-1})\ \forall\ j \in \{2,..,n\}$ et $P^Y(y_1) = 0$,

alors la loi de probabilité P^Z de $Z=X+Y$ est définie par :

$P^Z(z_k = x_i + y_j) = P^X(x_i) \,.\, P^Y(y_j)\ \forall\ i,j \in \{1,..,n\}$.

Ce résultat simplifie le calcul du produit de la convolution et sera utilisé par la suite lorsqu'il sera fait référence au produit de convolution.

La fonction de répartition du délai de bout en bout se déduit alors du calcul des probabilités cumulées en ordonnant (par ordre croissant) les délais z_k.

2.2.2 Validation de l'étude en émulation (Dummynet)

Nous avons défini une plate-forme d'émulation afin de pouvoir confronter les résultats mathématiques précédents à un contexte de réseaux de communication.

2.2.2.1 Introduction

L'émulation a été préférée à la simulation essentiellement pour son coté démonstratif ; nous n'en présentons ici que les grands principes ; une présentation plus détaillée sera fournie au Chapitre 5.

Le principe d'un émulateur de réseau est de placer des mécanismes en des points particuliers d'une plate-forme pour reproduire le comportement souhaité : médium de communication engendrant des pertes, du délai, des déséquencements de paquets, … À la différence de la simulation, l'émulation s'appuie sur des équipements réels, ce qui permet de mettre en place physiquement une plate-forme d'expérimentation.

De nombreux émulateurs sont disponibles avec des logiciels sous licence commerciale ou libre. Nous avons utilisé l'environnement FreeBSD, qui, par la commande *Dummynet*, permet de reproduire facilement des topologies complexes. Le principe de ce logiciel est d'intercepter, au niveau des cartes réseaux d'un PC émulateur, les paquets IP (que le paquet ait pour source ou destination l'adresse IP associée à cette carte). Nous avons pu configurer très simplement le comportement d'un réseau en permettant d'en modifier, pour une route donnée, les caractéristiques suivantes : délai de transit, capacité du lien, taux de perte de paquets.

2.2.2.2 Émulation d'un environnement mono domaine DiffServ

Nous avons vu dans les paragraphes précédents que nous pouvions caractériser les services IP entre deux routeurs de bordure d'un domaine DiffServ (pour une route et un état de charge donnés) par une fonction de répartition du délai de transit de paquets modélisable par une courbe relativement simple. Un routeur émulateur (sur lequel on a configuré *Dummynet*) assure que les paquets d'un flux

donné le traversant vont subir les mêmes conditions (en termes de délai et de perte) que s'ils traversaient un domaine DiffServ.

Pour cela, soit *n* points d'une courbe représentative d'une fonction de répartition réelle (par exemple extraite des campagnes d'expérimentation) qui donnent la probabilité *p* pour que les paquets aient un délai de transit inférieur à *t*.

Dummynet permet de définir des canaux logiques (appelés *tubes*) dans lesquels les paquets vont passer. Chaque tube est caractérisable par un délai de transit (entre autres) et *Dummynet* donne la possibilité de définir une probabilité de passage dans chacun des tubes.

Il suffit de faire correspondre les probabilités de passage dans un tube avec les délais de transit en accord avec la fonction de répartition choisie.

Pour émuler la traversée d'un domaine entre les routeurs R1 et R2 d'un domaine DiffServ (Figure 50), il suffit ainsi de mettre en place *n* tubes de délai t_i, chaque paquet ayant une probabilité p_i de passer dans le tube *i*.

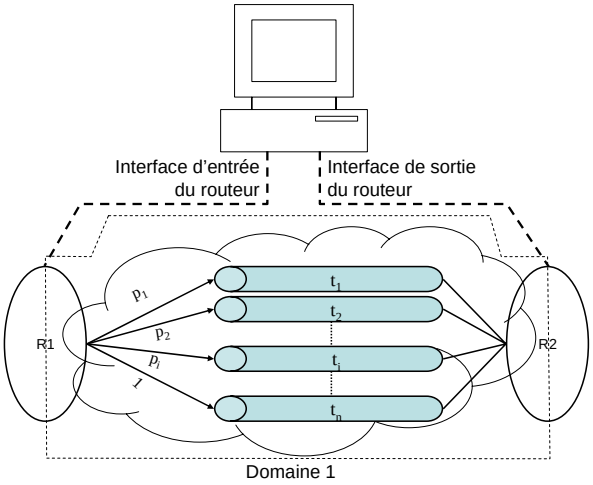


Figure 50 : Définition d'un domaine DiffServ sous *Dummynet*

2.2.2.3 Émulation d'un environnement multi domaines DiffServ

Pour émuler un environnement DiffServ multi domaines, nous avons défini pour chaque domaine un ensemble de règles émulant leur traversée entre deux points donnés. La traversée de plusieurs domaines a été émulée en définissant plusieurs règles *Dummynet* qui s'enchaînent l'une à la suite de l'autre.

L'exemple suivant illustre l'enchaînement des règles nécessaires pour avoir un comportement correspondant à la traversée de deux domaines. Dans cet exemple, chaque domaine est défini par 3 tubes et le filtrage s'effectue sur les paquets ICMP (IP) émis par l'adresse IP *adresse_source* à destination de l'adresse IP *adresse_destination ;* la commande *skipto* permet de passer directement à une autre règle, ce qui permet d'enchaîner les domaines.

```
# autoriser plusieurs passages dans Dummynet pour enchaîner les règles
sysctl net.inet.ip.fw.one_pass=0

##############################################################
#------------------ PREMIER DOMAINE -------------------------------
# création des règles à appliquer en entrant dans le premier domaine
ipfw add 101 prob 0.33 skipto 111 ip from adresse_source to adresse_destination
ipfw add 102 prob 0.66 skipto 112 ip from adresse_source to adresse_destination
ipfw add 103 skipto 113 ip from adresse_source to adresse_destination

# définition des tubes du premier domaine
# premier tube
ipfw add 111 pipe 111 ip from adresse_source to adresse_destination
ipfw pipe 111 config delay 20ms
ipfw add 111 skipto 201 ip from adresse_source to adresse_destination
# deuxième tube
…
# troisième tube
…

##############################################################
#------------------ SECOND  DOMAINE -------------------------------
# création des règles à appliquer en entrant dans le second domaine

ipfw add 201 prob 0.33 skipto 211 ip from adresse_source to adresse_destination
ipfw add 202 prob 0.66 skipto 212 ip from adresse_source to adresse_destination
ipfw add 203 skipto 213 ip from adresse_source to adresse_destination

# définition des tubes du second domaine
# premier tube
…
ipfw add 211 skipto 65000 ip from adresse_source to adresse_destination
# deuxième tube
…
## DERNIERE REGLE  -------------------------------
ipfw add 65000 allow ip from adresse_destination to adresse_destination
```

2.2.2.4 Tests expérimentaux

Nous avons créé deux domaines consécutifs caractérisés chacun par une fonction de répartition (F_X et F_Y) réduite à 5 points.

Les tests de mesures du délai ont été réalisés en utilisant la commande *ping* entre l'émetteur et le récepteur.

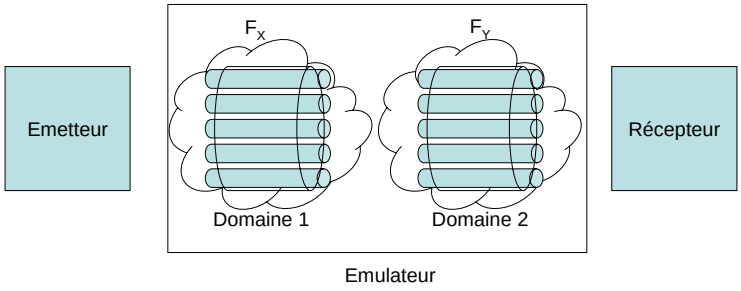


Figure 51 : Configuration des tests

L'objectif des tests est de vérifier que le délai de transit, de hôte émetteur à hôte récepteur, des paquets du flux généré par la commande *ping* vérifie bien la loi de probabilité (définie par la fonction de répartition) $F_Z = F_X * F_Y$.

Scénarios

Le Tableau 21 montre les 5 points choisis pour décrire les fonctions de répartition issues des expérimentations réalisées sur la plate-forme @IRS.

F_X		F_Y	
délai (ms)	probabilité	délai (ms)	probabilité
29	0.075	31	0.075
37	0.275	35	0.275
44	0.825	45	0.925
50	0.85	47	0.95
57	1	49	1

Tableau 21 : Caractérisation des fonctions de répartition

Les tests durent 5 minutes durant lesquelles un paquet de 1024 Octets est envoyé toutes les 0.5 seconde[26]. Pour chaque scénario, les tests ont été réalisés deux fois. Les scénarios et les noms des fonctions de répartition obtenues sont représentés dans le Tableau 22.

Scénarios		Nom de la courbe
1	seul le domaine 1 est émulé	domaine 1 test1 (et test 2)
2	seul le domaine 2 est émulé	domaine 2 test1 (et test2)
3	les 2 domaines sont émulés	multi 1er test (et 2ème test)

Tableau 22 : Scénarios et noms des courbes résultats

Résultats et analyse

Les résultats sont représentés sur la Figure 52.

La courbe appelée « convolution des 2 domaines » représente la convolution des points issus des deux courbes « domaine 1 test 1 » et « domaine 2 test 1 ».

La courbe appelée « convolution des 5 points » est tracée à partir du résultat du calcul de la convolution mathématique des 5 points du Tableau 21.

[26] Les tests ont été effectués avec des débits faibles pour des raisons de limites imposées par cette approche. Nous reviendrons dans le Chapitre 5 sur ces limites en précisant les contraintes à respecter.

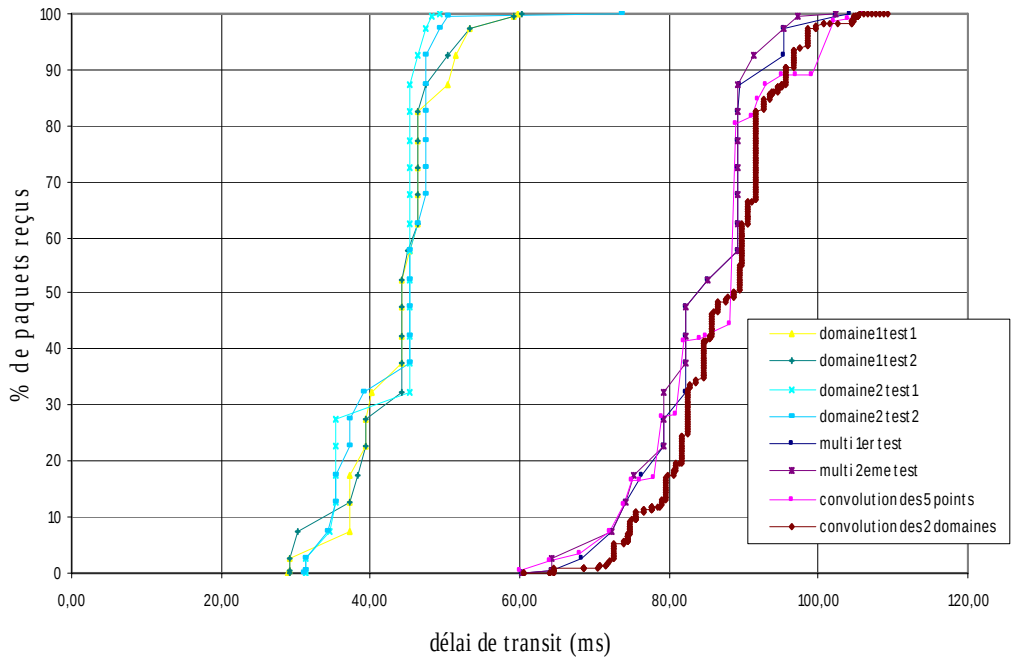


Figure 52 : Fonctions de répartition des délais de transit des paquets

Cette double série de tests permet de voir qu'il y a peu d'écart entre deux expérimentations : l'émulateur a donc un comportement répétitif aux cours des tests.

L'ensemble des tests permet de vérifier que dans une situation émulant une plate-forme réseau de type DiffServ, le résultat mathématique sur la convolution des lois de probabilité de deux variables aléatoires peut être retenu pour l'émulation d'un environnement DiffServ multi domaines.

Nous verrons dans le Chapitre 5 comment utiliser cette plate-forme pour déployer notre solution d'architecture de signalisation multi domaines.

3 Caractérisation du couplage Transport / IP

Ce paragraphe présente les travaux relatifs à la caractérisation de la QdS des services de bout en bout couplant niveaux IP et Transport offerts par notre architecture de communication dans un environnement multi domaines.

3.1 Caractérisation de la fiabilité

Soit ε le pourcentage de paquets perdus par congestion des routeurs du réseau.

La caractérisation du système en terme de fiabilité s'appuie sur une estimation du pourcentage de paquets reçus qui est :

- sans retransmission, égal à : $1 - \varepsilon$
- après 1 retransmission, égal à : $(1 - \varepsilon) + \varepsilon . (1 - \varepsilon) = 1 - \varepsilon^2$
- après 2 retransmissions, égal à : $(1 - \varepsilon) + \varepsilon . (1 - \varepsilon) + \varepsilon . [\varepsilon . (1 - \varepsilon)] = 1 - \varepsilon^3$
- …

– après n retransmissions, égal à : $1 - \varepsilon^{n+1}$

En définitive, pour un service de niveau IP dont le pourcentage de paquets perdus est connu (tels que les service AS ou GS), le pourcentage r_n de paquets reçus après n retransmissions peut être exprimé comme suit :

$$r_n = 1 - \varepsilon^{n+1}$$

ε désignant le pourcentage moyen de perte de paquets IP entre 2 sites sans aucune retransmission au niveau Transport.

3.2 Caractérisation du délai

Soit $f(t)$ la fonction représentant la probabilité qu'un paquet soit reçu (sans retransmission) avec un délai de transit inférieur à t. La caractérisation du système en terme de délai s'appuie sur l'estimation du pourcentage des paquets reçus avec un délai inférieur à un temps t donné.

Considérons à présent la possibilité que soient effectuées une ou plusieurs retransmissions des PDU de niveau Transport perdus par le réseau.

Soit $f_{n,T}(t)$ la fonction de répartition du délai après n retransmissions éventuelles :

– la relation entre $f_{1,T}(t)$ et $f(t)$ est alors : $f_{1,T}(t) = f(t) + \varepsilon \,.\, f(t - T)$, où :
  - ε désigne le pourcentage de perte moyen du réseau ;
  - T désigne la valeur du « *timer* » de retransmission des PDU de niveau Transport (supposée constante dans cette étude).
– la relation entre $f_{2,T}(t)$ et $f(t)$ est alors : $f_{2,T}(t) = f(t) + \varepsilon \,.\, f(t - T) + \varepsilon\,[\varepsilon \,.\, f(t - 2T)]$
– …
– la relation entre $f_{n,T}(t)$ et $f(t)$ est alors : $f_{n,T}(t) = f(t) + \varepsilon \,.\, f(t - T) + \ldots + \varepsilon\,[\varepsilon^{n-1} \,.\, f(t - nT)]$

En définitive, pour un service de niveau IP dont la fonction de répartition du délai est connu (tels que les service AS ou GS), la fonction $f_{n,T}(t)$ de répartition du délai de transit après n retransmissions peut être exprimée par la relation suivante :

$$f_{n,T}(t) = \sum_{i=0}^{n} \varepsilon^{i} \,.\, f(t - i.T)$$

$f(t)$ désignant la fonction de répartition du délai de transit des paquets IP entre 2 sites sans aucune retransmission au niveau Transport

Sur les bases de cette caractérisation, nous présentons maintenant l'algorithme de sélection automatique des services.

4 Algorithme de sélection

Ce paragraphe présente l'algorithme de sélection automatique des services de niveaux IP et Transport permettant de satisfaire la QdS requise par une application pour un canal de communication, dans un environnement multi domaines.

4.1 Hypothèses

L'algorithme de sélection automatique des services repose sur trois hypothèses :

- la première est que les pertes dans le réseau sont indépendantes entre elles ; cette hypothèse peut être discutée dans la mesure où ce type de pertes est essentiellement dû à des congestions de routeurs entraînant des rafales de pertes ; cependant, ces rafales portent sur des paquets qui n'appartiennent pas obligatoirement au même flux ;
- la seconde hypothèse réside dans la disponibilité de trois types de services au niveau IP, caractérisables, entre deux sites donnés (LAAS et LIP6 par exemple), en termes de taux de perte et de délai de transit des paquets IP :
  - un service garanti, tel que le service GS dans notre architecture, caractérisable par un taux de perte nul aux erreurs bits près (i.e. pas de perte par congestion des routeurs intermédiaires), et un délai de transit correspondant au délai minimal entre les sites ;
  - un service assuré, tel que le service AS dans notre architecture, caractérisable par une fonction de répartition du délai de transit et un taux de pertes moyen. Notons ici que la définition des sémantiques de garantie M et N vue au Chapitre 1 a pour objectif d'établir un lien entre la sémantique des services tels que AS et la tolérance de certaines applications à la fluctuation du réseau en terme de QdS ;
  - un service de type BE, n'offrant aucune garantie, ni sur le délai, ni sur le taux de perte ;
- la troisième hypothèse réside dans la disponibilité de trois services de niveau Transport :
  - ceux offerts par UDP et TCP ;
  - celui offert par le protocole FPTP que nous supposerons ici paramétrable par un pourcentage maximal admissible de paquets FPTP perdus (chaque PDU FPTP véhiculant un ADU[27]).

4.2 Principe de l'algorithme de sélection automatique des services

Afin de juger si une requête applicative peut être satisfaite, il s'agit successivement :

- de vérifier la cohérence de la requête et de rejeter celle-ci en cas d'incohérence ;
- puis en cas de cohérence :
  - d'effectuer le choix du couple (service IP, service Transport) permettant de satisfaire la requête ;

[27] ADU : Application Data Unit

- d'effectuer alors un contrôle d'admission, dès lors que le service IP retenu diffère du *Best Effort*. En cas d'échec de ce contrôle, il s'agit alors de revenir au point précédent et de tester un autre couple s'il y en a. En cas d'échec pour tous les couples, la requête est rejetée.

Étudions à présent plus en détail les deux premiers points de l'algorithme.

Vérification de la cohérence de la requête

Pour qu'une requête applicative soit jugée cohérente, deux conditions (évidentes) doivent être vérifiées[28] :

– le taux des paquets que souhaite recevoir l'application doit être supérieur au taux de paquets que souhaite recevoir l'application dans les délais, soit : $\boldsymbol{0 \leq \tau_d \leq \tau_r \leq 1}$;
– le délai maximal de transit, b, doit être supérieur au délai de transit minimum des paquets, noté t_0, observé entre les deux sites considérés, soit : $\boldsymbol{b > t_0}$.

Sélection du couple (service IP, service Transport)

Une fois la cohérence vérifiée, il s'agit de choisir le service de niveau IP et le service de Transport permettant de satisfaire les paramètres de QdS de la requête ainsi que leur sémantique de garantie.

Ce choix repose sur la caractérisation des performances des services de niveau IP, exprimées en les termes définis dans la section précédente : fonction de répartition du délai de transit et fiabilité moyenne après n retransmissions ($f_{n,T}$ et r_n). Notons ici que si la cohérence est vérifiée, il y aura toujours au moins un couple (service Transport, service IP) permettant de satisfaire la requête ; ceci étant, dans le cas où le service IP diffère du BE, l'acceptation de la requête sera conditionnée par une réponse positive de la part du contrôle d'admission.

En premier lieu, il s'agit donc de déterminer le nombre n de retransmissions des PDU Transport permettant de satisfaire la fiabilité requise tout en respectant le délai maximal requis. Ce nombre est évalué de la façon suivante. Soit :

– n_r le nombre de retransmissions nécessaires à la satisfaction du paramètre τ_r :
  - $\tau_r \leq 1 - \varepsilon^{nr+1} \Rightarrow n_r \leq ln\ (1 - \tau_r) / ln(\varepsilon)$ où : ln désigne la fonction logarithme népérien et ε le pourcentage moyen de perte ;
– n_d le nombre maximum de retransmissions compatible avec le paramètre b :
  - $n_d = Ent\ [(b - t_0) / T]$ où : Ent désigne la fonction partie entière, T la valeur du temporisateur de retransmission des TPDU (supposée fixe dans cette étude) et t_0 le délai minimum observé entre les deux sites considérés ;

  Il vient alors que : $\boldsymbol{n = Min\ (n_r, n_d)}$.

  Les principes de l'algorithme de sélection sont les suivants.

[28] Voir le paragraphe (3.1) du Chapitre 2 pour les définitions des paramètres τ_r, τ_d et b.

Un tableau (voir Tableau 23) indique, pour toutes les configurations possibles de (τ_r, σ_r[29]) et (τ_d, σ_d), un ou plusieurs couples (service Transport, service IP), par exemple (UDP/GS), permettant de satisfaire la requête. Lorsque plusieurs couples sont envisageables, ceux-ci sont classés par ordre de « coût », les critères de coût étant les suivants :

– au niveau IP : coût du service GS > coût du service AS > coût du service BE ;
– au niveau Transport : coût du protocole TCP > coût du protocole FPTP > coût du protocole UDP ;
– priorité à la minimisation du coût du service de niveau IP.

<table>
<tr><th>τ_d \ τ_r</th><th>$\tau_r = 0$</th><th colspan="4">$0 < \tau_r < 1$</th><th>$\tau_r = 1$</th></tr>
<tr><td>$\tau_d = 0$</td><td>UDP/BE
∀ la sémantique</td><td colspan="4">FPTP/BE
∀ la sémantique A, M ou N</td><td>TCP/BE
∀ la sémantique</td></tr>
<tr><td rowspan="4">$0 < \tau_d < 1$; [a,b]</td><td rowspan="4">∅</td><td>σ_d \ σ_r</td><td>A</td><td>M</td><td>N</td><td rowspan="4">UDP/GS
∀ la sémantique</td></tr>
<tr><td>A</td><td>UDP GS</td><td>UDP GS</td><td>UDP GS</td></tr>
<tr><td>M</td><td>UDP GS</td><td>C1</td><td>C2</td></tr>
<tr><td>N</td><td>UDP GS</td><td>C3</td><td>C4</td></tr>
<tr><td>$\tau_d = 1$; [a,b]</td><td>∅</td><td colspan="4">∅</td><td>UDP/GS
∀ la sémantique</td></tr>
</table>

Tableau 23 : Correspondance entre les paramètres de QdS et les services IP et Transport

Les cases notées ∅ indiquent une incohérence dans la requête, à savoir que la relation :

$0 \leq \tau_d \leq \tau_r \leq 1$ n'est pas satisfaite.

Ayant à satisfaire une requête formulée en termes d'un ensemble {(τ_r, *[0, b]*, σ_r) ; (τ_d, σ_d)}, l'algorithme accède à la case du tableau correspondant à la requête, puis évalue si le couple (service IP, service Transport) le moins coûteux permet de satisfaire la requête. Si tel est le cas, il sélectionne ce couple, sinon il teste le couple suivant.

Étudions à présent comment sont initialisées les cases du tableau. La construction du tableau est fonction :

– des services disponibles : UDP, TCP et FPTP au niveau Transport, GS, AS et BE au niveau IP ;
– de l'ensemble de définition des paramètres et des sémantiques, soit pour τ_r et τ_d : l'intervalle [0,1] (découpé en 3 sous-intervalles : 0,]0,1[et 1), et pour σ_r et σ_d : l'ensemble {A,M,N}.

Pour chacune des combinaisons (τ_r, σ_r) / (τ_d, σ_d) possibles, nous avons déterminé toutes les solutions permettant de satisfaire chacun des couples pris séparément, puis nous avons reporté dans le tableau la ou les solutions satisfaisant les deux couples conjointement. Illustrons la démarche adoptée par le biais de l'exemple suivant.

[29] σ_r (resp. σ_r) représente la sémantique associée au paramètre τ_r (resp. τ_d)

Soit à étudier la combinaison ($0 < \tau_r < 1$, $\sigma_r = A$) / ($0 < \tau_d < 1$, $[0, b]$, $\sigma_d = M$) :

- pour satisfaire la requête relative au couple (τ_r, σ_r), deux possibilités sont envisageables :
  - * / GS, où * désigne n'importe quel service de Transport ;
  - $FPTP_\infty$ / *, où * désigne n'importe quel service de niveau IP et ∞ un nombre de retransmissions non borné ;
- pour satisfaire la requête relative au couple (τ_d, σ_d), trois possibilités sont envisageables :
  - UDP / AS si $\tau_d \leq f(b)$;
  - $FPTP_n$ / AS si $\tau_d \leq f_{n,T}(b)$, où n indique que FPTP se borne à ré émettre au plus n fois un PDU non acquitté ;
  - UDP / GS si $b \geq t_0$ (condition qui est automatiquement vérifiée dans le test de la cohérence de la requête).

L'unique solution satisfaisant les deux couples et qui est reportée dans le tableau est donc : UDP/GS.

Notons enfin que les performances des services (t_0 pour GS et ($f(b)$, ε) pour AS), pour chaque route envisagée, sont accessibles dans des équipements dont nous étudierons le fonctionnement dans le Chapitre 4.

4.3 Exemple

Illustrons ces propos par deux exemples :

- pour une application requérant : $\tau_r = 0.75$, $\sigma_r = M$ et $\tau_d \leq 0.70$, $b = 125$ ms, $\sigma_d = A$, il n'y a qu'un seul choix possible : le couple (UDP, GS) de la case grisée clair du tableau ; ce couple permet de satisfaire la requête sous réserve que le délai minimal t_0 observé entre les deux sites considérés soit inférieur ou égal à 125 ms (condition évaluée lors du test de la cohérence de la requête) ;
- pour une application requérant : $\tau_r = 0.80$, $\sigma_r = M$ et $\tau_d \leq 0.70$, $b = 125$ ms, $\sigma_d = M$, la case C1 (en gris foncé sur le tableau) comporte plusieurs solutions. L'algorithme sélectionnera par ordre de préférence décroissant :
  - soit le couple (UDP, AS) si $\tau_r \leq 1 - \varepsilon$ et $\tau_d \leq f(b)$;
  - soit le couple (FPTP avec n retransmissions, AS) si $\tau_r \leq 1 - \varepsilon^n$ et $\tau_d \leq f_{n,T}(b)$;
  - soit le couple (UDP, GS) si $t_0 \leq 125$ ms (ce que le test de cohérence a vérifié).

Notons que le couplage des services respectivement fournis par TCP au niveau Transport et par GS ou AS au niveau IP, n'a pas été approfondi dans notre étude car il n'intervient pas dans le tableau, cependant dans (Lochin, 2004) les lecteurs pourront trouver ces résultats complémentaires.

5 Conclusion

Ce chapitre a tout d'abord présenté une étude portant sur la caractérisation des services IP dans un environnement mono domaine puis multi domaines. Cette étude est la suite des expérimentations réalisées sur la plate-forme @IRS vues au chapitre précédent.

En environnement mono domaine, des simulations ns-2 ont permis d'étendre le domaine de validité des résultats expérimentaux, à savoir que dans une configuration de charge donnée et pour une route donnée, entre deux équipements de bordure, les performances des services des classes IP AF et EF ne dépendent pas ou peu du nombre et de la charge des flux à QdS. À la suite de ces simulations, nous avons proposé un modèle analytique des performances des services IP basé sur la fonction de répartition des délais de bout en bout des paquets.

En environnement multi domaines, nous avons rappelé certaines des propriétés de la fonction de répartition et de la convolution, outil indispensable à la caractérisation des services dans ce contexte. Des tests sur une plate-forme émulant le comportement de domaines DiffServ a permis de valider le choix de notre modèle.

Dans un second temps, nous avons caractérisé le service de bout en bout résultant du couplage des services Transport, parmi ceux offerts par les protocoles UDP, TCP et FPTP, et IP.

Enfin, cette caractérisation a été intégrée dans un mécanisme de sélection automatique des services présenté dans le quatrième paragraphe. Ce mécanisme de sélection vient compléter notre système de communication en permettant aux programmeurs d'application de s'affranchir du paramétrage des connexions.

Concernant la caractérisation des services de l'architecture de communication, plusieurs limites restent cependant ouvertes sur lesquelles nous reviendrons dans la conclusion générale de ce mémoire :

- le modèle de source de trafic que nous avons considéré lors des simulations est simple : c'est une source à débit constant. Des modèles plus évolués, basés par exemple sur le re-jeu de trace réelle, pourraient être étudiés ;
- l'étude de l'impact des mécanismes de contrôle de congestion sur la QdS de bout en bout n'a pas été effectuée. En effet, l'ensemble des tests a été fait en s'appuyant au niveau Transport sur UDP. Il serait intéressant d'évaluer les conséquences de l'utilisation de protocoles de Transport intégrant de nouveaux mécanismes comme TFRC ;
- enfin, la caractérisation du service de bout en bout au travers d'un environnement multi domaines repose sur les propriétés mathématiques de la convolution. Pour cela, nous avons dû faire l'hypothèse que le délai de transit des paquets d'un flux pouvait être assimilé à une variable aléatoire et qu'il y avait indépendance dans la répartition des délais lorsque nous considérons la traversée de plusieurs domaines. Cette hypothèse, qui nous paraît réaliste, est cependant à étayer.

Chapitre 4 Architecture de signalisation multi domaines

Introduction

Ce chapitre présente notre proposition d'architecture de signalisation pour le multi domaines, qui constitue la contribution majeure de cette thèse.

Le chapitre est structuré en trois parties. Le paragraphe (1) expose le rôle des différents protocoles de l'architecture, leur localisation, leurs règles d'enchaînement ainsi que le format de leurs PDU. Le paragraphe (2) présente la spécification UML 2.0 de l'architecture de signalisation déployée sur chacun des éléments majeurs de notre système (hôte, *bandwidth broker* et routeur de bordure). Les diagrammes de séquences d'un certain nombre de scénarios sont également fournis. En conclusion de ce chapitre (paragraphe 3) sont discutées les principales limites de notre proposition d'architecture.

1 Définition de l'architecture

L'architecture de signalisation que nous proposons est intégrée dans trois types d'équipement :

- les hôtes d'extrémité[30] (émetteur et récepteur) ;
- le routeur de bordure du site d'extrémité côté hôte émetteur (R_{bR}) ;
- les *Bandwidth Brokers* (BB).

La Figure 53 illustre les différents équipements d'une plate-forme multi domaines typique.

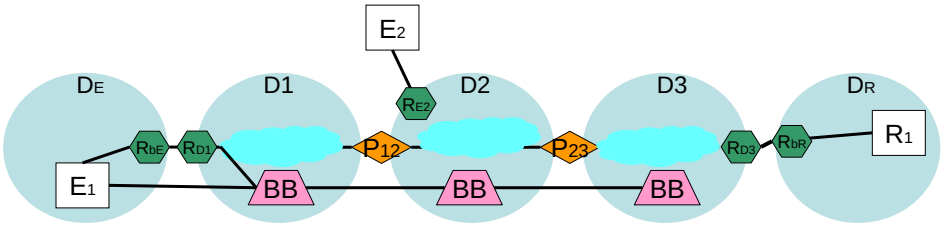


Figure 53 : Plate-forme multi domaines

Nous distinguerons deux rôles pour les BB selon qu'ils sont situés dans le domaine de l'hôte émetteur (domaine local) ou dans un domaine distant. Dans la suite du document, nous les appellerons respectivement BB *local* ou BB *remote*. Les domaines D_E et D_R ne sont pas considérés comme des domaines DiffServ, mais comme des réseaux locaux surdimensionnés ; ils n'ont de ce fait pas de BB.

Quatre protocoles de signalisation sont définis :

[30] Nous considérons que l'émetteur E_1 se trouve dans un réseau d'entreprise surdimensionné – et non DiffServ donc sans BB–. Le cas d'un émetteur relié directement à un domaine DiffServ est représenté par l'émetteur E_2. Il est de même pour le récepteur.

– le protocole de création d'un canal, mis en œuvre entre l'hôte émetteur et l'hôte récepteur ;
– le protocole de gestion de la QdS, mis en œuvre entre l'hôte émetteur et le BB *local* ;
– le protocole de signalisation, mis en œuvre entre les BB ;
– le protocole de configuration du routeur de bordure, mis en œuvre entre le BB *local* et le routeur de bordure du site émetteur.

Pour l'échange des PDU, ces protocoles se basent sur le service offert par le protocole de Transport TCP (qui garantit ordre et fiabilité sur le transfert des données), excepté pour les PDU de rafraîchissement et de libération des ressources, qui sont véhiculés en UDP et pour lesquels des mécanismes de fiabilisation basés sur des *timers* sont mis en place.

Notons que notre proposition d'architecture de signalisation multi domaines est compatible avec un environnement mono domaine, constituant en effet un cas particulier du multi domaines.

Les paragraphes suivants vont permettre de détailler : la localisation des trois protocoles (1.1), leur rôle (1.2) et leur enchaînement (1.3).

1.1 Localisation

Comme nous l'avons précisé précédemment, notre architecture est déployée sur trois types d'équipement.

1.1.1 Hôtes d'extrémité

Nous distinguons l'architecture de l'émetteur de celle du récepteur.

Dans un hôte émetteur, Figure 54, nous retrouvons deux des protocoles :
– le protocole de création d'un canal ;
– le protocole de gestion de la QdS.

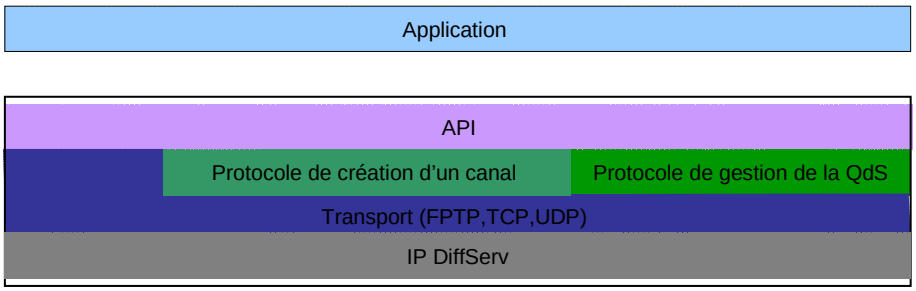


Figure 54 : Hôte émetteur

Dans un hôte récepteur, Figure 55, seul le protocole de création d'un canal est intégré.

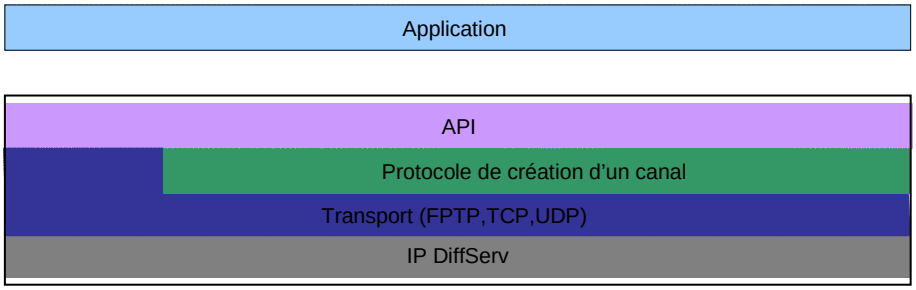


Figure 55 : Hôte récepteur

1.1.2 BB local et remote

L'architecture d'un BB *local* intègre trois protocoles de signalisation (Figure 56) :

- le protocole de gestion de la QdS ;
- le protocole de signalisation inter BB ;
- le protocole de configuration du routeur de bordure.

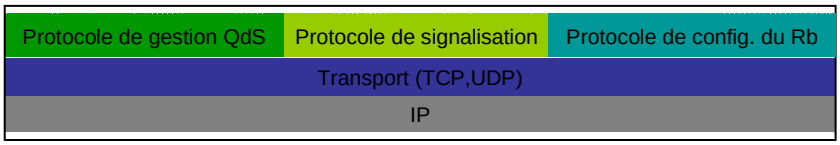


Figure 56 : BB *local*

L'architecture d'un BB *remote* (Figure 57) n'intègre que le protocole de signalisation inter BB.

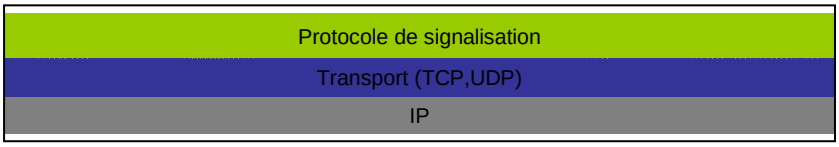


Figure 57 : BB *remote*

1.1.3 Routeur de bordure

Le routeur de bordure du site émetteur intègre le protocole de configuration du routeur de bordure, Figure 58.

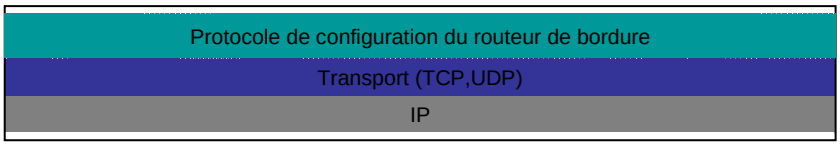


Figure 58 : Routeur de bordure

1.2 Rôle des différents protocoles

Dans ce paragraphe, nous donnons le rôle et la description sommaire des différents protocoles. Le paragraphe 2 fournit la spécification UML 2.0 de l'architecture de ces protocoles sur chacun des équipements impliqués. Les noms des PDU qui seront présentés en détail dans le paragraphe (1.4) sont basés sur la sémantique suivante. PDU_*X* est un PDU du protocole :

- de création d'un canal, si $X = 1$;
- de gestion de la QdS, si $X = 2$;
- de signalisation, si $X = 3$;
- de configuration du routeur de bordure, si $X = 4$.

Les noms des PDU s'apparentent à ceux définis dans l'approche COPS.

Le **protocole de création d'un canal** permet aux deux entités d'application d'établir (si possible) entre elles un canal de bout en bout (dans le cadre d'une session déjà établie) offrant une QdS correspondant à celle requise.

Recevant un PDU REQUEST_1 contenant la QdS requise en réception, l'entité émettrice vérifie si cette QdS correspond bien à celle annoncée par l'entité d'application émettrice (faute de quoi le canal n'est pas créé). S'il y a correspondance, la création du canal est effectuée si la QdS requise peut être satisfaite (condition testée par l'activation et la réponse du protocole de gestion de la QdS introduit ci-après) ; l'entité émettrice émet alors un PDU ANSWER_1 à destination de l'entité réceptrice l'informant de l'acceptation ou non de la requête (et du protocole de Transport à activer en conséquence si la requête est acceptée).

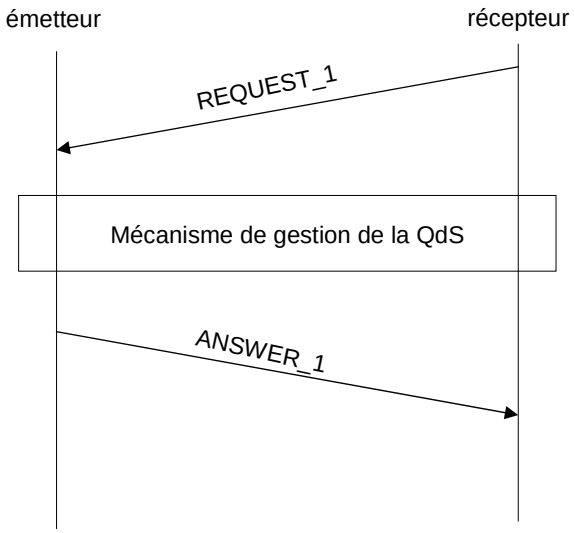


Figure 59 : Protocole de création d'un canal

Le **protocole de gestion de la QdS** est mis en œuvre entre l'émetteur et le BB *local*.

Il permet à l'hôte émetteur d'exprimer une requête QdS au BB *local* et de recevoir en retour :

- si la requête est satisfaisable : le protocole de Transport et la classe IP adaptés ;

– sinon : une notification du rejet de la requête.

Il permet (via un PDU CLOSE_2 émis à destination du BB *local*) de requérir le relâchement d'une réservation effectuée pour le canal considéré.

Il permet enfin (via un PDU REFRESH_2) de réamorcer le *timer* du BB *local* associé à la réservation effectuée pour le canal considéré, ce *timer* permettant de pallier le risque que les ressources ne soient pas relâchées si (par exemple) la demande de relâchement de réservation s'est perdue ou si l'hôte émetteur tombe en panne.

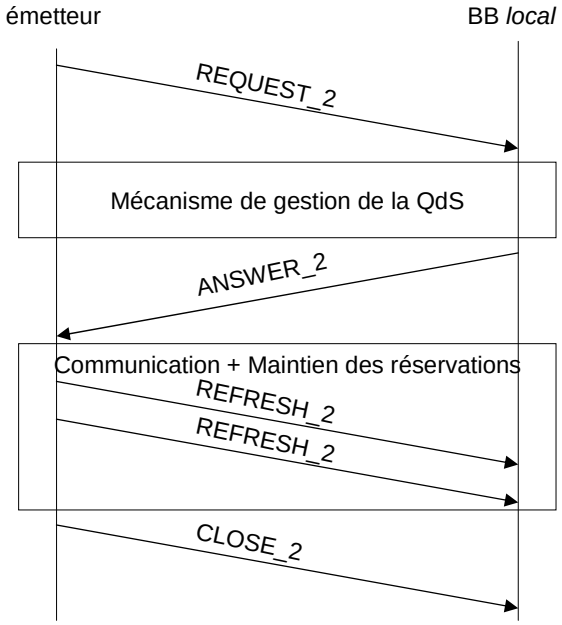


Figure 60 : Protocole de gestion de la QdS

Le **protocole de signalisation inter BB** est mis en œuvre entre un BB *local* et un BB *remote* ou entre deux BB *remote*.

Via un PDU REQUEST_3, il permet à un BB de demander les caractéristiques et la disponibilité des différents services sur un AS donné, et d'effectuer une pré réservation auprès du BB de cet AS. La réponse à cette requête est effectuée via un PDU ANSWER_3, toujours émis à destination du BB *local*.

Il permet également d'entériner une pré réservation (via un PDU RESERVE_3) ou de relâcher une réservation effectuée pour un canal (via un PDU CLOSE_3).

Enfin, il permet (via un PDU REFRESH_3 émis par le BB *local*) de réamorcer le *timer* associé à une réservation (effectuée pour un canal) de chacun des différents BB *remote*, ces *timers* permettant de pallier le risque que les ressources ne soient pas relâchées si (par exemple) une demande de relâchement de réservation s'est perdue ou si le BB local tombe en panne.

Le PDU RELEASE_3 permet de relâcher les ressources réservées.

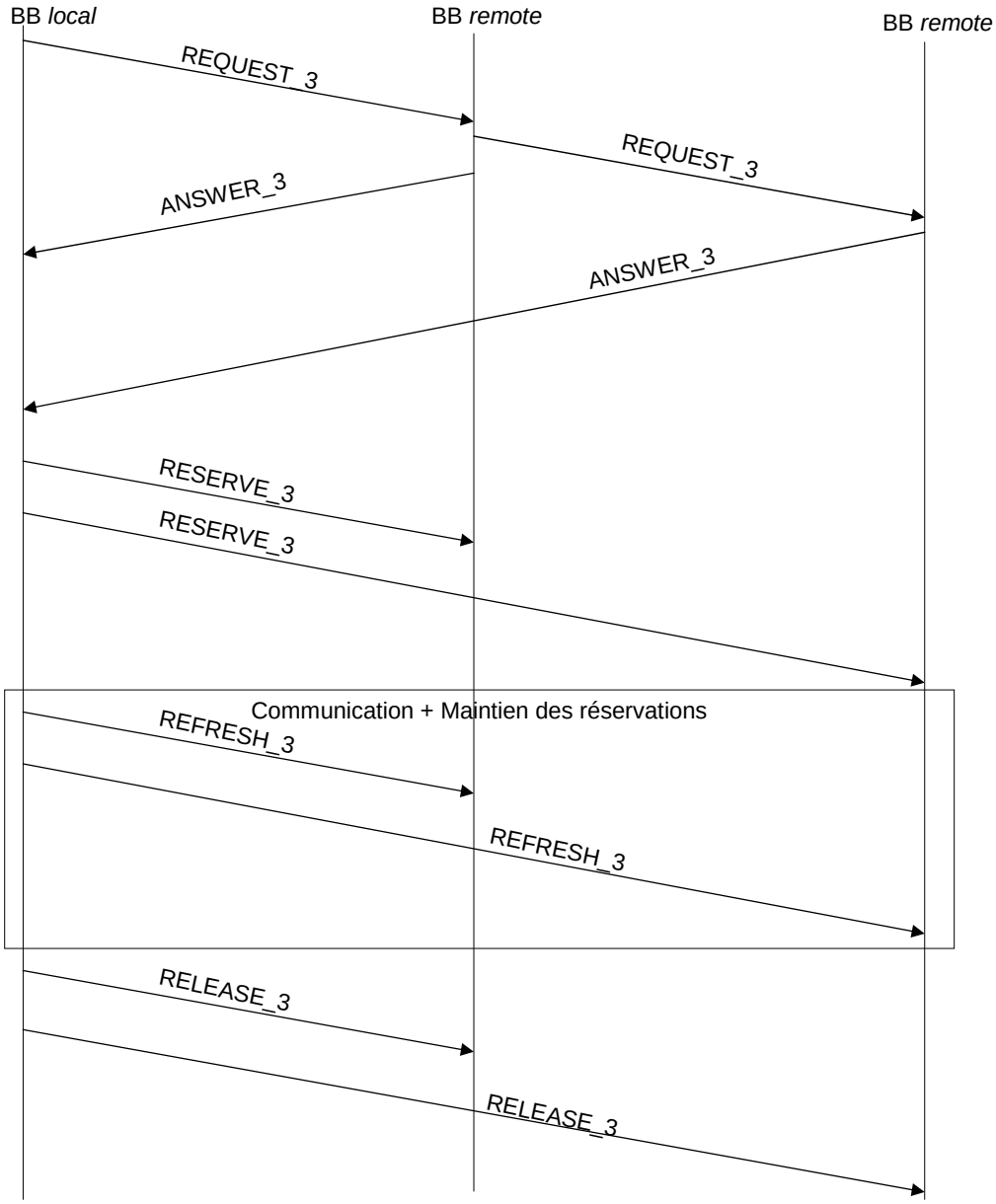


Figure 61 : Protocole de signalisation inter BB

Le **<u>protocole de configuration du routeur de bordure</u>** permet au BB *local* (via un PDU RESERVE_4) de configurer le routeur de bordure du domaine émetteur afin qu'il mette en place les mécanismes de classification, de *policing*, et de marquage adéquats pour un canal donné. L'état associé est maintenu par un PDU de rafraîchissement (PDU REFRESH_4) émis par le BB *local*.

Le PDU RELEASE_4 permet de relâcher les ressources.

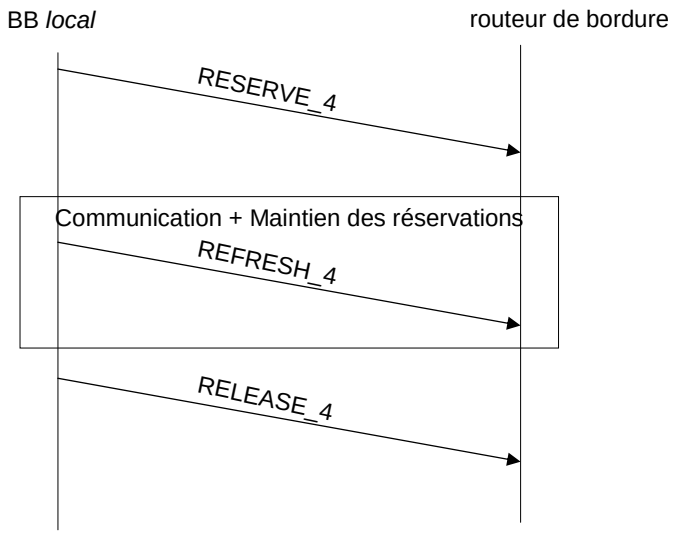


Figure 62 : Protocole de configuration du routeur de bordure

1.3 Enchaînement

L'enchaînement des quatre protocoles, voir Figure 63, est le suivant :

- l'hôte récepteur réclame la création d'un canal auprès de l'hôte émetteur en précisant les paramètres de QdS, via le protocole de création d'un canal ;
- sur réception de cette requête, l'hôte émetteur demande à son BB *local* les protocoles de Transport et classe de service IP qu'il doit utiliser pour satisfaire (si possible) la requête, via le protocole de gestion de la QdS ;
- le BB *local* propage cette requête aux BB *remote* impliqués, via le protocole de signalisation inter BB (les détails concernant le protocole de routage qui permet à un BB de connaître l'adresse du prochain BB à contacter sont donnés dans le chapitre suivant) ;
- lorsque tous les BB *remote* ont répondu au BB *local*[31], via le protocole de signalisation, le BB *local* répond à l'hôte émetteur, via le protocole de gestion de la QdS, et configure son routeur de bordure, via le protocole de configuration du routeur de bordure.
- sur réception de cette réponse, l'hôte émetteur répond à l'hôte récepteur via le protocole de création d'un canal.

[31] La description des PDU montrera que nous avons mis en place un mécanisme qui permet au BB *local* de savoir si tous les BB *remote* ont répondu.

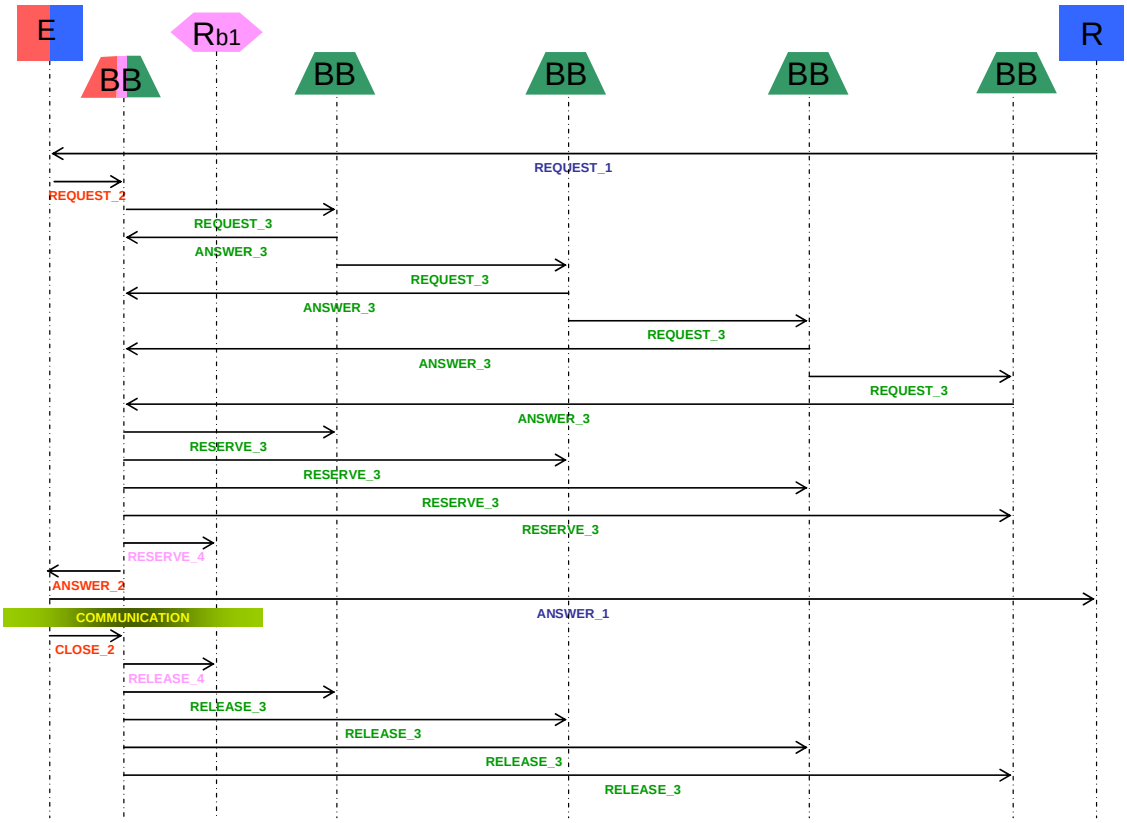


Figure 63 : Scénarios d'enchaînement des protocoles

1.4 Format des PDU

Pour chacun des protocoles, le format des PDU inclut un code à deux chiffres (premier octet) qui permet d'identifier le type de ces PDU (méthode utilisée également dans le protocole SIP) avec la convention suivante :

- le premier chiffre identifie le rôle du PDU, ainsi :
  - 0 pour une requête,
  - 1 pour une réponse,
  - 2 pour une fermeture de connexion,
  - 3 pour une réservation de ressources,
  - 4 pour une libération les ressources,
  - 5 pour un maintien des réservations de ressources ;
- le deuxième chiffre identifie le protocole d'appartenance, ainsi :
  - 1 signifie qu'il s'agit d'un PDU du protocole de création d'un canal,
  - 2 signifie qu'il s'agit d'un PDU du protocole de gestion de la QdS,
  - 3 signifie qu'il s'agit d'un PDU du protocole de signalisation inter BB,
  - 4 signifie qu'il s'agit d'un PDU du protocole de configuration du routeur de bordure.

1.4.1 Protocole de création d'un canal

Pour ce protocole, nous avons défini deux PDU :

- REQUEST_1 : pour la demande de connexion de l'hôte récepteur (du flux) à l'hôte émetteur ;
- ANSWER_1 : pour la réponse de l'hôte émetteur à l'hôte récepteur.

1.4.1.1 REQUEST_1

Ce PDU a une taille de 10 octets et contient trois types d'information :

- les paramètres de QdS incluant leurs sémantiques de garantie ;
- le numéro de port sur lequel l'entité réceptrice souhaite recevoir les données applicatives ;
- le profil de trafic du flux que l'application émettrice compte générer sur le canal considéré. Ce profil est ici exprimé sous la forme d'un débit moyen ; ultérieurement, nous prévoyons d'intégrer des profils de trafic plus complexes (*Token Bucket*, TSPEC de RSVP, …), au moyen du champ non utilisé.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0				1				τ_r								τ_d								σ_r				σ_d			
b																Reception Port															
Bandwidth																Non utilisé															

1.4.1.2 ANSWER_1

Ce PDU a une taille de 5 octets et contient quatre informations :

- la réponse de l'émetteur. Elle est codée sur un octet de façon à permettre en cas d'échec d'indiquer le motif de cet échec (QdS émetteur et récepteur incompatibles, ressources indisponibles, …) ;
- le protocole de transport à mettre en œuvre pour la réception des données si la requête est acceptée ;
- le nombre de retransmissions pour le cas où le protocole est FPTP ;
- le numéro de port que l'émetteur utilise pour envoyer les données.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0				1				Answer								Transport Protocol								Retransmission Number							
SenderDataPort																Non utilisé															

1.4.2 Protocole de gestion de la QdS

Pour ce protocole, nous avons défini quatre PDU :

- REQUEST_2 : pour la demande de connexion de l'émetteur à son BB *local* ;

- ANSWER_2 : pour la réponse du BB *local* à l'émetteur ;
- REFRESH_2 : pour maintenir une réservation sur le BB *local* ;
- CLOSE_2 : pour relâcher la réservation associée à un canal donné sur le BB *local*.

1.4.2.1 REQUEST_2

Ce PDU a une taille de 16 octets et contient, outre son type, onze champs principaux :

- un triplet identifiant le flux avec l'adresse IP de l'hôte récepteur et les numéros de port utilisés sur les hôtes émetteur et récepteur pour le canal.

  Note : l'adresse IP de l'hôte émetteur est accessible via l'en-tête du paquet IP véhiculant (via TCP) le REQUEST_2 (ce qui sera également le cas pour les PDU REFRESH_2 et CLOSE_2 décrits ci après). En revanche, le numéro de port source du paquet TCP encapsulant le REQUEST_2 est relatif à la connexion de signalisation (et non au canal à établir), ce qui justifie la présence dans le REQUEST_2 du numéros de port utilisé sur l'hôte récepteur pour le canal.
- un identifiant de flux réduit dont l'unicité est assurée par l'émetteur et qui sera utilisé par la suite par les autres PDU (pour un gain de taille des PDU) ;
- les paramètres de QdS, à savoir τ_r, σ_r, τ_d, σ_d, b ;
- le profil de trafic (même sémantique que pour le REQUEST_1).

0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31
0	2	Receiver Address	
Receiver Address	Receiver Port		StreamId_Sender
StreamId_Sender	τ_r	τ_d	σ_r σ_d
b		Bandwidth	

1.4.2.2 ANSWER_2

Ce PDU a une taille de 7 octets et contient, outre son type, cinq champs principaux :

- l'identifiant de flux réduit (l'adresse de l'émetteur est accessible via l'en-tête du paquet IP).
- la réponse (même sémantique que pour ANSWER_1)
- sur trois champs : le protocole de Transport, (ainsi que le nombre de retransmissions s'il s'agit de FPTP) et la classe de service IP permettant de satisfaire la requête.

0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31
1	2	StreamId_Sender	Answer
Transport Protocol (UDP,TCP,FPTP(n))	Retransmission Number (if FPTP)	IP Service Class (AS, GS, BE)	Non utilisé

1.4.2.3 REFRESH_2

Ce PDU a une taille de 3 octets. Il permet de maintenir l'état de réservation des ressources sur le BB *local* et contient, outre son type :

– l'identifiant de flux réduit (l'adresse de l'émetteur est accessible via l'en-tête du paquet IP).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
5				2				StreamId_Sender																Non utilisé							

1.4.2.4 CLOSE_2

Ce PDU a une taille de 3 octets. Il permet de réclamer le relâchement de la réservation associée à un canal et contient, outre son type:

– l'identifiant de flux réduit (l'adresse de l'émetteur est accessible via l'en-tête du paquet IP).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2				2				StreamId_Sender																Non utilisé							

1.4.3 Protocole de signalisation inter BB

Pour ce protocole, nous avons défini cinq PDU :

– REQUEST_3 : pour la demande de caractérisation des services et de leur état de disponibilité ainsi que la demande de pré réservation effectuée par le BB *local* au prochain BB *remote* ;
– ANSWER_3 : pour la réponse d'un BB *remote* au BB local ;
– RESERVE_3 : pour la réservation de ressources par le BB *local* sur un BB *remote* ;
– REFRESH_3 : pour le maintien de la réservation sur un BB *remote* (PDU émis par le BB *local*) ;
– RELEASE_3 : pour le relâchement des ressources sur un BB *remote* (PDU émis par le BB *local*).

1.4.3.1 REQUEST_3

Ce PDU a une taille de 14 octets et contient, outre son type :

– un numéro de séquence (1 octet) qui sera réutilisé pour la réponse du BB *remote* et qui permet au BB *local* de gérer les déséquencements entre les futures réponses des BB *remote* ;
– un numéro identifiant le flux (2 octets) dont l'unicité est garantie pas le BB *local* et qui sera utilisé par les autres PDU (gain de taille) ;
– le profil de trafic (même sémantique que pour ANSWER_1) ;
– l'adresse du BB local pour que tous les BB *remote* puissent répondre directement au premier BB ;
– l'adresse du récepteur pour que les BB *remote* puissent rechercher la route du destinataire.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0				3				Sequence Number								StreamId_BBLocal															
BB Local Address																															
Receiver Address																															
Bandwidth																Non utilisé															

1.4.3.2 ANSWER_3

Ce PDU a une taille de 35 octets et contient, outre son type, cinq types de champs :

- le numéro identifiant le flux (2 octets) dont l'unicité est garantie par le BB *local* ;
- le numéro de séquence (1 octet) ;
- les caractéristiques (30 octets) des services AS et GS données par cinq points représentatifs de la fonction de répartition (voir Chapitre 3) ; chacun de ces points est défini par un couple abscisse/ordonnée correspondant au délai (2 octets) et au pourcentage de paquets reçus dans ce délai (1 octet) ;
- la position (champ *LAST*) du BB *remote* dans la communication, pour savoir si le BB *remote* qui répond est le dernier à être contacté (ce qui revient à savoir si le récepteur est dans le même AS).
- la disponibilité des services.

Note : le champ marqué d'une astérisque (1er bit du 35ème octet) correspond au champ *LAST* qui prend la valeur 1 si le récepteur appartient au même domaine que le BB *remote*, 0 sinon.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1				3				StreamId_BBLocal																Sequence Number							
AS (dT point 1)																AS (% point 1)								AS (dT point 2)							
AS (dT point 2)								AS (% point 2)								AS (dT point 3)															
AS (% point 3)								AS (dT point 4)																AS (% point 4)							
AS (dT point 5)																AS (% point 5)								GS (dT point 1)							
GS (dT point 1)								GS (% point 1)								GS (dT point 2)															
GS (% point 2)								GS (dT point 3)																GS (% point 3)							
GS (dT point 4)																GS (% point 4)								GS (dT point 5)							
GS (dT point 5)								GS (% point 5)								*		AS=OK/KO				GS=OK/KO		Non utilisé							

1.4.3.3 RESERVE_3

Ce PDU a une taille de 6 octets et contient, outre son type, trois champs principaux:

- le numéro de flux réduit (1 octet), l'adresse du BB local étant extraite de l'en-tête du paquet IP ;
- la classe de service à réserver ;

– le débit demandé (2 octets) pour la réservation ; ce débit est ici exprimé sous la forme d'un débit moyen, son expression pouvant être étendue par la suite.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
3				3				StreamId_BBLocal																IP Service Class							
Bandwidth																Non utilisé															

1.4.3.4 RELEASE_3

Ce PDU a une taille de 3 octets et contient, outre son type, un champ principal :

– le numéro de flux réduit (1 octet), l'adresse du BB local étant extraite de l'en-tête du paquet IP.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
4				3				StreamId_BBLocal																Non utilisé							

1.4.3.5 REFRESH_3

Ce PDU a une taille de 7 octets et contient, outre son type, un champ principal :

– le numéro de flux réduit (1 octet), l'adresse du BB local étant extraite de l'en-tête du paquet IP.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
5				3				StreamId_BBLocal																Non utilisé							

1.4.4 Protocole de configuration du routeur de bordure

Pour ce protocole, nous avons défini trois PDU :

– RESERVE_4 : émis par le BB *local* à destination de son routeur de bordure pour configurer ce dernier. Cette configuration consiste en l'enregistrement par le routeur de bordure d'un état identifiant le flux sur lequel porte la réservation, son profil de trafic annoncé et le champs DSCP à attribuer aux paquets IP. C'est sur la base de cet état que le routeur de bordure applique ses mécanismes de classification, de *policing/shaping* et de marquage ;
– REFRESH_4 : émis par le BB *local* à destination de son routeur de bordure pour maintenir l'état identifiant le flux ;
– RELEASE_4 : émis par le BB *local* à destination de son routeur de bordure pour relâcher l'état identifiant le flux sur le routeur de bordure

Nous allons à présent voir en détail le format des PDU.

1.4.4.1 RESERVE_4

Ce PDU a une taille de 18 octets et contient, outre son type, cinq champs principaux :

- un quadruplet identifiant le flux : adresses IP et numéros de port des hôtes émetteur et récepteur ;
- le numéro de flux réduit (1 octet) ;
- la classe de service IP ;
- le débit demandé (2 octets) pour la réservation (ce débit est ici exprimé sous la forme d'un débit moyen, son expression pouvant être étendue par la suite).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
3				4				Sender Address																							
Sender Address								Sender Port																Receiver Address							
Receiver Address																								Receiver Port							
Receiver Port								StreamId_BBLocal																IP Service Class							
Bandwidth																Non utilisé															

1.4.4.2 RELEASE_4

Ce PDU a une taille de 3 octets et contient, outre son type, un champ principal :

- le numéro de flux réduit (1 octet), l'adresse du BB local est récupérée de l'en-tête des paquets IP.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
4				4				StreamId_BBLocal																Non utilisé							

1.4.4.3 REFRESH_4

Ce PDU a une taille de 3 octets et contient, outre son type, un champ principal:

- le numéro de séquence (1 octet), l'adresse du BB local est récupérée de l'en-tête des paquets IP.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
5				4				StreamId_BBLocal																Non utilisé							

2 Spécification de l'architecture en UML

Dans ce paragraphe, nous présentons la spécification UML 2.0 de l'architecture déployée sur chacun des équipements impliqués : hôte (émetteur ou récepteur), BB (*local* et *remote*) et routeur de bordure du domaine émetteur.

Cette spécification a été réalisée au moyen l'outil TAU de la société Télélogic que nous introduisons dans une première partie.

2.1 Présentation de UML / Tau

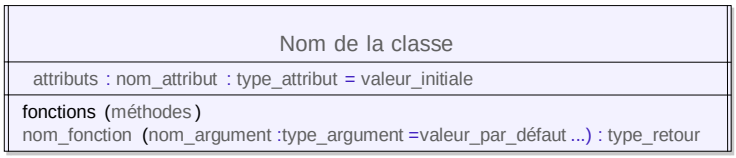
2.1.1 UML

Unified Modeling Language, dont le développement a commencé en 1994, est un outil de modélisation objet dérivé de méthodes comme Booch ou OMT (*Object Modeling Technique*). Il s'agit d'une notation standardisée qui facilite la conception de programmes, ainsi que leur description pour des non informaticiens. Ce mode de conception repose donc sur les principes de la programmation objet : manipuler des classes d'entités, classes constituées d'attributs (des variables) et de méthodes (des fonctions utilisant les attributs). Les classes définissent un type d'objet, l'objet proprement dit étant une instance de la classe correspondante dont l'état est donné par les valeurs instantanées de ses attributs. Les classes, ou types d'objet, sont en relation les unes avec les autres. Il peut s'agir de simples relations de dépendance (une classe a besoin d'une autre), ou de relations de hiérarchie (héritage). Définir une hiérarchie permet en particulier d'envisager certains objets comme l'agrégation de sous-objets. L'UML modélise les objets et leurs liens (en ce sens, les objets et leurs liens étant déjà des représentations, on peut parler de méta modèle) au moyen de vues constituées de diagrammes. On distingue les vues statiques, qui représentent "physiquement" le système à modéliser au moyen de diagrammes d'objets, de classes, de cas d'utilisation, de composants et de déploiement; et les vues dynamiques, qui montrent le fonctionnement du système au moyen de diagrammes de séquence, de collaboration, d'états transitions et d'activités. Au total, UML manipule donc 9 types de diagrammes.

Nous ne détaillerons pas ces 9 types de diagrammes mais nous rappelons uniquement la définition de ceux que nous utiliserons par la suite pour décrire notre architecture de signalisation.

2.1.1.1 Diagrammes d'objets et diagrammes de classes.

Une classe est représentée comme suit en UML :



Nous y retrouvons :

- le nom de la classe ;
- ses attributs ;
- ses méthodes.

2.1.1.2 Diagrammes de cas d'utilisation

UML permet, par des diagrammes de cas d'utilisation, de représenter comment le système modélisé se comporte du point de vue de l'utilisateur, à savoir un acteur externe. On décrit ainsi la

participation d'un acteur à un cas d'utilisation, lequel regroupe plusieurs scénarios d'utilisation du système. Notons que le terme "cas d'utilisation" (en anglais *use case*) désigne en fait un scénario d'utilisation (par exemple un acteur "client" va "consulter l'état d'une commande") et, par abus de langage, l'ensemble des scénarios d'utilisation. La Figure 64 représente un exemple de diagramme d'utilisation. Nous y retrouvons :

- les paquetages, qui sont des regroupements destinés à clarifier la manière dont un modèle est subdivisé; ils peuvent contenir des sous-paquetages ;
- la "note", qui est un simple commentaire d'un élément du diagramme ;
- la notion de "stéréotype": un stéréotype qualifie un certain type d'information; par exemple, si une note doit être considérée comme un élément du modèle, et non un simple commentaire (en d'autres termes, si la note véhicule du contenu dans la sémantique du modèle), elle est une "contrainte". Un stéréotype est noté entre les caractères "<<" et ">>" ;
- un acteur (il peut y en avoir plusieurs) ;
- un système (ce qui est modélisé) ;
- des cas d'utilisation qui peuvent avoir des relations entre eux de types :
  - *includes* : si un cas utilise un autre cas ;
  - *extends* : si un cas étend un autre cas.

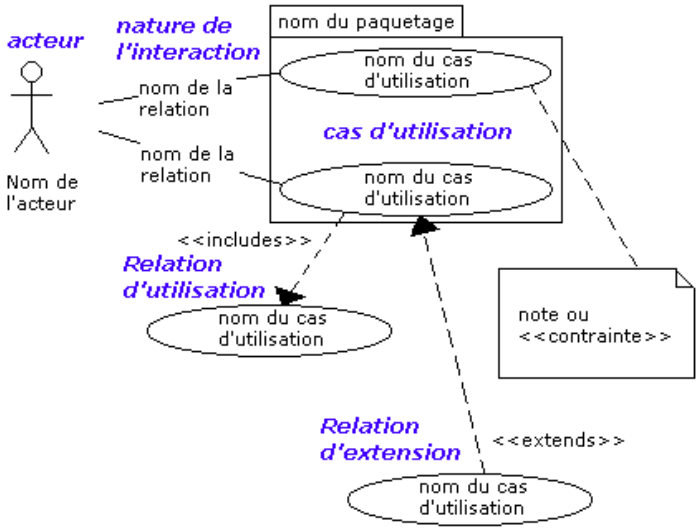


Figure 64 : Diagramme de cas d'utilisation

2.1.1.3 Diagrammes d'activités et d'états transitions

Ces diagrammes décrivent des changements d'état, d'un objet ou d'un composant. Un état est défini par les valeurs (instantanées) des attributs de l'objet ou des objets considérés. Une transition peut être déclenchée par un événement, ou au contraire être automatique. Dans le premier cas, la transition est désignée par l'événement qui la déclenche. Une transition peut également être conditionnelle.

Par exemple, examinons le diagramme de la Figure 65.

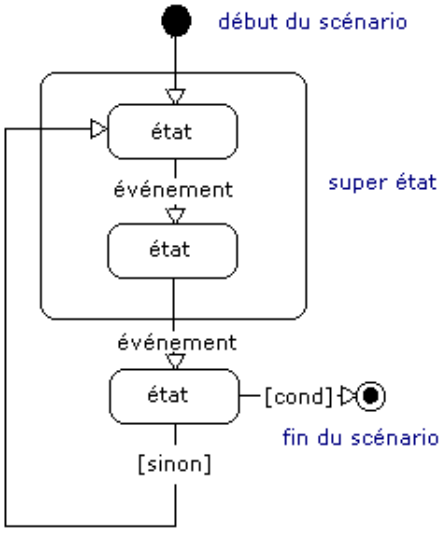


Figure 65 : Diagramme d'états

On considère un scénario, qui a donc un début et une fin, et différents états des objets du modèle lors du déroulement de ce scénario. Celui-ci peut lui-même être déclenché par un événement, lequel est donc externe au scénario: la transition (symbolique) vers le premier état est donc modélisée comme une transition automatique.

Dans notre exemple, le deuxième état est atteint à partir du premier suite à un événement. Ces deux états sont liés ici au sein d'un super état. Un troisième état, auquel on accède suite à un autre événement à partir du deuxième état, est soumis à une transition conditionnelle: un événement particulier, et un seul, provoque la fin du scénario : tout autre ramène à l'état de départ.

Par ailleurs, des groupes d'états transitions simultanés peuvent être représentés sur un même diagramme, en utilisant des automates à agrégation d'états (qui englobent au sein d'un super état les séries d'états transitions simultanées séparées par des lignes en pointillés).

Le passage d'un état à un autre se fait lorsque le système reçoit un signal qui peut être la réception d'un message, la fin d'un *timer*, …

Les symboles les plus courants sont représentés dans la Figure 66.

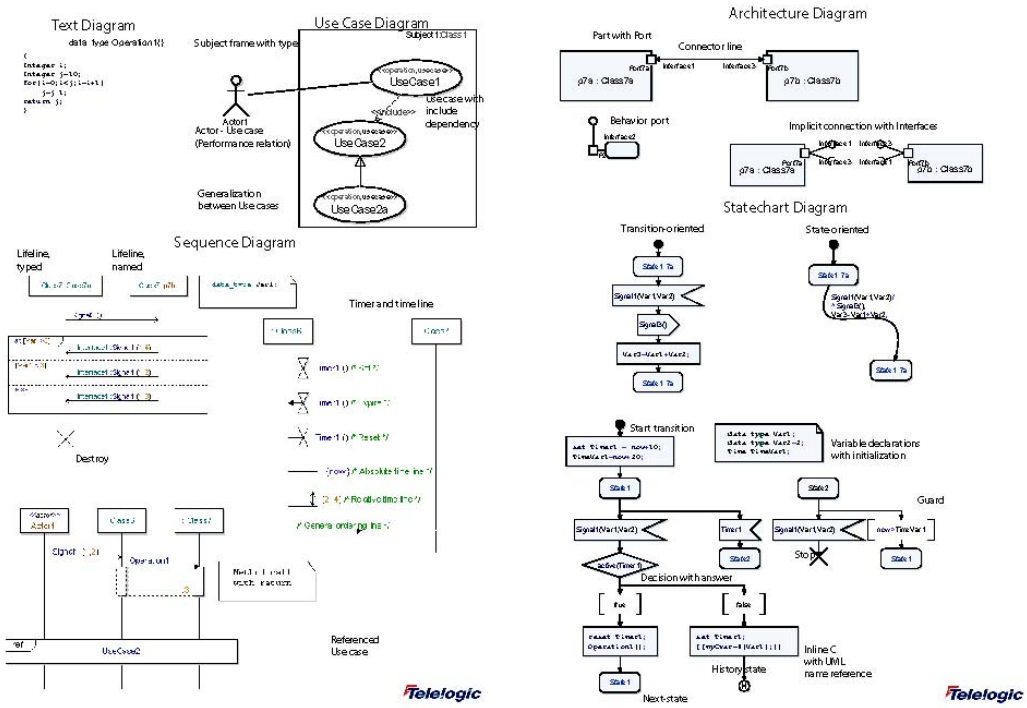


Figure 66 : Principaux symbole employés en UML (source TAU Télélogic)

De plus amples détails sur l'utilisation d'UML peuvent se trouver dans l'ouvrage très complet de (Doldi, 2003).

2.1.2 Tau

Télélogic Tau Generation2 est une suite intégrée de logiciels qui permet d'éditer, de simuler, de générer du code (C et bientôt Java) et basée sur des langages standard, pour le développement d'applications complexes, de systèmes d'information ou de systèmes industriels embarqués.

Fondée sur la norme UML 2.0, elle permet de modéliser des systèmes de grande taille et complexes, en utilisant des notations graphiques adoptées par l'industrie, et de valider les spécifications très tôt dans le processus de développement, la simulation dynamique se faisant sans avoir à générer de code. Les erreurs sont ainsi corrigées en amont du projet, ce qui permet de gagner du temps dans la phase d'implémentation d'un système.

Par contre, Tau ne permet pas de faire une validation formelle des systèmes conçus, les scénarios de simulations pouvant toutefois tester les éventuels risques de blocage.

2.2 Spécification de l’architecture au niveau d’un hôte

2.2.1 Principe

Ce paragraphe présente la spécification de l’architecture déployée au niveau des hôtes (émetteur et récepteur).

Dans un premier temps, la Figure 67 présente le diagramme de classe modélisant l’architecture sur les hôtes émetteur et récepteur, en incluant les entités d’application. Nous y retrouvons les instances :

- d’application;
- d’API (voir note ci-dessous) ;
- de protocoles (désignés dans le diagramme par système de communication) via lesquels les PDU de signalisation et de données sont échangés (typiquement TCP/IP, UDP/IP ou FPTP/IP);
- d'un élément appelé « serveur API » (coté émetteur uniquement), que nous introduisons ci-après.

Concernant les instances d’API ; afin de ne pas surcharger les diagrammes, les instances d’API ne se limitent pas aux seules primitives de service mises à disposition des applications mais elles incluent également les entités de protocoles activés suite aux appels de primitives ; dans les descriptions qui suivent, nous utiliserons le terme « d’instance d’API » avec cette signification.

Le serveur API n’est pas un élément indispensable dans la spécification de l’architecture : il a été défini lors de la phase d’implémentation de l’architecture pour répondre à un besoin que nous précisons ci-dessous. Nous ne présentons pas dans les diagrammes qui suivent la spécification du serveur API.

Le rôle du serveur API est d’assurer du côté émetteur que le numéro de port qui va être utilisé pour la connexion (TCP) de signalisation (associée à un canal donné) entre hôtes émetteur et récepteur n’est pas déjà attribué.

Pour cela, il s’exécute sur l’hôte émetteur et met en œuvre un protocole de communication, que nous appellerons par la suite « protocole de pré-signalisation », entre lui même et : d’une part, l’instance d’API émettrice, d’autre part l’instance d’API réceptrice.

Ce protocole comporte quatre PDU :

- ADDPORT_0, envoyé par l’entité d’API émettrice au serveur, pour l’enregistrement d’un nouveau port de signalisation ;
- SERVERACK_0, envoyé par le serveur à l’entité d’API émettrice, pour la confirmation de l’enregistrement ;
- SEEKPORT_0, envoyé par l’entité d’API réceptrice au serveur, pour la consultation du numéro de port de signalisation ;
- GIVEPORT_0, envoyé par le serveur à l’entité d’API réceptrice, pour la réponse à la consultation.

Une fois lancé, le serveur API se met en attente (sur deux numéros de port connus) :

- d'une requête d'enregistrement d'une nouvelle connexion de signalisation (PDU ADDPORT_0) de la part de l'entité d'API émettrice précisant le numéro de port que le récepteur doit utiliser pour établir la connexion ; s'il est valide, l'enregistrement est alors acquitté par le serveur API (via un PDU SERVERACK_0) ;
- d'une requête de consultation pour un flux donné (PDU SEEKPORT_0) en provenance de l'entité d'API réceptrice, cette requête ne pouvant être satisfaite (réponse du serveur via un PDU GIVEPORT_0) que si l'entité d'API émettrice s'est préalablement enregistrée.

La suite de la description du protocole est donnée au travers de la description des diagrammes d'états des API émettrice et réceptrice.

Nous rappelons ici les règles d'enchaînement des primitives. L'utilisateur doit :

- ouvrir une session ;
- créer autant de canaux de communication qu'il a identifié de flux ;
- commencer la communication (envoi ou réception de données) ;
- fermer les canaux qu'il a ouverts ;
- fermer la session ouverte.

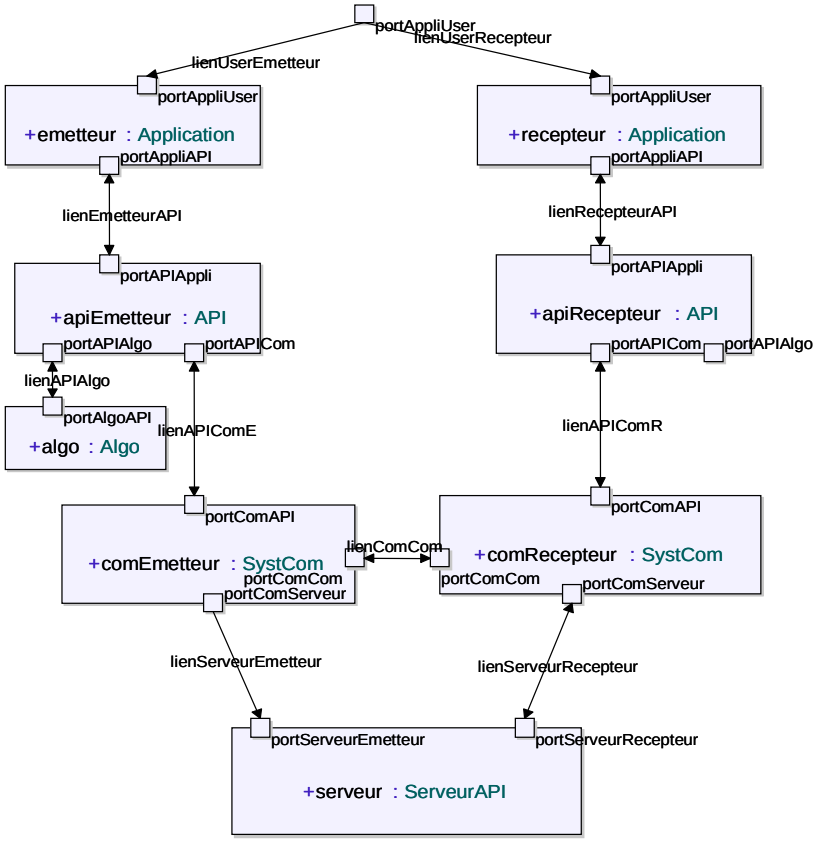


Figure 67 : Modélisation de l'architecture

Les diagrammes d'état (orientés transition) suivants illustrent le comportement de l'API.

Détaillons le diagramme de la Figure 68 illustrant la partie du comportement commune aux API émettrice et réceptrice.

Lorsque l'application invoque la primitive *openSession*, l'entité d'API (que nous appellerons dorénavant API émettrice) enregistre le nom de la session, puis se remet en attente d'un appel de primitive.

Lorsque l'application invoque la primitive *openChannel*, l'API vérifie que la session pour laquelle l'application souhaite ajouter un canal existe. Si elle n'existe pas, elle retourne un code d'erreur pour avertir l'application. Deux comportements sont alors possibles en fonction de la valeur du paramètre « rôle » (émetteur ou récepteur) de la primitive :

– rôle = émetteur : l'API (émettrice) récupère un numéro de port libre pour la signalisation puis le transmet (via le PDU ADDPORT_0) au serveur pour l'enregistrement de ce numéro. Il active alors un *timer* « retransmissiondelay3 » d'attente d'une demande de création de canal en provenance de l'API réceptrice (PDU REQUEST_1) sur le numéro de port libre, et se place dans l'état « WaitingForServerAck ». La suite du comportement est décrit dans le paragraphe (2.2.2) ;
– rôle = récepteur : l'API (réceptrice) envoie un PDU SEEKPORT_0 pour obtenir le numéro de port de signalisation qu'il doit utiliser par la suite. Il active alors un *timer* « retransmissiondelay1 » puis se met en attente de la réponse du serveur et se place dans l'état « WaitingForServerAnswer ». La suite du comportement est décrite dans le paragraphe (2.2.3).

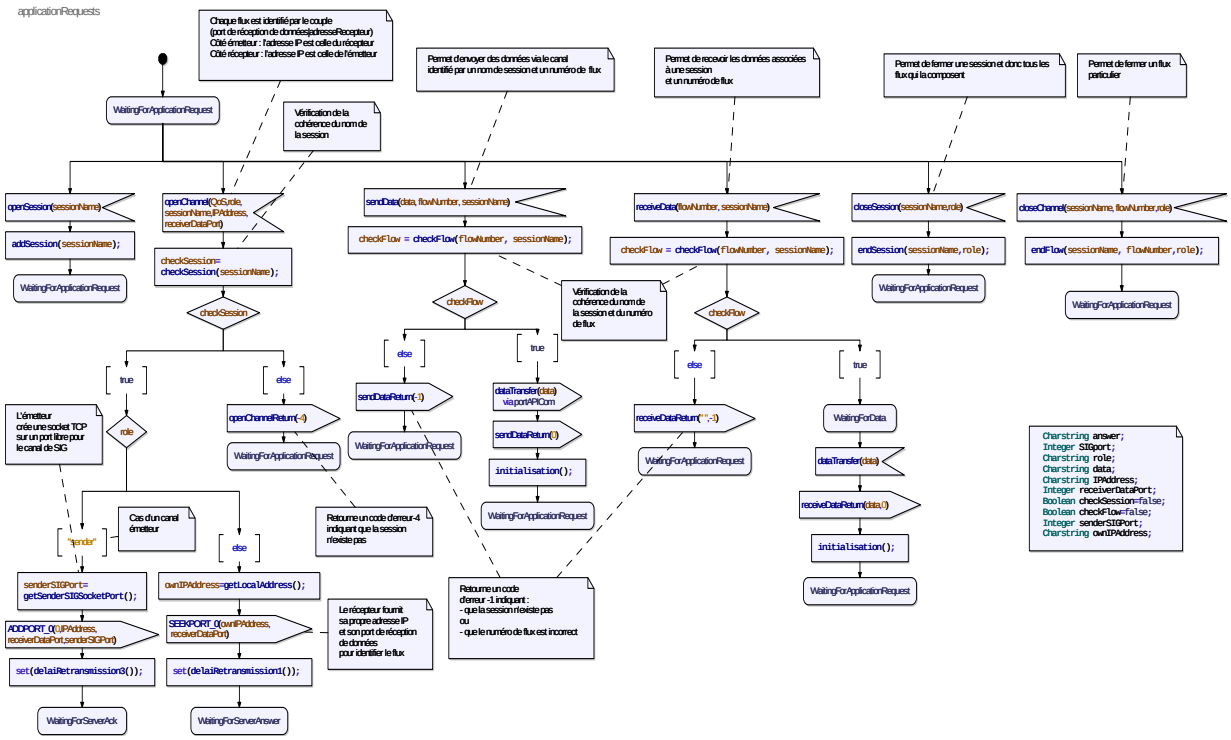


Figure 68 : Comportement de l'API

Lorsque l'application invoque la primitive *sendData* (resp. *receiveData*), l'API émettrice (resp. réceptrice) vérifie la cohérence entre le nom de la session et le numéro du canal. En cas d'incohérence, un code d'erreur est retourné à l'application ; sinon, la donnée est envoyée (resp. reçue).

Lorsque l'application invoque la primitive *closeSession* (resp. *closeChannel*), l'API ferme la session (resp. le canal) après vérification de l'existence de la session (resp. de l'appartenance du flux à une session existante).

2.2.2 Comportement de l'API émettrice

La Figure 69 illustre la suite de la spécification du comportement de l'API émettrice.

A partir de l'état « WaitingForServerAck », l'alternative est la suivante :

- le *timer* « delayretransmission3 » arrive à expiration, ce qui signifie que le serveur n'est pas encore démarré (le risque de perte du PDU est quasi nul vu que les deux entités du pré protocole de signalisation s'exécutent sur la même machine). Un mécanisme de retransmission du PDU ADDPORT_0 est activé, qui replace le système dans l'état « WaitingForServerAck ». Au bout de dix tentatives, l'API émettrice retourne un code d'erreur à l'application (en sortie de la primitive) ;
- une fois reçu, l'acquittement d'enregistrement du serveur (PDU SERVERACK_0), l'API émettrice se met en attente d'un PDU REQUEST_1 de la part de l'API réceptrice (dans le cadre du protocole d'établissement d'un canal). A la réception de ce PDU, l'API émettrice vérifie l'égalité entre les QdS requises côté émetteur et côté récepteur. Si l'égalité n'est pas vérifiée, l'API émettrice notifie l'échec de création du canal (via le PDU ANSWER_1 pour l'API réceptrice et un code erreur pour l'application). Sinon, il transmet au BB *local* une requête de création d'un canal (PDU REQUEST_2 du protocole de gestion de la QdS) et se met en attente de la réponse du BB local (PDU ANSWER_2). Deux cas de figure se présentent :
  - le BB local répond positivement à la requête. L'API émettrice retourne alors : (1) un PDU ANSWER_1 à l'API émettrice lui notifiant l'acceptation de la requête et lui indiquant le protocole de Transport à utiliser[32] et (2) un numéro de canal à l'application (en sortie de la primitive) qui sera utilisé ensuite pour l'appel aux autres primitives ;
  - le BB local répond négativement. L'API émettrice retourne alors : (1) un PDU ANSWER_1 notifiant l'échec à l'entité réceptrice et (2) un code d'erreur à l'application (en sortie de la primitive).

[32] Ainsi que le numéro de port qui sera utilisé par l'émetteur pour l'émission des données – nécessaire pour le protocole FPTP

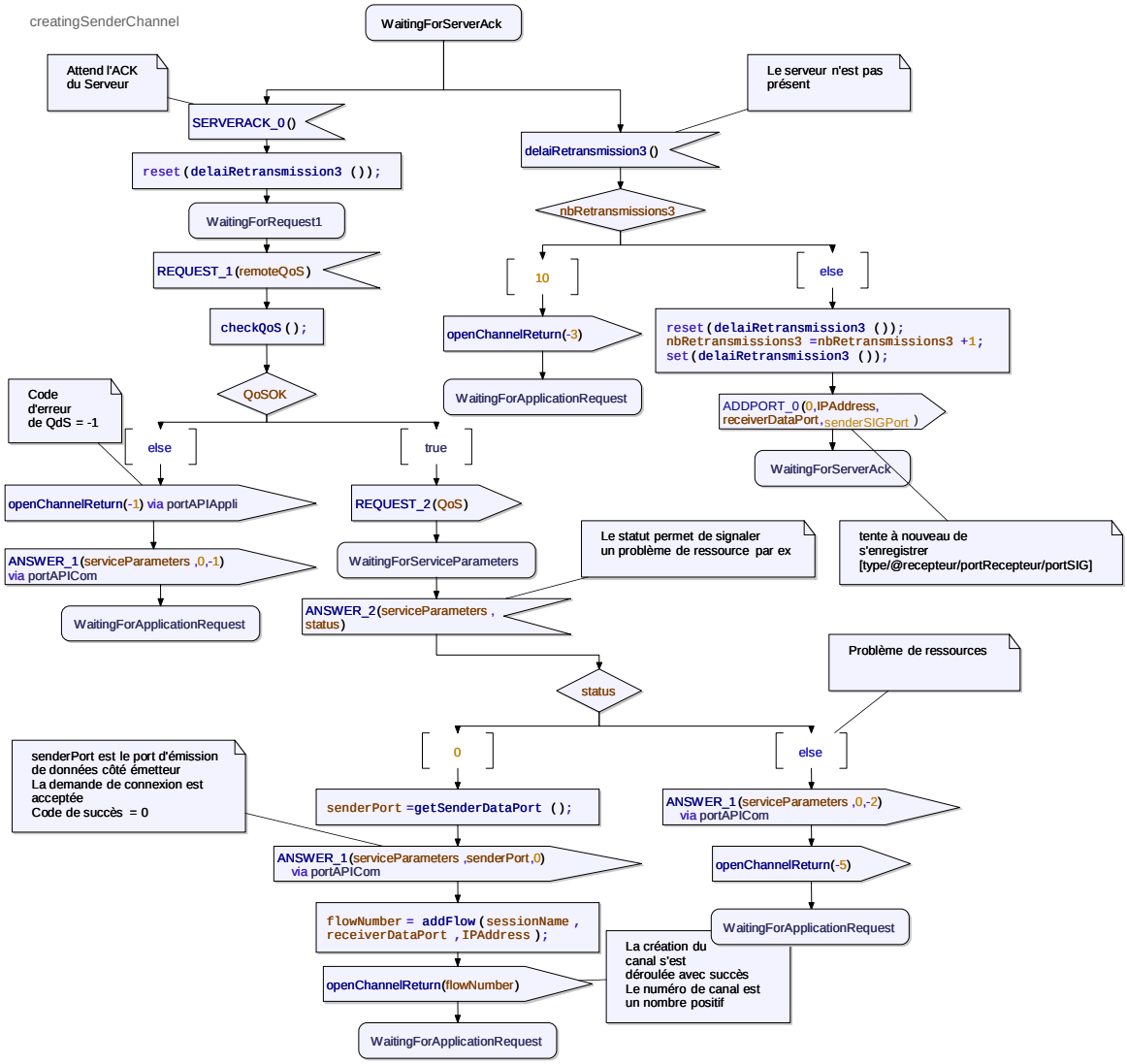


Figure 69 : Comportement de l'API émettrice (suite)

2.2.3 Comportement de l'API réceptrice

La Figure 70 illustre la suite de la spécification du comportement de l'API réceptrice

A partir de l'état « WaitingForServerAnswer», l'alternative est la suivante :

– le timer « delayretransmission1 » arrive à expiration. Un mécanisme de retransmission du PDU SEEKPORT_0 est activé, qui replace l'API réceptrice dans l'état « WaitingForServerAnswer ». Au bout de dix tentatives, l'API réceptrice retourne une code d'erreur à l'application (en sortie de la primitive) ;

– le serveur répond à la consultation (via le PDU GIVEPORT_0) en indiquant le numéro de port qui doit être utilisé par l'API réceptrice pour établir la connexion de signalisation. Deux cas de figure se présentent :
  - l'API réceptrice émet une requête de création de canal à l'API réceptrice via le PDU REQUEST_1, puis se place dans l'état « WaitingForAnswer1 » et se met en attente de la réponse ;
  - le serveur signale que l'émetteur ne s'est pas encore enregistré ; dans ce cas, soit (1) l'API réceptrice recommence la procédure d'enregistrement après avoir attendu un certain temps, soit (2) l'API réceptrice en est déjà à sa dixième tentative (p = 10), il renvoie alors un code d'erreur à l'application (en sortie de la primitive).

A partir de l'état « WaitingForAnswer1 », l'alternative est la suivante :

– l'API réceptrice reçoit un PDU ANSWER_1 notifiant l'acceptation de la requête, elle retourne alors un numéro de canal à l'application qui devra être utilisé pour les autres primitives, puis elle se replace en attente d'un appel à une primitive ;
– l'API réceptrice reçoit un PDU ANSWER_1 notifiant l'échec de la création du canal, elle renvoie alors un code d'erreur à l'application (en sortie de la primitive).

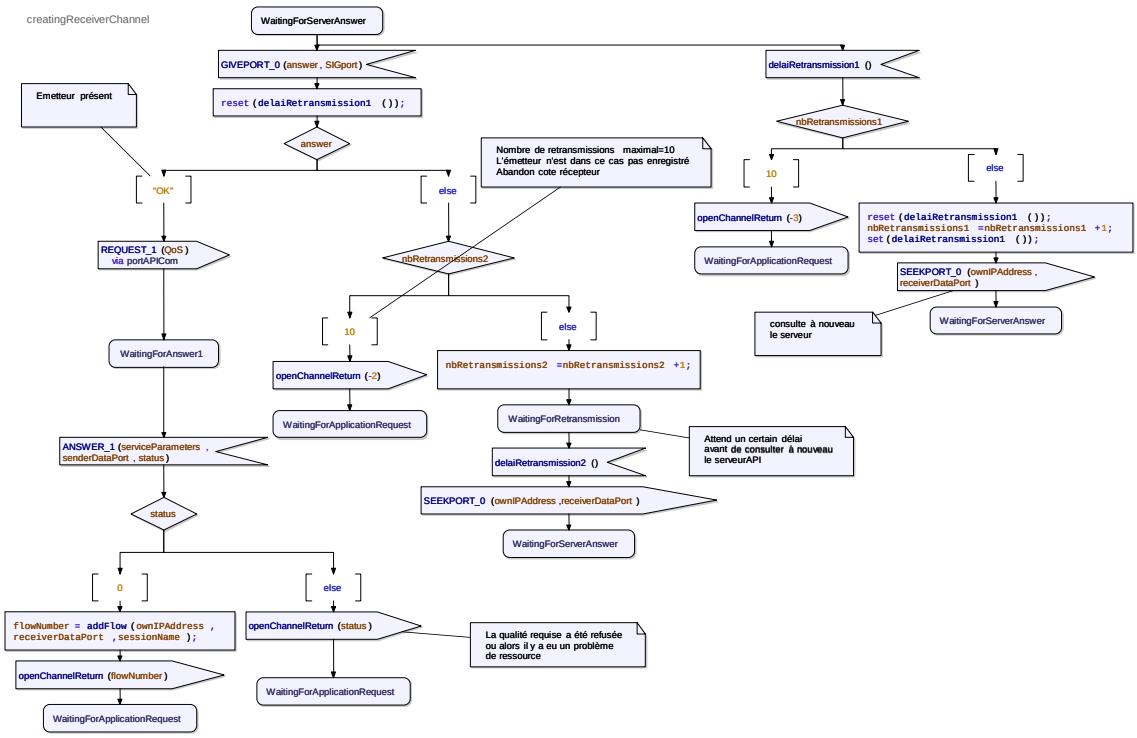


Figure 70 : Comportement de l'API réceptrice (suite)

2.3 Spécification de l'architecture au niveau d'un BB (*local* ou *remote*)

L'architecture de signalisation déployée au niveau d'un BB (*local* ou *remote*) implique trois protocoles :

- le protocole de gestion de la QdS ;
- le protocole de signalisation inter BB ;
- le protocole de configuration du routeur de bordure.

Nous représentons tout d'abord le diagramme d'état d'un BB en distinguant le comportement d'un BB *local* de celui d'un BB *remote*. Nous représentons ensuite les diagrammes de séquences associés à deux cas d'utilisation :

- traitement d'une requête de réservation de ressources ;
- traitement d'une requête de libération de ressources.

2.3.1 Diagrammes d'état des BB

Quelque soit son rôle, *local* ou *remote*, un BB déploie la même architecture de protocoles. C'est sur réception d'un PDU REQUEST_2 qu'il adopte le comportement d'un BB *local* et sur réception d'un PDU REQUEST_3 qu'il adopte celui d'un BB *remote*.

2.3.1.1 Comportement d'un BB *local*

Sur réception d'un PDU REQUEST_2, le BB *local* consulte une table de routage spécifique qui lui indique l'adresse du prochain BB *remote* à qui envoyer un PDU REQUEST_3. Il se met alors en attente des réponses de tous les BB *remote* impliqués dans la route entre hôtes émetteur et récepteur.

Sur réception du dernier PDU ANSWER_3 (attribut *Last = True*), il active l'algorithme de sélection des services décrit dans le Chapitre 3. Notons qu'un mécanisme de numérotation des PDU ANSWER_3 permet de contrôler qu'il n'y a pas eu de déséquencement dans les réponses des différents BB *remote*.

La requête de l'émetteur peut ne pas être satisfaite pour deux raisons majeures (hors problèmes du type panne d'un BB) :

- soit les ressources sont momentanément indisponibles car réservées par des flux déjà présents ;
- soit les ressources sont définitivement indisponibles car les paramètres de QdS ne peuvent être satisfaites au regard des capacités du système.

Dans les deux cas, le BB *local* libère les ressources pré-réservées sur les BB *remote* par émission d'un PDU RELEASE_3 et répond par le PDU ANSWER_2 à l'émetteur en précisant que la requête ne peut pas être satisfaite.

Dans le cas où les performances des services des différents AS autorisent l'acceptation de la requête, le BB *local* entérine les réservations des BB *remote* en leur envoyant un PDU RESERVE_3.

Il configure alors par un PDU RESERVE_4 le routeur de bordure du site émetteur.

Il retourne ensuite un PDU ANSWER_2 à l'hôte émetteur lui signalant que la requête a été acceptée puis il se met en attente de PDU REFRESH_2 périodiques qui permettent de maintenir la réservation. Dans le cas où ces PDU n'arrivent pas, les ressources sont relâchées automatiquement sur tous les équipements (BB *local* et *remote* ainsi que sur le routeur de bordure) ; ce mécanisme permet de ne pas conserver un état de réservation si le PDU CLOSE_2 de l'émetteur est perdu. Lorsque le BB *local* reçoit un PDU REFRESH_2, il transmet aux autres équipements un PDU REFRESH_3 (ou _4).

Lorsqu'un canal est relâché par un hôte émetteur, le BB *local* reçoit un PDU CLOSE_2 et transmet alors un PDU RELEASE_3 (ou _4) à chacun des autres équipements (BB *remote* et routeur de bordure) pour que ceux-ci libèrent leurs ressources.

La Figure 71 décrit une partie du comportement du BB *local* ; nous n'avons pas inclus dans cette figure les mécanismes des *timers* pour ne pas la surcharger.

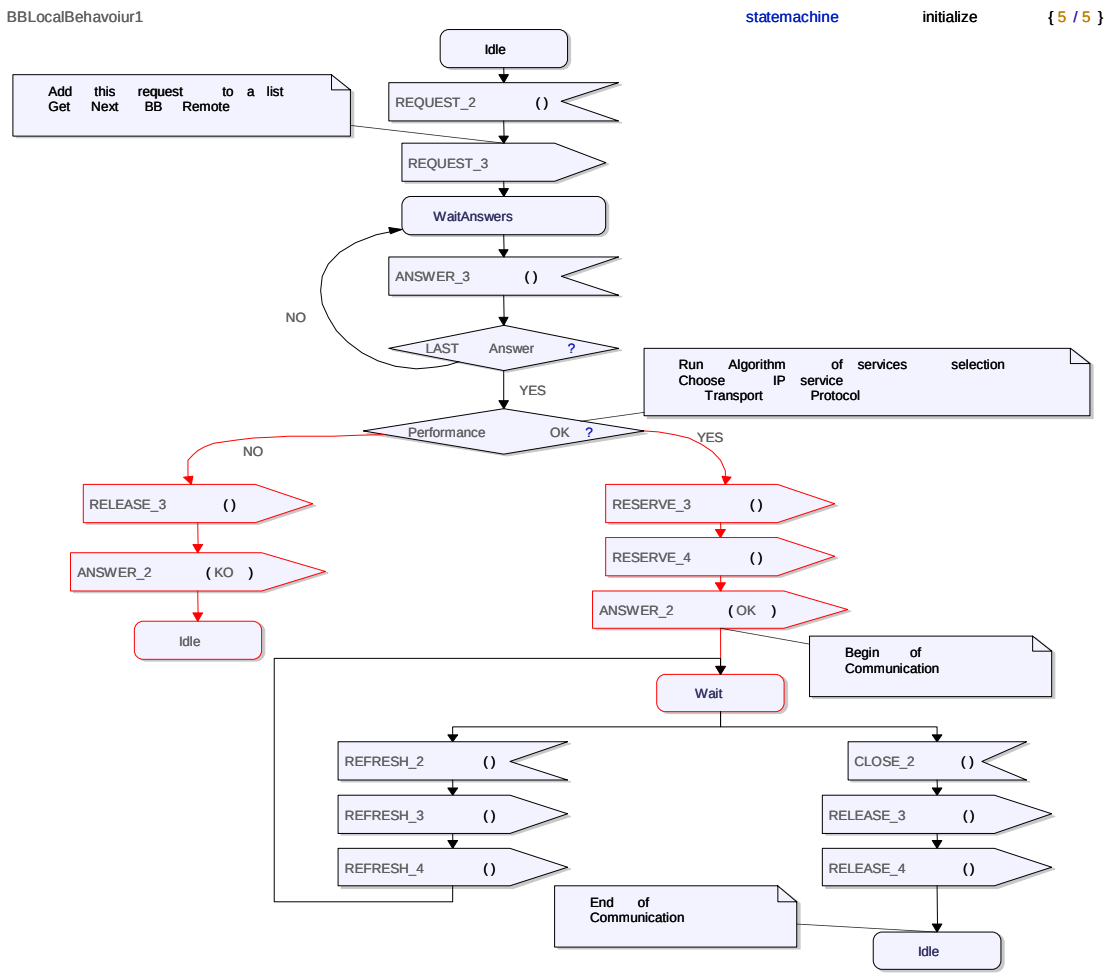


Figure 71 : Comportement d'un BB local

2.3.1.2 Comportement d'un BB *remote*

Lorsqu'un BB reçoit un PDU REQUEST_3, il adopte le comportement d'un BB *remote*.

Sur réception de ce PDU, il vérifie dans un premier temps que la bande passante requise (correspondant au profil de trafic annoncé pour le flux) est disponible dans au moins une des classes de service, et à défaut renvoie au BB *local* une réponse ANSWER_3 refusant la demande.

Dans le cas où les ressources sont compatibles avec la requête, le BB *remote* fait une pré-réservation sur l'ensemble des classes. Il y a alors deux possibilités :

- le BB *remote* est le dernier BB (le récepteur est directement accessible sans passer par un autre domaine) ; il renvoie alors au BB *local* une réponse ANSWER_3 indiquant qu'il est le dernier BB ;
- le récepteur ne fait pas partie de l'AS géré par ce BB ; la requête est alors transférée au prochain BB *remote* en même temps qu'il répond au BB *local*.

Dans ces deux cas, le BB *remote* se met en attente :

- soit d'un PDU RESERVE_3 qui lui indique d'entériner la pré-réservation pour la classe de service sélectionnée et de libérer celles des autres classes si besoin ;
- soit d'un PDU RELEASE_3 qui lui indique de relâcher toutes les pré-réservations, dans le cas où globalement (sur l'ensemble des réponses des BB) la requête ne peut pas être satisfaite.

Dans le cas où la requête est confirmée, le BB *remote* se met en attente d'un PDU périodique REFRESH_3 (pour maintenir l'état de réservation) ou d'un PDU RELEASE_3 lorsque la communication est terminée.

La Figure 72 montre le diagramme d'état d'un BB *remote*.

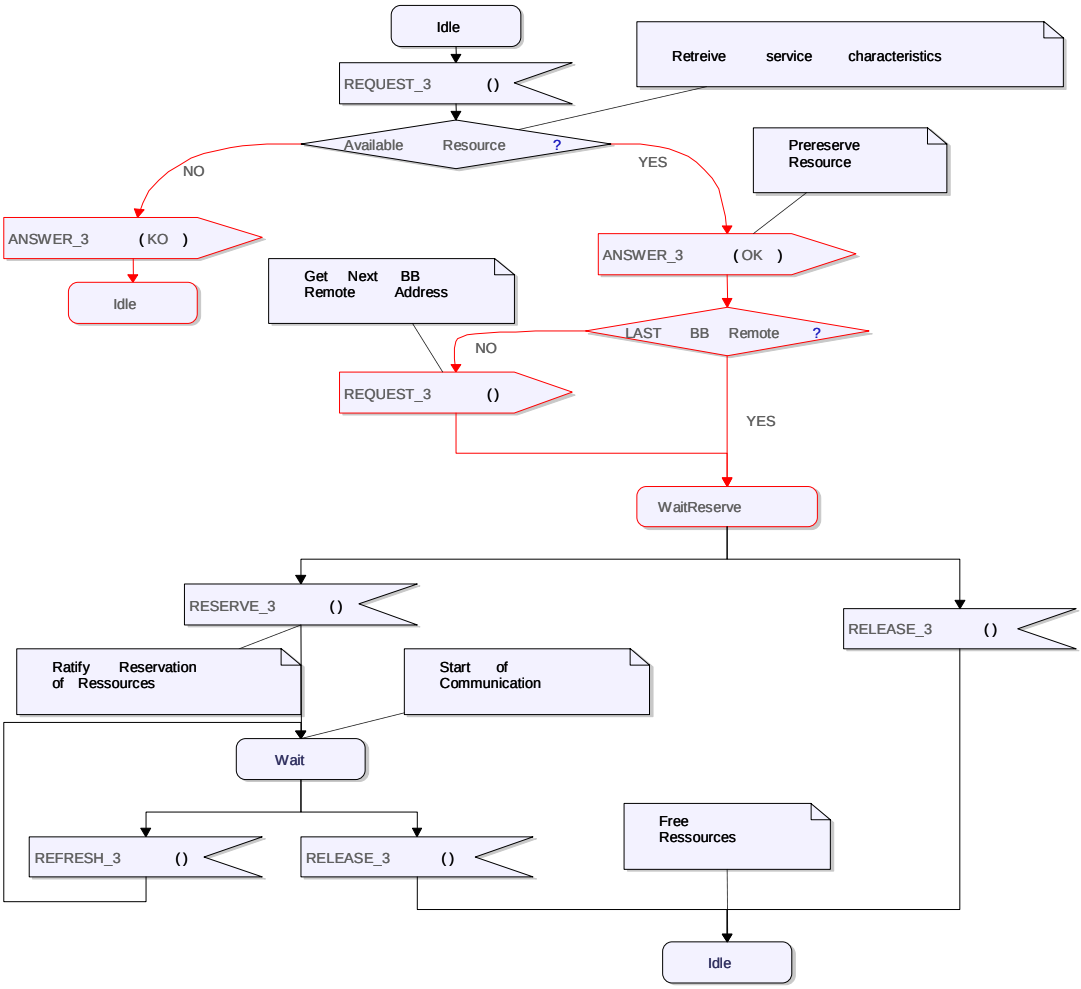


Figure 72 : Comportement d'un BB *remote*

2.3.2 Diagrammes de séquence

Les trois diagrammes de séquence fournis Figure 74, Figure 75 et Figure 76 illustrent le déroulement des messages pour les deux cas d'utilisation suivants (Figure 73) :

- traitement d'une requête de réservation de ressources (REQUEST_2) issue d'une demande de création d'un canal par une application ;
- traitement d'une requête de libération de ressources (CLOSE_2) issue d'une demande de fermeture d'un canal par une application.

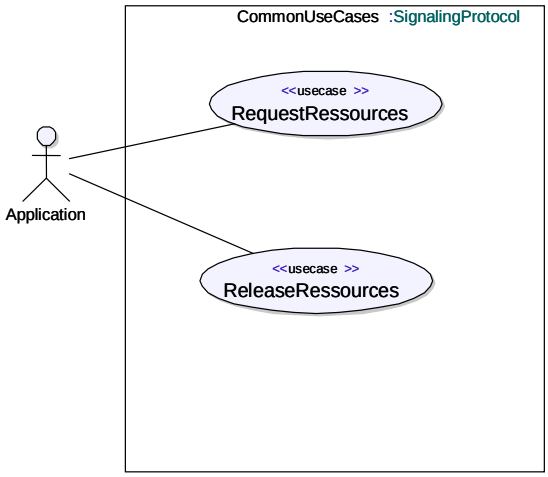


Figure 73 : diagramme d'utilisation

Dans les deux cas d'utilisation, nous avons considéré un scénario dans lequel il y a trois BB *remote*.

2.3.2.1 Cas d'une requête de réservation de ressources

Les diagrammes de séquence des Figure 74 et Figure 75 montrent le déroulement des messages lors de la requête de l'application émettrice. Nous y trouvons le déclenchement des *timers* que nous n'avions pas représentés dans les diagrammes d'état pour des raisons de place, mais que nous détaillons dans ce paragraphe.

Diagramme de séquence de la Figure 74

Lorsqu'un BB *remote* reçoit un PDU REQUEST_3, il déclenche un *timer* appelé PRE_RESA. Sur le diagramme, nous avons représenté l'alternative suivante (notée *alt* sur la figure), selon que les réponses des BB *remote* arrivent :

- avant l'expiration du *timer* du BB *local* : dans ce cas, celui-ci se met dans l'état « ANSWERS RECEIVED » ;
- après l'expiration du *timer* du BB *local* : dans ce cas, la requête est refusée et, sur chaque BB *remote*, l'expiration du *timer* PRE_RESA engendre l'annulation de toutes les pré-réservations.

Dans l'état « ANSWERS RECEIVED », trois scénarios sont possibles :

- PERFORMANCE = OK : le BB *local* envoie un PDU de réservation RESERVE_3 à chacun des BB *remote,* ce qui active un *timer* appelé TIMER_RESA et conduit le BL *local* à l'état « START COMMUNICATION » ;
- TIMER=EXPIRED : le *timer* PRE_RESA de l'un au moins des BB *remote* a expiré, ce qui engendre l'annulation du canal en cours et le relâchement des pré réservations ;

– PERFORMANCE=KO : les performances du système ne sont pas compatibles avec la requête de QdS : le BB *local* informe l'hôte émetteur et les ressources seront libérées sur les différents équipements.

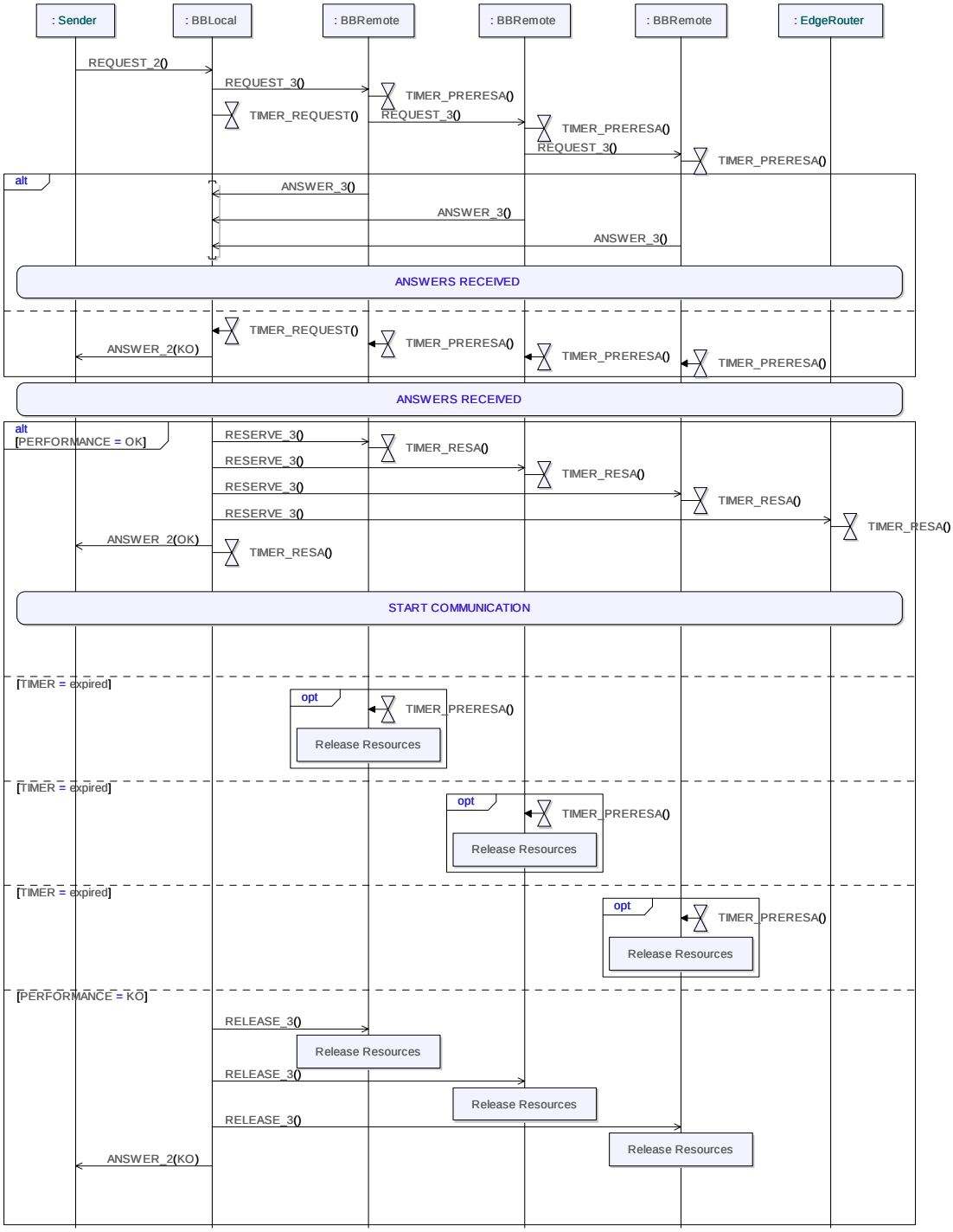


Figure 74 : Diagramme de séquence associé à une requête de réservation de ressources (partie I)

Figure 75

A partir de l'état « START COMMUNICATION », l'alternative suivante se présente :

– l'hôte émetteur envoie périodiquement un PDU REFRESH_2 au BB *local* qui le transmet (REFRESH_3) à tous les BB *remote,* ceux-ci réactivant alors le *timer* TIMER_RESA associé à la réservation. Si le *timer* TIMER_RESA d'un BB *remote* expire, ce dernier libère les ressources locales attribuées au flux correspondant ;
– l'hôte émetteur n'envoie pas de PDU REFRESH_2, le BB *local* libère alors toutes les ressources locales attribuées au flux correspondant et n'émet plus de PDU REFRESH_3 à destination des BB *remote*.

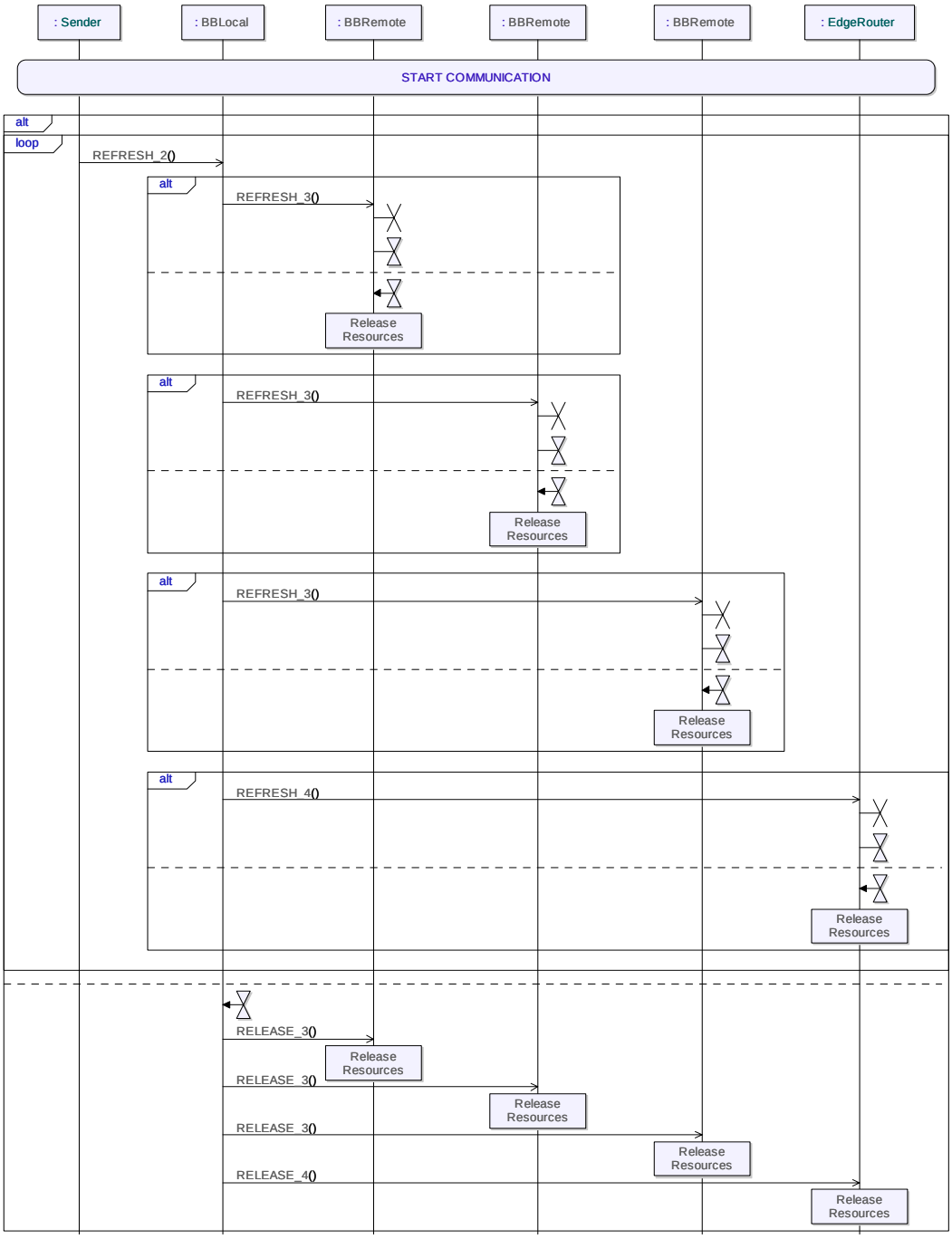


Figure 75 : Diagramme de séquence associé à une requête de réservation de ressources (partie II)

2.3.2.2 Cas d'une requête de libération de ressources

Le diagramme de séquence correspondant à la libération des ressources suite à la réception d'un PDU CLOSE_2 sur le BB *local* est simple. Le BB arrête son *timer* TIMER_RESA et transmet la requête aux autres équipements par l'envoi d'un PDU RELEASE_3 (ou 4).

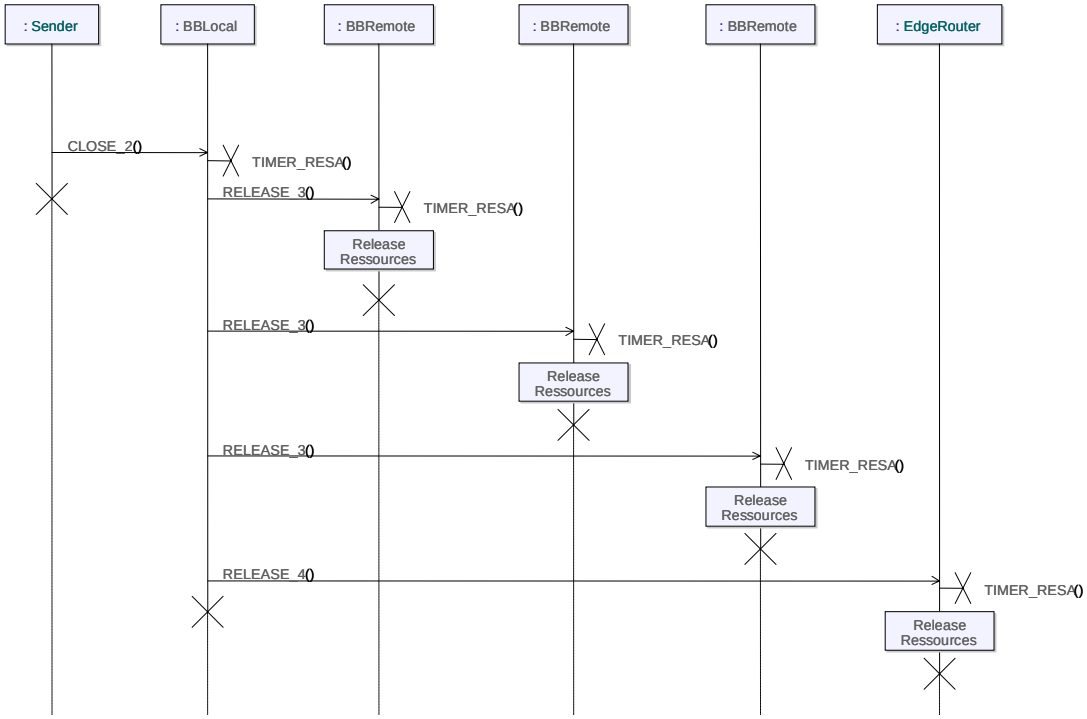


Figure 76 : Diagramme de séquence associé à une requête de libération de ressources

2.4 Spécification de l'architecture au niveau d'un routeur de bordure

Au niveau d'un routeur de bordure, la spécification de l'architecture est relativement simple. La Figure 77 représente cette spécification.

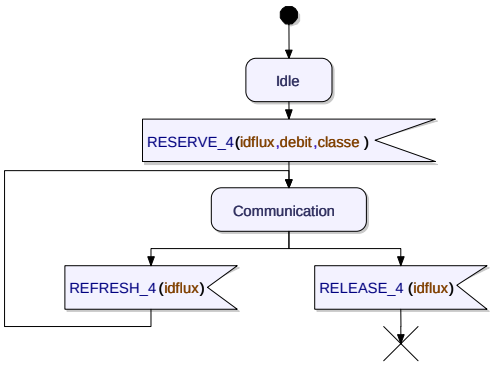


Figure 77 : Architecture au niveau d'un routeur de bordure

À la réception d'un PDU RESERVE_4, le routeur de bordure configure son interface d'entrée (pour le *policing, shaping*,… par flux), puis il se met en attente de PDU REFRESH_4. Dans le cas où

ce PDU n'arrive pas, un *timer* permet de relâcher les ressources. Le PDU RELEASE_4 permet de relâcher les ressources réservées pour un flux donné.

3 Conclusion

Dans ce chapitre, nous avons présenté l'architecture de signalisation de notre système de communication.

Nous avons défini les différents protocoles, leur localisation et les règles d'enchaînement ainsi que le format de leur PDU.

Nous avons spécifié en UML l'architecture mise en œuvre sur chacun des équipements du système, en représentant d'abord les diagrammes d'état des équipements concernés puis les diagrammes de séquence pour deux scénarios : une requête de réservation de ressources et une requête de libération de ressources.

De cette présentation, nous faisons ressortir trois limitations principales :

- le système de communication proposé, même s'il abstrait son utilisateur du choix et de la configuration des services Transport et IP sous-jacents, lui impose de fournir des paramètres de QdS dont la valeur n'est pas triviale à déterminer ; en outre, l'accès au système pour une application déjà existante apparaît peu évident : le chapitre 5 apportera des éléments de solution à ces deux limites ;
- l'architecture de signalisation proposée ne prend pas en compte le facteur de mise à l'échelle vis-à-vis du nombre de flux à QdS ;
- enfin, la solution proposée n'est valable que pour le modèle de l'Internet défini au chapitre 2 de ce mémoire.

Ces deux dernières limites seront discutées plus avant en conclusion générale de ce mémoire, et elles induiront les principales perspectives de nos travaux.

Chapitre 5 Tests du système de communication

1 Objectif

Nous distinguons trois objectifs dans ce chapitre :

- la validation expérimentale du système de communication proposé et, de façon plus précise, de son architecture de signalisation ;
- la définition de mécanismes permettant de faciliter la configuration des paramètres de QdS d'un canal de communication, en proposant des paramètres plus adaptés à des familles d'application ;
- la définition des modifications à faire sur une application multimédia existante pour qu'elle puisse accéder à notre système de communication.

La démarche envisagée est la suivante :

- vis-à-vis du premier objectif, il faut :
  - définir et configurer la plate-forme retenue pour les expérimentations, émulant à l'aide de *Dummynet* un environnement multi domaines ;
  - spécifier un certain nombre de scénarios de tests visant à valider :
    - une phase réussie d'établissement de session ;
    - les différents cas d'échec d'établissement d'un canal ;
    - le comportement du système vis-à-vis d'une rupture de réservation due à la perte de plusieurs PDU de maintien des réservations ;
    - la dualité du comportement (*local* et *remote*) d'un BB.
- vis-à-vis du deuxième objectif, elle consiste à définir, par famille d'applications, un module d'adaptation, permettant à l'application (via une API de plus haut niveau) ou à un utilisateur externe (par le biais d'une IHM), d'injecter des paramètres de QdS compréhensibles du point de vue de l'application (ou de l'utilisateur externe), que le système de communication se chargera de mettre en correspondance avec les paramètres de l'API ;
- vis-à-vis du troisième objectif, nous envisageons deux cas d'utilisation de l'API :
  - la modification de l'application pour qu'elle utilise l'API telle qu'elle a été définie ;
  - la modification de l'application pour qu'elle accède au système sans qu'elle ait conscience de l'API.

Le chapitre est structuré comme suit. Dans le paragraphe (2), nous présentons les applications multimédias utilisées pour tester le système de communication. Dans le paragraphe (3), nous donnons une description détaillée de la plate-forme émulant un environnement multi domaines DiffServ. Dans le paragraphe (4), nous présentons les scénarios expérimentaux réalisés et les résultats obtenus pour valider les choix d'implémentation du système de communication.

2 Intégration des applications tests

Dans ce paragraphe, nous présentons les deux applications retenues pour tester le déploiement du système de communication (2.1). Nous rappelons que l'objectif ici n'est pas d'évaluer les performances des applications mais d'étudier :

- les mécanismes pour simplifier le paramétrage des connexions (2.2), ce qui correspond au deuxième objectif du chapitre ;
- les modifications minimales des applications, nécessaires pour qu'elles accèdent à notre système (2.3), ce qui correspond au troisième objectif du chapitre.

2.1 Description des applications

2.1.1 Serveur vidéo

La première application est une application de diffusion de vidéo à la demande.

Cette application met en jeu deux entités, respectivement émettrice (serveur) et réceptrice (client) ; le récepteur spécifie divers paramètres tels que l'URL de la vidéo à télécharger, le port de réception des données, l'utilisation ou non de l'audio, et enfin le protocole de transport à utiliser (UDP ou TCP). Sitôt reçu, le média est présenté à l'aide d'un *player*, côté récepteur.

Sur le plan de sa conception, cette application, basée sur l'API JMF/RTP de Sun, a été développée initialement pour utiliser deux services de transport différents (fournis par TCP et UDP). Si RTP[33] est un protocole de niveau applicatif théoriquement indépendant du protocole de transport choisi, il est très souvent utilisé au-dessus de UDP, compte tenu des fortes contraintes temporelles des applications multimédias. Sun a toutefois prévu la possibilité d'étendre l'API JMF/RTP à d'autres protocoles de transport tels que TCP, SCTP, DCCP. Nous avons donc pu l'étendre au protocole FPTP. Le choix des concepteurs de cette application nous a permis de pouvoir utiliser cette possibilité lorsque nous avons porté l'application sur le système de communication.

[33] RTP et RTCP (Schulzrinne, 1999) sont deux protocoles offrant aux applications un service leur permettant de numéroter les paquets applicatifs et de les estampiller temporellement dans le but qu'elles puissent gérer plus facilement les problèmes de gigue, pertes,…

2.1.2 Visioconférence

La deuxième application est une application de visioconférence classique mettant en jeu deux entités. Ces dernières peuvent être à la fois émettrice et réceptrice de flux audio et vidéo.

Il est important de signaler ici qu'il a été délibérément choisi de travailler sur une application dont le choix de conception est différent de celui du serveur vidéo, dans le but d'étendre le cadre de notre étude aux modifications à apporter aux applications. Cette différence se traduit par le fait que la visioconférence n'a pas été conçue pour fonctionner au-dessus d'un autre service de transport que celui offert par UDP (les programmeurs n'ont donc pas utilisé les fonctionnalités de l'API JMF/RTP permettant d'utiliser un autre protocole de Transport qu'UDP).

2.2 Modules d'adaptation

Dans ce paragraphe, nous présentons deux extensions de l'API afin de faciliter le paramétrage des canaux. La première offre aux programmeurs d'application différentes primitives de création d'un canal (*openChannel(…)*) plus adaptées aux applications. La deuxième permet aux programmeurs de ne pas spécifier la QdS désirée, le système de communication intégrant une interface graphique (IHM) destinée aux utilisateurs afin de configurer leur communication au moment du lancement des applications. Nous rappelons que jusqu'à présent, le programmeur avait comme alternative :

- soit de prévoir une interface permettant aux futurs utilisateurs de saisir les paramètres de QdS définis dans le chapitre 2 ;
- soit de fixer d'une manière statique les paramètres à l'appel de la primitive *openChannel*.

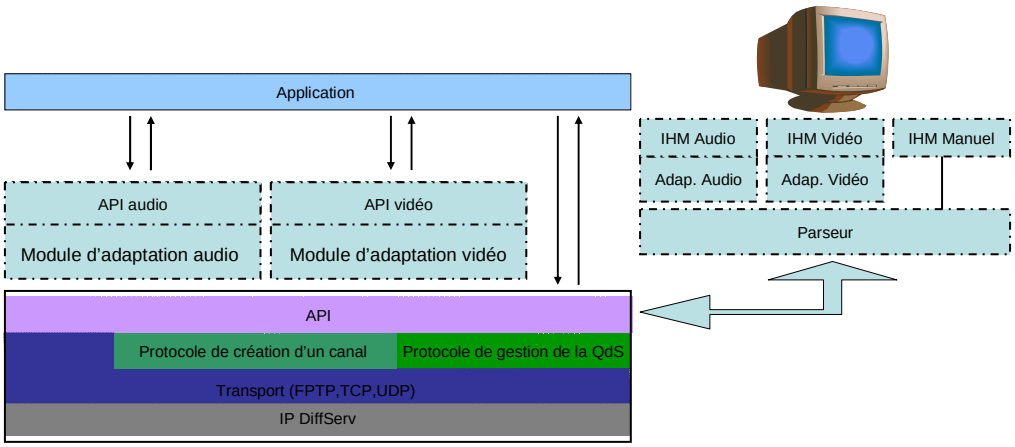


Figure 78 : Extension de l'architecture d'un hôte

L'essentiel de la difficulté dans l'utilisation de notre système de communication réside dans le fait que le programmeur d'application doit paramétrer la création des canaux à l'aide des paramètres de QdS que nous avons spécifiés précédemment dans le Chapitre 2. En effet, si le choix de ces paramètres a été effectué afin de satisfaire l'ensemble des applications multimédias, nous avons conscience que les utilisateurs risquent d'avoir des difficultés pour effectuer ce paramétrage.

Sur la Figure 78, nous avons représenté le principe des modules d'adaptation qui permettraient d'offrir aux programmeurs d'application différentes primitives *openChannel* avec des paramètres de QdS adaptés aux principales applications multimédias. Les programmeurs utiliseraient donc des API *ad hoc* pour chacune des applications. Les modules permettent de traduire les nouveaux paramètres en terme de paramètres de QdS génériques utilisés par l'algorithme de sélection automatique des services. Cette traduction se fait sur les hôtes d'extrémité avant l'établissement du protocole de création de canal ; ce ne sont donc que les paramètres de QdS définis initialement qui sont envoyés dans la requête de QdS au BB local et qui seront utilisés sur le BB local.

Dans le cas où le programmeur de l'application ne prévoit pas de paramétrer la QdS de l'application (ce qu'il pourrait faire par exemple soit en fixant les paramètres, soit en proposant une interface ad hoc), nous proposons une interface graphique qui permet aux utilisateurs de l'application de paramétrer la connexion. Pour cela, le programmeur invoque la primitive *openChannel (String sessionName, String remoteAddress, int dataPort, String role, qosClass QoS)* avec une classe *qosClass* nulle. Dans ce cas, le système propose une interface graphique qui permet aux utilisateurs de paramétrer leur connexion. Cette IHM a deux modes distincts :

- un mode manuel dans lequel les paramètres de QdS sont entrés directement ;
- un mode simplifié mettant en œuvre les mêmes modules d'adaptation que précédemment. L'utilisateur n'ayant pas forcément la connaissance des paramètres précis à employer, l'IHM lui offre la possibilité de caractériser la QdS via des paramètres adaptés, comme le nombre d'images par seconde ou la résolution, dans le cas d'une vidéo, la fréquence d'échantillonnage dans le cas de l'audio,…

La Figure 79 représente l'IHM mise à disposition des utilisateurs. Son menu permet à l'utilisateur soit de configurer une connexion, soit de charger une configuration pré-enregistrée. La fenêtre se découpe en trois zones principales. Dans la première (cadre 1), l'utilisateur peut rentrer les paramètres de QdS du canal. La deuxième (cadre 2) est un rappel des choix effectués. Dans la troisième zone (cadre 3), l'utilisateur peut suivre la progression de la connexion grâce à la barre d'avancement. C'est également dans cette zone que les messages d'erreur (QdS émettrice différente de la QdS réceptrice, serveur d'enregistrement non présent, ressources non disponibles…) apparaissent.

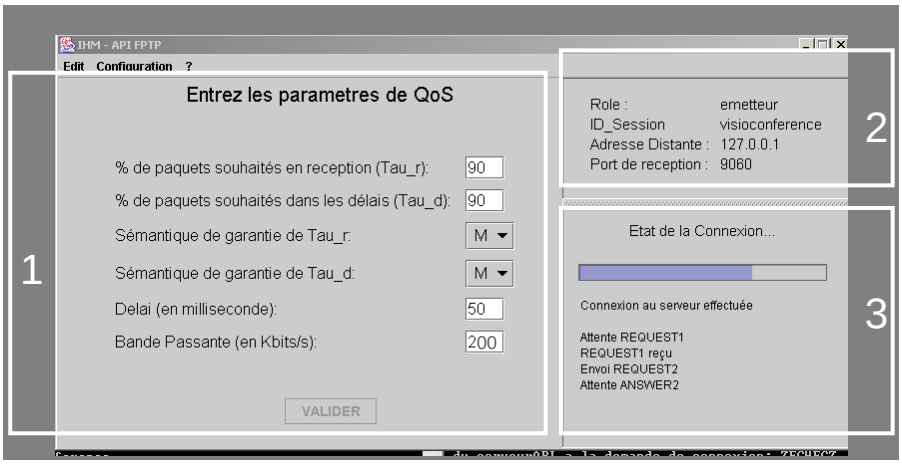


Figure 79 : IHM utilisateur

2.3 Modification minimale des applications

Dans ce paragraphe, nous proposons d'étudier un exemple particulier d'intégration d'applications existantes avec notre système de communication. La méthode décrite ici n'est pas la seule, car il faut bien souvent modifier plus profondément les applications avec le désagrément que chaque application requiert une adaptation particulière. L'intégration des applications nouvelles ne présente pas de difficulté particulière, l'API de notre système de communication, largement décrite dans le chapitre 2, s'utilisant comme une API Transport classique.

Les applications multimédias utilisant la JMF font quasiment toutes appel à UDP pour des raisons évidentes de contraintes temporelles. L'accès au service Transport se fait alors via la classe *DatagramSocket* du paquetage *java.net* avec les primitives qui lui sont rattachées. Le type d'intégration présenté dans ce paragraphe est fortement lié au langage objet et au principe de surcharge (de classe ou de méthode). En effet, afin d'avoir un minimum de modifications à apporter aux applications, nous avons intégré la possibilité de surcharger la classe *DatagramSocket* pour lui adjoindre l'ensemble de notre API. L'idée est de limiter le portage d'une application à la modification d'une importation de paquetage.

La méthode décrite ci-dessus se révèle très efficace hormis le problème de la distinction entre les *DatagramSocket* dédiées aux connexions RTP et celles dédiées à RTCP. En effet, il ne faut pas avoir recours au protocole de création d'un canal de notre architecture de signalisation pour les *DatagramSocket* relatives aux données de contrôle (RTCP).

Il est donc incontournable d'utiliser deux paquetages différents (*java.net* et celui de notre API) au sein de la même classe. La modification du code de l'application se limite seulement à opérer la distinction entre ces deux types de *DatagramSocket*. La connaissance de la JMF nécessaire et l'immersion dans le code de l'application sont donc sans commune mesure avec celles impliquées par une modification systématique.

Par ailleurs, l'idée de la surcharge implique que l'application n'a aucune connaissance des paramètres tels que la QdS à demander, l'adresse distante, le nom de la session, le rôle, le numéro de port, éléments indispensables pour l'invocation des primitives *openSession(…)* et *openChannel(…)* : nous avons donc intégré l'IHM définie précédemment pour pallier l'absence de ces paramètres. Lors de l'ouverture d'une session, l'utilisateur doit entrer un nom de session, un port de réception de données si l'application n'en a pas spécifié un en particulier, un rôle (émetteur/récepteur) et enfin l'adresse IP distante. Notons que quelques-uns de ces paramètres peuvent être fixés dans la surcharge de *DatagramSocket*, et que l'IHM interdit alors leur modification.

3 Description de la plate-forme de test

Dans ce paragraphe, nous présentons la plate-forme d'émulation qui nous a permis de tester le déploiement de notre système de communication.

L'intérêt d'utiliser un émulateur (au lieu d'un simulateur) est double. Le premier est qu'un environnement d'émulation réseau permet d'utiliser directement les protocoles et applications existants, à l'inverse d'un simulateur (type *ns-2*) qui demanderait de concevoir les modules nécessaires. Le second concerne l'aspect démonstratif plus convaincant puisqu'il conserve les protocoles et applications réels et non leurs modèles.

Nous présentons dans un premier temps les principes de conception de la plate-forme d'émulation (3.1). Dans un deuxième temps, nous présentons les principaux choix d'implémentation de l'environnement multi domaines (3.2).

3.1 Plate-forme d'émulation

3.1.1 Présentation de Dummynet

Dummynet est un logiciel basé sur l'utilisation du pare-feu disponible dans le système d'exploitation FreeBSD. C'est un *shaper* de trafic développé à l'origine par (Rizzo, 1997) pour tester les connexions TCP. Il permet de modifier les propriétés du trafic traversant l'émulateur en appliquant un délai, une bande passante et un taux d'erreur paquet.

Le pare-feu délivré par la distribution de FreeBSD est appelé IP *firewall* et est accessible par la commande *ipfw*. Il intercepte les paquets, en fonction de filtres, au niveau des interfaces physiques de l'émulateur et les redirige vers le logiciel. Pour le filtrage, il possède un ensemble de règles qui permet de rediriger uniquement les paquets souhaités. Les règles se construisent à partir des champs des en-têtes des paquets IP, et permettent de filtrer selon un ou plusieurs des paramètres suivants[34] :

– interface physique d'émission ou de réception ;

[34] D'autres paramètres peuvent être utilisés (comme par exemple les *flags* des paquets TCP), mais nous présentons ici les plus courants que nous retrouverons dans notre plate-forme. Les lecteurs désireux d'avoir une liste exhaustive des paramètres peuvent se reporter aux pages du manuel de la commande *ipfw* sous FreeBSD.

– adresse IP source et/ou destination ;
– port destination ;
– protocole.

Pour les paquets correspondant au filtre d'une règle, *Dummynet* permet de :

– limiter le débit du flux de ces paquets concernés ;
– modifier les délais de transit de ces paquets ;
– introduire des pertes de paquets.

3.1.2 Émulation d'un domaine Diffserv

Dans le chapitre 3, nous avons abordé le concept de plate-forme d'émulation pour valider les propriétés de la convolution (adaptée à un environnement Internet) par des tests à l'aide de la commande *ping*. Le flux ICMP généré avait alors un faible débit. Le choix de ce débit vient d'une contrainte forte liée à l'utilisation de l'émulateur *Dummynet* pour reproduire le comportement d'un domaine DiffServ. Nous rappelons brièvement le principe de l'émulateur de domaines DiffServ dans la Figure 80.

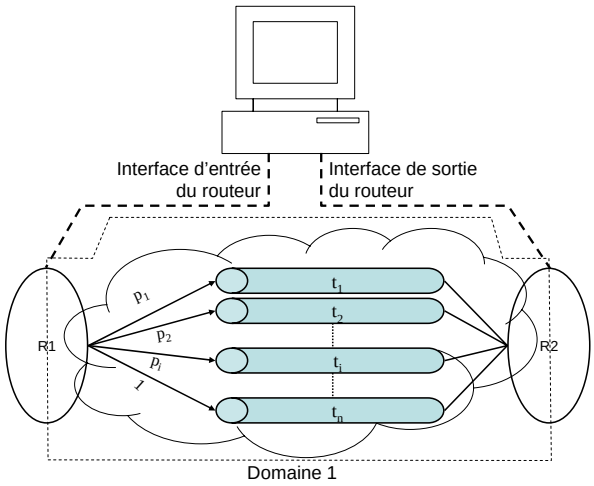


Figure 80 : Rappel de l'émulation d'un domaine DiffServ

Lorsqu'un paquet correspondant aux règles du filtre est détourné vers *Dummynet*, le logiciel donne une probabilité p_i de passage dans les tubes dont le délai de transit t_i est paramétré en accord avec la fonction de répartition souhaitée.

La contrainte qu'il faut obligatoirement vérifier pour émuler le comportement d'un domaine DiffServ est que la gigue maximum sur le réseau doit être inférieure au plus petit temps d'émission inter paquets d'un flux. Pour justifier cette contrainte, prenons le cas d'un flux de n paquets avec :

– le paquet p arrivant dans l'émulateur à la date t_p et auquel le logiciel applique un délai d_p ;

– le paquet $p+1$ arrivant à la date t_p+i_{p+1} (avec i_{p+1} le temps inter paquets entre les paquets p et $p+1$) et auquel le logiciel applique un délai d_{p+1}.

Il est nécessaire alors que :

$t_p+d_p < t_p+i_{p+1} + d_{p+1}$ soit $i_{p+1} > d_p - d_{p+1}$

Dans le cas contraire, cela signifierait que le paquet $p+1$ arrive avant le paquet p ; or dans l'environnement réel de référence (la plate-forme @IRS), nous n'avons pas constaté de déséquencement dans l'ordre d'arrivée des paquets d'un flux.

Une solution simple à implémenter est donc, de considérer que la gigue maximum sur le réseau doit être inférieure au plus petit temps d'émission inter paquets d'un flux. Soit pour un flux de n paquets :

$\forall p \in [1,n-1]\ d_{max} - d_{min} < i_{p+1}$

avec d_{max} (resp. d_{min}) le plus grand (resp. petit) délai que le logiciel puisse appliquer.

Pour les applications que nous avons retenues, nous avons réalisé des mesures des tailles des paquets et des débits générés à l'aide d'un analyseur de réseau. Le choix du paramétrage des applications a été dicté pour respecter cette contrainte.

3.1.3 Présentation de la plate-forme

L'environnement multi domaines a déjà été présenté dans le chapitre 2. L'objectif des tests est de déployer notre système sur un environnement reporté sur la Figure 81.

Nous y retrouvons deux types de clients : (1) isolé connecté directement à un domaine DiffServ ou (2) réseau d'entreprise qui permet aux machines de l'entreprise d'accéder au domaine DiffServ par un routeur[35].

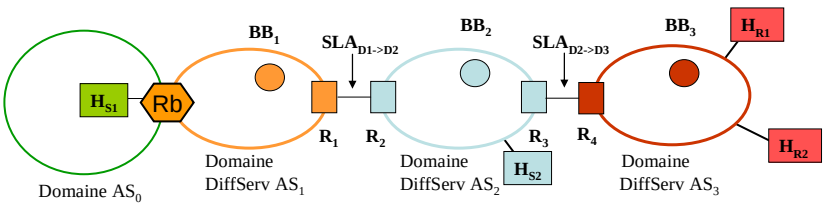


Figure 81 : Topologie retenue

Trois domaines DiffServ (AS_1, AS_2 et AS_3) sont mis en place. Le réseau d'entreprise est représenté par le domaine AS_0 relié au domaine DiffServ AS_1 par un routeur de bordure (R_b). Le client isolé est représenté par l'hôte H_{S2}, directement relié au domaine AS_2. Les *bandwidth brokers* sont les machines notées BB_1, BB_2 et BB_3. Les hôtes récepteurs sont rattachés au domaine DiffServ AS_3.

[35] En réalité, il se peut qu'il n'y ait pas un routeur mais deux routeurs : un placé dans le réseau de l'entreprise, l'autre dans le domaine de l'ISP. Cependant, par souci de clarté, nous ne représentons qu'un seul équipement.

Les tests seront effectués avec une application distribuée entre H_{S1} et H_{R1} ainsi qu'entre H_{S2} et H_{R2} ce qui permettra de tester au travers de BB_2 le double comportement des BB (*local* et *remote*).

Une simplification supplémentaire a été apportée au niveau des équipements inter domaines (routeurs R_1, R_2, R_3 et R_4). Pour des raisons de ressources matérielles, une seule machine assure la liaison entre deux domaines. Il en est de même pour la liaison entre AS_0 et AS_1. Chacune de ces machines n'assure qu'un rôle de routeur inter domaines (les mécanismes de *policing, shaping*,… ne sont pas implémentés sur notre plate-forme).

3.2 Choix d'implémentation retenus

Dans ce paragraphe, nous donnons quelques précisions sur les choix d'implémentation retenus. Le paragraphe (3.2.1) présente les hypothèses que nous avons faites sur les tables de routage des BB. Dans le paragraphe (3.2.2), nous étudions la configuration des contrats de QdS entre les AS et les clients. Le paragraphe (3.2.3) présente l'algorithme utilisé par les BB pour retrouver l'adresse du prochain BB à contacter lors d'une requête de QdS ainsi que les caractéristiques des services pour une route donnée.

3.2.1 Configuration des tables de routage des BB

La configuration des tables de routage des BB s'apparente fortement aux tables de routage BGP classiques dont nous avons reporté un extrait dans la Figure 82 (données extraites de la table du routeur *route-views.oregon-ix.net*).

Table 1	
Neighbor	AS
4.0.0.2	1
12.127.0.249	7018
128.177.255.6	3557
134.24.127.30	1740
134.55.20.229	293
144.228.241.81	1239
155.229.0.36	4565
157.22.9.7	715
163.179.232.37	2551
165.87.32.5	2685
167.142.3.6	5056
192.38.7.63	4513

Table 2						
Network	GateWay	Path				
192.5.100.0	167.142.3.6	5056	701	6226		
198.199.136.0	205.215.45.50	4006	1239	14801		
164.224.0.0	207.172.6.173	6079	7170	22	5855	
208.48.110.0	198.32.162.18	4513	701	13445		
193.54.0.0/16	167.142.3.6	5056	1	5511	2200	
129.88.0.0	167.142.3.6	5056	1	5511	2200	1942
147.127.0.0	167.142.3.6	5056	1	5511	2200	2188
137.227.0.0	167.142.3.6	5056	1	297	1842	
192.144.75.0/23	167.142.3.6	5056	1	5511	2200	
140.93.0.0	167.142.3.6	5056	1	5511	2200	1305

Figure 82 : Extrait d'une table BGPv4

Ce routeur est accessible pour le public par une connexion *telnet* classique et permet de découvrir sa table BGP. Différentes commandes sont disponibles pour en extraire les données contenues. Dans la première table, nous retrouvons l'adresse des routeurs de bordure (1ère colonne) permettant l'accès aux AS voisins (2ème colonne). Pour notre utilisation, qui est de pouvoir retrouver l'adresse du prochain BB à contacter pour transmettre la requête, il suffit de compléter cette table en rajoutant l'adresse du prochain BB qui gère la QdS dans chacun des AS voisins. Par la suite, nous complèterons donc cette première table comme il est représenté dans le Tableau 24.

Neighbor	AS	BB IP Address
…	…	…

Tableau 24 : Modélisation des tables BGP

La deuxième table de la Figure 82 montre comment il est possible, à partir d'une adresse de réseau de destination, de connaître le prochain routeur de bordure et la liste des AS successifs qui seront traversés. Nous conserverons cette table sans y apporter de modification. Nous avons reporté dans les dernières lignes (grisées) de cette table, les adresses réseaux de l'ENSICA, l'IMAG, l'ENSEEIHT, du LIP6, un réseau F&T RD, et enfin celui du LAAS-CNRS !!!

Le paragraphe 3.2.3 expliquera comment les BB utilisent ces deux tables pour propager les requêtes de QdS.

3.2.2 Configuration des tables de QdS des BB

Dans ce paragraphe (Tableau 25), nous présentons les tables de configuration des BB, dans lesquelles sont reportés

- tous les contrats négociés avec les clients (AS ou clients isolés) ;
- les caractéristiques des services IP :
  - soit entre deux routeurs de bordure de l'AS du BB considéré ;
  - soit entre deux clients isolés de l'AS du BB considéré ;
  - soit entre un client isolé de l'AS du BB considéré et un AS voisin (et inversement).

Nom du client											
bandwidth		AS_max	AS_disp	GS_max	GS_disp						
		50	30	40	20						
destinations		abs1	ord1	abs2	ord2	abs3	ord3	abs4	ord4	abs5	ord5
Destination 1	AS	30	0	35	25	40	50	45	75	47	100
	GS	25									
Destination 2	AS	28	0	32	25	39	50	40	75	42	100
	GS	22									
…	AS	…	…	…	…	…	…	…	…	…	…
	GS	…									

Tableau 25 : Table de caractérisation des services d'un BB

La première case du tableau indique le nom du client (qui peut être un AS ou un client isolé).

La deuxième ligne indique la quantité de bande passante que le client a acheté pour les deux services (colonne AS_max et GS_max) ainsi que les quantités effectivement disponibles, à l'instant donné, compte tenu des réservations en cours (colonne AS_disp et GS_disp).

Les lignes suivantes donnent les performances des services pour les destinations accessibles. Les performances sont mémorisées pour le service :

- AS, sous la forme de cinq couples de points : abs*i* = délai en millisecondes et ord*i* = pourcentage de paquets reçus ;
- GS, sous la forme d'un délai de transit maximum.

Nous rappelons que le modèle « en escalier » a été retenu pour représenter les performances des services AS. Ainsi, nous avons, par exemple pour la première destination, un délai minimum de 30 ms, 25% des paquets auront un délai de transit inférieur à 35ms, 50% inférieur à 40ms, …

3.2.3 Algorithme de routage spécifique aux BB

Nous présentons dans ce paragraphe l'algorithme de routage utilisé par un BB lorsqu'il reçoit une requête de QdS et qu'il cherche à déterminer l'adresse IP du prochain BB auquel il doit (éventuellement) transférer la requête et quelles sont les caractéristiques des services à retourner au BB *local* (sauf s'il est lui-même le BB *local*).

Sur réception d'une requête (PDU REQUEST_2 ou REQUEST_3), le BB connaît :

- l'adresse du récepteur final (contenue dans le PDU) ;
- l'adresse de celui (BB ou client) qui lui a envoyé la requête (contenue dans l'en-tête du paquet IP) ;
- l'adresse du BB local (contenue dans l'en-tête du paquet IP si c'est un PDU REQUEST_2 sinon dans le PDU lui-même).

L'adresse du récepteur lui permet de savoir si ce récepteur est directement accessible ou s'il doit propager la requête. Si ce récepteur se trouve :

- directement accessible, le BB doit retourner au BB *local* (sauf s'il est lui-même le BB *local*) les caractéristiques de la route entre le point d'entrée dans son domaine et la destination. Pour cela, il

regarde l'adresse du BB (ou du client) qui lui a envoyé la requête (qui peut être le BB *local* ou le BB *remote* le précédant). À partir de cette adresse et avec l'aide de la deuxième table BGP dont nous illustrerons son utilisation dans l'exemple suivant, il détermine le numéro d'AS d'appartenance de l'émetteur de la requête et en déduit l'adresse du routeur de bordure via lequel la requête est entrée dans son domaine. La table de caractérisation des services lui permet alors de retourner les caractéristiques entre le point d'entrée et la destination ;

– en dehors de son domaine, le BB doit tout d'abord retourner au BB *local* les caractéristiques de la route entre le point d'entrée dans son domaine et le point de sortie. Pour identifier le point d'entrée, la même procédure que précédemment est employée. Pour le point de sortie, la deuxième table BGP permet de connaître le prochain AS qui sera traversé. La table des caractéristiques lui donne alors les caractéristiques entre le point d'entrée et de sortie. Dans un deuxième temps, le BB doit également transférer la requête au prochain BB ; pour cela, il a besoin de l'adresse de celui-ci, qu'il retrouve dans la première table BGP (grâce à notre extension de la table BGP).

Considérons l'exemple de la Figure 83 pour illustrer le déroulement de cet algorithme au niveau du BB2 de l'AS2.

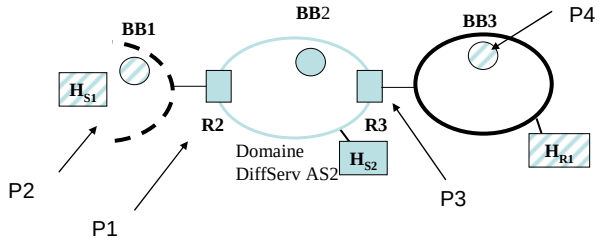


Figure 83 : Exemple de routage du BB_2

Les tables de routage du BB2 (extraites des scénarios de tests présentés dans le paragraphe (4)) sont les suivantes :

Table 1		
Neighbor	AS	BB IP Address
@ de R2	1	@ de BB1
@ de R3	5	@ de BB3

P4

Table 2		
Network	GateWay	Path
@ de AS0	@ de R2	1 0
@ de AS1	@ de R2	1
@ de AS3	@ de R3	3

P1 P2 P3

Tableau 26 : Configuration des tables de routage de BB2

Dans cet exemple, l'émetteur est H_{S1} et les récepteurs sont H_{S2} ou H_{R1}.

Lorsque BB2 reçoit une requête, il regarde l'adresse du récepteur ; il y a deux possibilités :

– le récepteur est directement accessible (par exemple H_{S2}) ; le récepteur est donc un de ses clients : l'adresse IP du récepteur appartient à l'AS du BB2 ou appartient à un AS d'un réseau client (non DiffServ) d'entreprise. Il regarde alors l'adresse source du paquet IP (dans notre exemple l'adresse de BB1) qu'il vient de recevoir. La deuxième table BGP lui permet de trouver l'adresse du routeur de bordure par lequel le paquet est entré dans le domaine – ici R2 (point P1) – ainsi que le numéro d'AS (point P2) correspondant au BB_1. La table des caractéristiques permet au BB de retourner les caractéristiques des services entre R2 et le récepteur client (H_{S2}) ;
– le récepteur n'est pas directement accessible (par exemple H_{R1}). À partir de l'adresse de destination et de la deuxième table BGP, le BB connaît le prochain AS à atteindre (dans notre cas l'AS3 – point P3 –). De la même façon que dans le cas précédent, le BB détermine le numéro de l'AS d'où provient la requête (points P1 et P2), et peut donc retourner au BB local les caractéristiques des services entre les points d'entrée et de sortie de son AS. Pour propager la requête, il consulte alors la première table BGP qui lui fournit l'adresse du prochain BB (dans notre cas le BB3 – point P4 –).

4 Tests expérimentaux

4.1 Configuration de la plate-forme

Sur la base de la Figure 81, la plate-forme d'émulation (Figure 84) est constituée de trois AS DiffServ (AS3, AS4 et AS5), d'un AS non DiffServ AS2 incluant un client émetteur (H_{S1}) relié à AS3 par un routeur de bordure (Rb), d'un hôte émetteur (H_{S2}) directement relié à AS4 et de deux hôtes récepteurs (H_{R1} et H_{R2}) situés sur AS5. Les passerelles inter AS sont les routeurs sur lesquels le logiciel *Dummynet* est activé. En pratique, toutes les machines sont connectées au même commutateur Ethernet sur lequel nous avons créé autant de VLAN que d'AS (VLAN 2 à 5). Chaque AS est ainsi supposé ne comporter qu'une seule adresse de réseau IP.

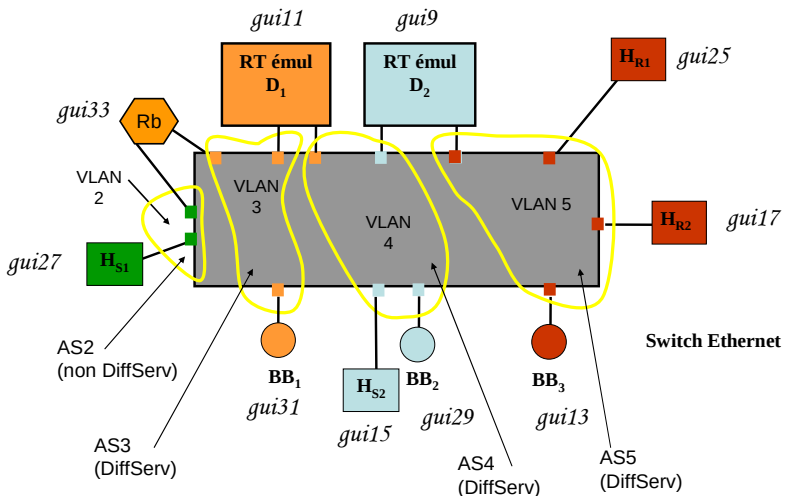


Figure 84 : Plate-forme d'émulation

Les conventions pour le plan d'adressage des machines sont les suivantes :

- les adresses réseaux des différents AS sont sous la forme 192.168.*vid*.0/24, avec *vid* le numéro de VLAN associé ;
- les adresses des machines sont sous la forme 192.168.*vid*.*x*, avec *x* le numéro de la machine *guix*.

Les tables de routage et des caractéristiques contenues dans les BB sont les suivantes.

BB1

Routage

Table 1		
Neighbor	AS	BB IP Address
192.168.3.33	2	-
192.168.3.11	4	192.168.4.29

Table 2		
Network	GateWay	Path
192.168.2.0	192.168.3.33	2
192.168.4.0	192.168.3.11	4
192.168.5.0	192.168.3.11	4 5

Description de la QdS

Client : AS2											
bandwidth		AS_max	AS_disp	GS_max	GS_disp						
		50	30	40	20						
destination		abs1	ord1	abs2	ord2	abs3	ord3	abs4	ord4	abs5	ord5
AS4	AS	30	0	35	25	40	50	45	75	47	100
	GS	25									

Client : AS 4											
bandwidth		AS_max	AS_disp	GS_Max	GS_disp						
		40	10	20	20						
destination		abs1	ord1	abs2	ord2	abs3	ord3	abs4	ord4	abs5	ord5
AS2	AS	32	0	38	25	41	50	49	75	50	100
	GS	32									

BB2

Routage

Table 1		
Neighbor	AS	BB IP Address
192.168.4.11	3	192.168.3.31
192.168.4.9	5	192.168.5.13

Table 2		
Network	GateWay	Path
192.168.2.0	192.168.4.11	3 2
192.168.3.0	192.168.4.11	3
192.168.5.0	192.168.4.9	5

Description de la QdS

Client : 192.168.4.15											
bandwidth		AS_max	AS_disp	GS_max	GS_disp						
		8	4	8	4						
destination		abs1	ord1	abs2	ord2	abs3	ord3	abs4	ord4	abs5	ord5
AS3	AS	31	0	34	25	43	50	47	75	50	100
	GS	31									
AS5	AS	29	0	36	25	41	50	42	75	45	100
	GS	29									

Client : AS3											
bandwidth		AS_max	AS_disp	GS_max	GS_disp						
		40	10	20	0						
destination		abs1	ord1	abs2	ord2	abs3	ord3	abs4	ord4	abs5	ord5
192.168.4.15	AS	28	0	33	25	40	50	41	75	44	100
	GS	28									
AS5	AS	31	0	34	25	41	50	44	75	45	100
	GS	31									

Client : AS5											
bandwidth		AS_max	AS_disp	GS_max	GS_disp						
		60	40	40	30						
destination		abs1	ord1	abs2	ord2	abs3	ord3	abs4	ord4	abs5	ord5
192.168.4.15	AS	33	0	35	25	42	50	46	75	49	100
	GS	33									
AS3	AS	30	0	33	25	42	50	44	75	46	100
	GS	30									

BB3

Routage

Table 1		
Neighbor	AS	BB IP Address
192.168.5.9	4	192.168.4.29

Table 2		
Network	GateWay	Path
192.168.2.0	192.168.5.9	4 3 2
192.168.3.0	192.168.5.9	4 3
192.168.4.0	192.168.5.9	4

Description de la QdS

Client : AS4											
bandwidth		AS_max	AS_disp	GS_max	GS_disp						
		8	8	3	3						
destination		abs1	ord1	abs2	ord2	abs3	ord3	abs4	ord4	abs5	ord5
192.168.5.25	AS	27	0	32	25	33	50	43	75	45	100
	GS	30									
192.168.5.17	AS	26	0	31	25	32	50	42	75	45	100
	GS	30									

4.2 Description des scénarios

Dans le Chapitre 4, nous avons présenté la spécification des protocoles en UML2.0, ce qui nous a permis de faire une validation fonctionnelle des architectures de signalisation et de communication. L'objectif des scénarios décrits dans ce paragraphe est de valider le déploiement des protocoles de signalisation. Le principe des tests est donc de lancer une ou deux application(s) multimédia(s) avec différents paramètres de QdS et débits requis puis d'analyser le comportement des BB vis-à-vis de la sélection et de la réservation des services. Pour cela, nous avons spécifié un certain nombre de scénarios de tests (Tableau 27) visant à valider les phases d'établissement de connexion, les différents cas d'échec d'établissement d'un canal et le comportement du système vis-à-vis d'une rupture de réservation due à la perte de plusieurs PDU de maintien de réservation. Nous avons également vérifié le double comportement d'un BB : *local* et *remote*.

		Scénario
Connexion réussie		1
Connexion échouée	Requête de la QdS de l'émetteur ≠ Requête de la QdS du récepteur	2
	Ressources ne permettant pas de satisfaire momentanément la QdS	3
	Ressources ne permettant pas de satisfaire définitivement la QdS	4
	Connexion non disponible	5
Pertes des PDU de maintien des ressources	REFRESH_2	6
	REFRESH_3	7
	REFRESH_4	8
Double comportement simultané d'un BB : *local* et *remote*		9

Tableau 27 : Scénarios de tests

4.3 Résultats des tests

Ce paragraphe présente les résultats obtenus pour les scénarios précédents ainsi qu'une synthèse sous la forme de captures d'écran qui permet de visualiser le comportement d'un BB en temps réel.

Scénario 1

Les résultats obtenus ont permis de vérifier que les services Transport et IP sélectionnés sont en accord avec la requête de l'application. Toutes les combinaisons possibles de couple Transport / IP ont été testées, avec notamment, le cas le plus complet qui est celui d'une requête dont les sémantiques de garanties auraient autorisé le service AS, mais les performances du réseau en terme de délai de

transit de bout en bout – calculé à partir de la convolution des caractéristiques des trois domaines – ont imposé la sélection du service GS : le pourcentage de paquets reçus dans les délais était inférieur à la valeur donnée dans la requête du client.

Scénario 2

Ce scénario a permis de contrôler le bon fonctionnement des mécanismes de vérification de l'égalité entre les QdS émettrice et réceptrice. Dans le cas contraire, les utilisateurs sont amenés à re-saisir leurs paramètres.

Scénario 3 et 4

Ces scénarios ont permis de vérifier que les mécanismes de réservation de ressources sur un BB rejettent les requêtes dans les cas suivants :

- le client a déjà utilisé les ressources dont il dispose ; cependant sa requête pourra être satisfaite ultérieurement lorsque celui-ci aura relâché des ressources (sous condition que les caractéristiques des services soient compatibles) ; ce cas de figure correspond à une demande de débit comprise entre AS_max et AS_disp et entre GS_max et GS_disp ;
- le client demande plus de ressources que ce qui est prévu dans son contrat ; en d'autres termes, le débit demandé par la requête est supérieur à AS_max et GS_max.

Scénario 5

Ce scénario a permis de valider qu'une erreur due à l'absence momentanée ou définitive du serveur d'API (décrit au Chapitre 4) est convenablement traitée. Cette étape permet au récepteur de récupérer le numéro de port qui sera utilisé pour la connexion de signalisation.

Scénarios 6, 7 et 8

Ces scénarios ont permis de vérifier que la perte d'un PDU de maintien des ressources provoque bien la déconnexion de l'ensemble des équipements avec la libération des ressources.

Scénario 9

Ce scénario a permis de vérifier qu'un BB *local* peut simultanément avoir un comportement de type *local* et *remote* pour différents flux.

Synthèse

Les Figure 85, Figure 86 et Figure 87 sont les captures d'écran réalisées à partir de la machine BB2 (*gui29*). Nous commentons ces figures pour vérifier le comportement de ce BB lorsque deux applications lancées sur H_{S1} (*gui27*) et H_{S2} (*gui15*) émettent une requête pour établir une connexion

avec H_{R1} (*gui25*) et H_{R2} (*gui17*). Nous nous sommes placés dans un cas où les connexions vont pouvoir s'établir correctement, pour ne pas surcharger les figures.

Étape 1 : Initialisation du BB et réception par gui15 de la requête de QdS

Les numéros de cadre sont relatifs à la Figure 85.

Le cadre 1 montre l'initialisation du système avec la mise en attente d'une requête pour le BB.

Le cadre 2 montre que le BB a reçu une requête de la part d'un de ces clients (REQUEST_2 DEMAND RECEIVED).

Le cadre 3 montre que le BB récupère l'adresse du prochain BB *remote* (Next BBR /192.168.5.13 : gui13) et effectue les pré réservations.

Le cadre 4 montre que la requête a été transférée (Request done !) et que le BB a reçu une réponse de la part du BB *remote* (ANSWER_3 RECEIVED).

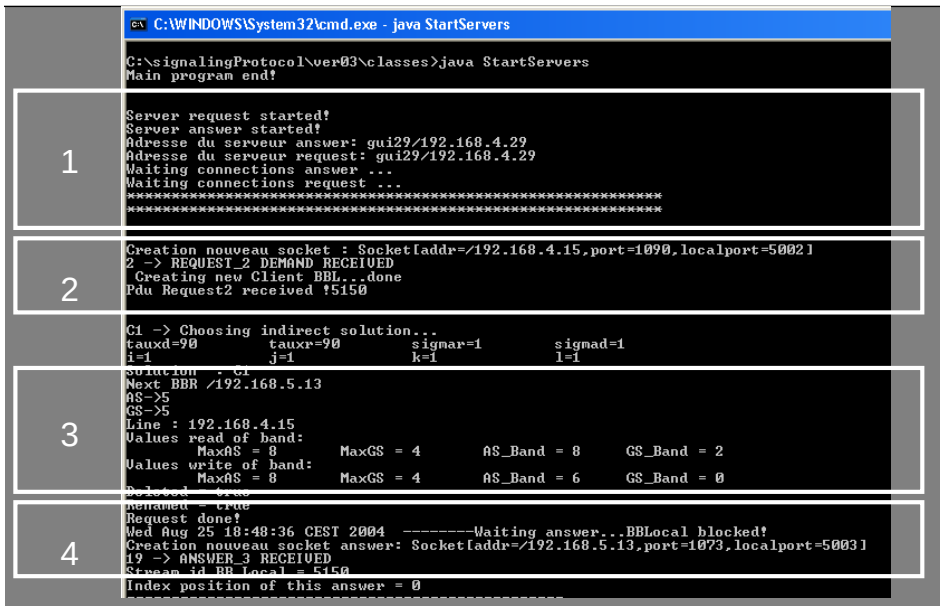


Figure 85 : Test de la dualité du comportement d'un BB (1)

Étape 2 : Sélection des services, réponse à l'émetteur et attente des PDU de maintien

Les numéros de cadre sont relatifs à la Figure 86.

Le cadre 1 montre que toutes les réponses (All answers received → timer canceled) ont été reçues (gui13 est effectivement le seul BB *remote*) avec désactivation du *timer,* puis que le BB a effectué le choix des services.

Le cadre 2 montre que le couple UDP/GS a été retenu et que le BB a actualisé alors les tables de disponibilité (contenues dans les tables des caractéristiques des services vues précédemment) en conséquence. Il transmet également au BB *remote* l'entérinement des réservations (Reserve sent to BB Remote no 2).

Le cadre 3 montre que le BB retourne à l'hôte émetteur le couple de service (Answer2(0) sent to Sender …) puis qu'il se met en attente d'un PDU REFRESH_2 pour maintenir la réservation ou CLOSE_2 pour fermer la connexion et libérer les ressources.

Le cadre 4 montre que le BB reçoit bien les PDU de maintien des ressources (Refresh_2 received …).

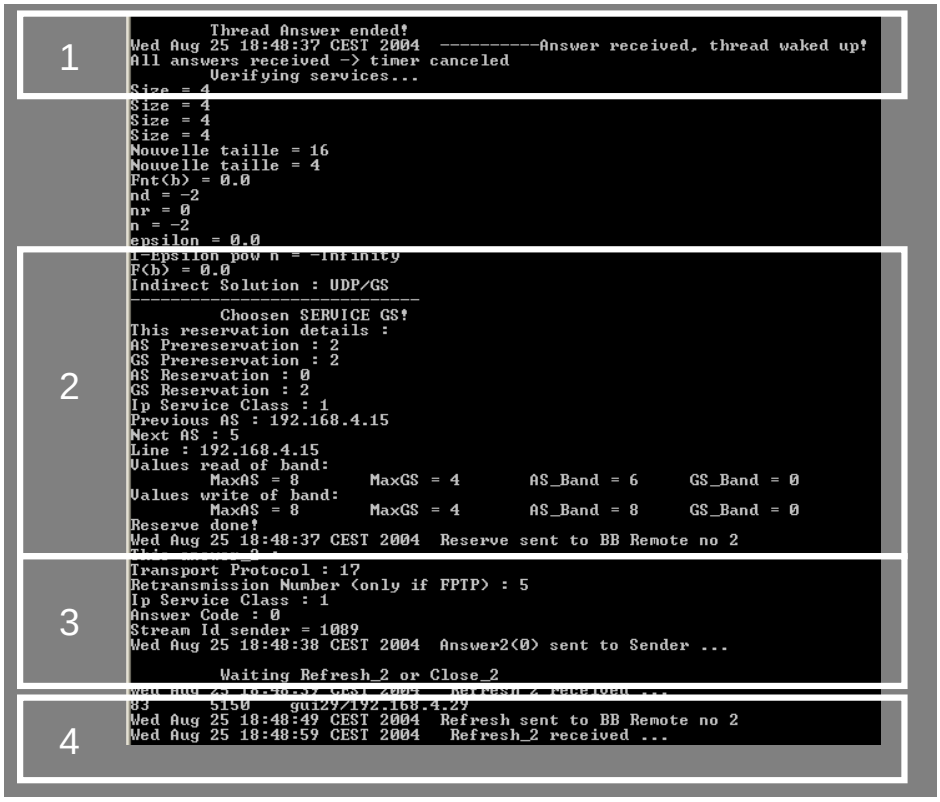


Figure 86 : Test de la dualité du comportement d'un BB (2)

<u>Étape 3 : Réception d'une requête supplémentaire d'un hôte distant.</u>

Les numéros de cadre sont relatifs à la Figure 87.

Le cadre 1 montre qu'une nouvelle requête émise par le BB gui31 (REQUEST_3 DEMAND RECEIVED) est arrivée sur le BB.

Le cadre 2 montre que le BB a retrouvé le numéro d'AS de gui31 (Previous AS : 3) et le numéro du prochain AS (Next AS : 5) ainsi que les pré-réservations.

Le cadre 3 montre que le BB doit entériner la réservation pour le service GS (GS service asked).

Le cadre 4 montre alors que le BB se met en attente des PDU de rafraîchissement (REFRESH_2 ou 3) ou de fermeture de connexion (CLOSE_2 ou RELEASE_3) suivant les flux.

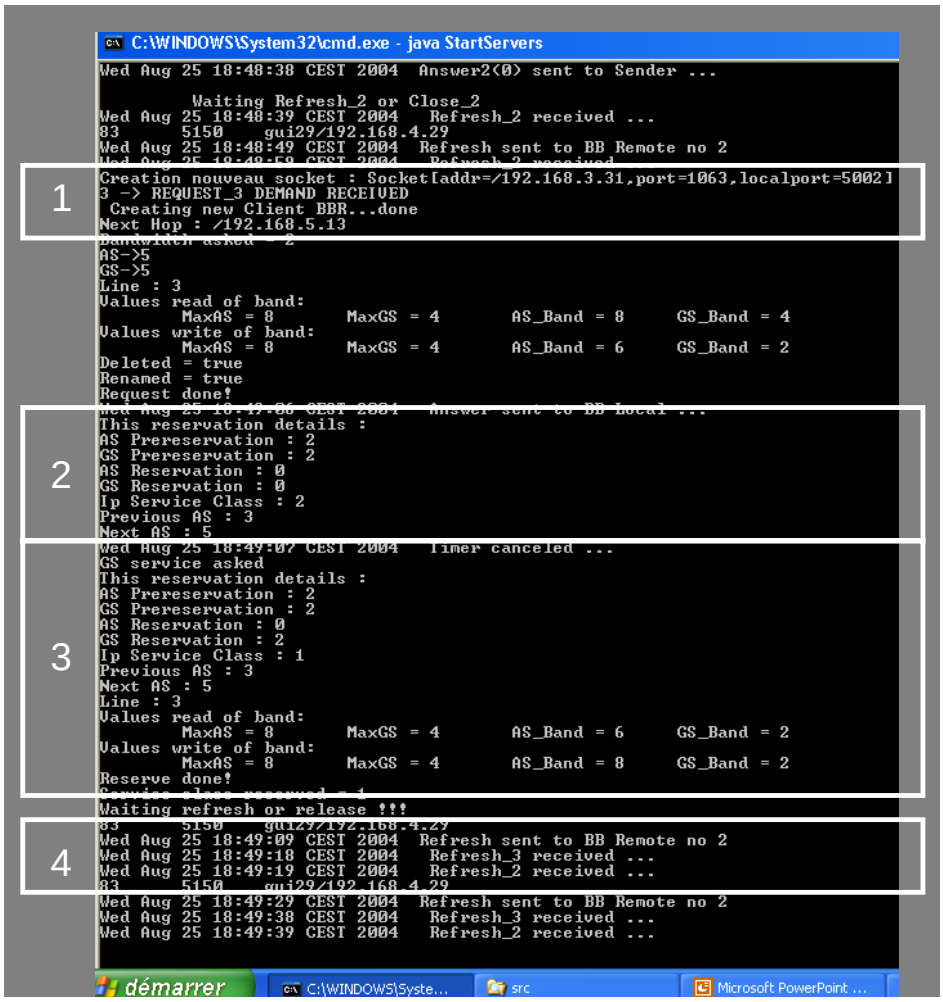


Figure 87 : Test de la dualité du comportement d'un BB (3)

Étapes 4 & 5 : Communications et libération des ressources

Ces étapes (non représentées) ont permis de vérifier que les ressources sont bien retenues jusqu'à la fin des communications.

4.4 Problème de la généralisation à *n* domaines

Les tests réalisés sur une plate-forme composée de trois domaines ont permis de confirmer la bonne adéquation entre la spécification du système et les choix d'implémentation. Ceci étant, la robustesse du système de communication vis-à-vis du facteur d'échelle reste à étudier.

Notons cependant que la modification des tables BGP, avec l'extension nécessaire pour mémoriser l'adresse du prochain BB, ne paraît pas irréalisable même pour un environnement à grande échelle. Pour indication, la table BGP du routeur *route-views.oregon-ix.net* (dont les tables 1 et 2 de la Figure 82 ont été extraites) occupe plus de 350 Moctets. Rajouter une adresse IP ne « coûte » que 8 octets supplémentaires par AS voisins (qui sont en nombre limité).

En outre, la charge en terme de ressource CPU d'un BB est faible, même si les implémentations faites en Java (pour faciliter le portage d'une plate-forme à l'autre) restent plus « coûteuses » que si elles avaient été faites en langage *C*.

La principale limite vient du nombre de connexions qui sont ouvertes lorsque les BB reçoivent des requêtes. Le chapitre relatif aux conclusions générales et perspectives reviendra sur ce point.

5 Conclusion

Dans ce chapitre, nous avons présenté les tests réalisés afin de valider les choix d'implémentation du système de communication et le déploiement des architectures de communication et de signalisation.

Dans un premier temps, nous avons présenté les applications multimédias utilisées pour les tests en abordant le problème de leur intégration dans le système de communication en choisissant deux types d'applications, chacune ayant un conception initiale distincte : prévoyant ou non de pouvoir utiliser une autre architecture que RTP/UDP. Nous avons également montré les mécanismes déployés pour faciliter son utilisation. En définissant le principe d'API *ad hoc* proposant des primitives et des paramètres plus adaptés aux applications multimédias classiques, la tâche du programmeur d'application sera encore facilitée. Enfin, nous avons également présenté une IHM qui permet à l'utilisateur de configurer sa connexion dans le cas où le programmeur ne prévoit pas de le faire.

Dans un second temps, nous avons détaillé la plate-forme émulant un environnement DiffServ multi domaines. Nous avons également présenté les extensions nécessaires aux tables BGP d'un BB incluant l'adresse des BB voisins, afin que notre système puisse être déployé. Le logiciel *Dummynet* a été utilisé pour son utilisation simple et performante. Cependant il présente dans notre cas d'utilisation (émulation de domaines DiffServ) une contrainte à respecter pour avoir un comportement correct : il faut impérativement veiller à ce que la gigue maximale observable sur le réseau soit inférieure au temps inter paquets. Les scénarios de tests ont été présentés ainsi que les résultats obtenus validant les choix d'implémentation du système et son déploiement sur une plate-forme multi domaines DiffServ.

Conclusion générale et Perspectives

L'engouement de ces dernières années pour les applications multimédias distribuées dans l'Internet est incontestable. Pour répondre aux contraintes nouvelles de ces applications (fiabilité partielle, délai borné,…), les communautés réseaux et télécommunications ont initié de nombreux travaux suivant (en particulier) deux axes :

- d'une part, ont émergé des propositions d'architectures de communication visant à intégrer les services Transport et IP pour offrir des garanties de QdS de bout en bout ;
- d'autre part, la constitution de l'Internet en domaines autonomes (par définition administrés par des fournisseurs distincts) a conduit à la proposition d'architectures de signalisation dites multi domaines, visant à assurer la continuité du service lors de la traversée de plusieurs domaines. Ces propositions s'attachent à donner des solutions à un ou plusieurs problèmes liés au contexte multi domaines tels que la définition d'un SLA standard, la découverte des services mis en place dans les différents domaines, la réservation des ressources, …

La thèse présentée dans ce mémoire contribue à apporter une solution architecturale couplant la prise en compte des problématiques de ces deux axes.

1 Conclusions

Les travaux présentés dans ce mémoire consistent en la proposition d'un système de communication couplant d'un part une architecture de communication intégrant niveaux Transport (UDP, TCP, FPTP) et IP (DiffServ) et d'autre part, une architecture de signalisation adaptée à un environnement (DiffServ) multi domaines. L'objectif ciblé est double : il est (1) de maîtriser la QdS de bout en bout, et (2) de fournir aux programmeurs d'application une API suffisamment simple leur permettant d'accéder de façon optimale au système de communication défini en minimisant les ressources utilisées.

La démarche adoptée pour aboutir à la réalisation de ce système de communication a été la suivante :

- étude de l'environnement IP ciblé en considérant un contexte mono puis multi domaines, ce qui nous a permis de poser le cadre de la thèse ;
- mesures et modélisation des performances des services DiffServ et Transport dans un contexte mono puis multi domaines ;
- étude et contribution au déploiement d'une architecture de communication intégrant services DiffServ et Transport ;
- définition et spécification d'une architecture de signalisation adaptée au contexte multi domaines ;

– déploiement du système de communication couplant les deux architectures précédentes sur une plate-forme émulant le comportement d'un environnement DiffServ multi domaines.

Pour chacune des étapes précédentes, nous détaillons à présent les contributions majeures de nos travaux.

1.1 Environnement IP DiffServ

La première étape de notre travail a été d'étudier l'environnement IP DiffServ pour poser le cadre de notre solution. Nous avons ainsi abordé les approches existant au niveau des architectures de communication multi services (Transport et IP) et au niveau des propositions des architectures de signalisation adaptées au multi domaines.

Nous avons alors défini l'environnement générique suivant, composé :

– d'hôtes d'extrémité pouvant appartenir à des domaines surdimensionnés non DiffServ ou pouvant être reliés directement à un domaine DiffServ ;
– de domaines de transit DiffServ implémentant chacun plusieurs classes de service ;
– d'entités administrant les ressources (réservation de ressources, caractérisation des services,…) sur le modèle des *Bandwidth Brokers* (BB).

1.2 Modélisation des performances des services

La deuxième étape a été de mesurer les performances des services IP et Transport sur la base de :

– trois services au niveau IP : le service GS qui offre à l'utilisateur le même service que celui offert par une ligne louée, le service AS qui donne la garantie à l'utilisateur d'une bande passante minimum et le service Best Effort de l'Internet classique ;
– trois services au niveau Transport fournis par les protocoles TCP, UDP et FPTP, ce dernier protocole offrant des garanties d'ordre (inter et intra flux) partiel et de fiabilité partielle.

Ces mesures ont été réalisées initialement sur la plate-forme expérimentale déployée dans le cadre du projet RNRT @IRS. Les résultats ainsi obtenus ont ensuite été confrontés à ceux d'une étude menée en simulation ns-2 étendant le cadre de l'étude expérimentale.

A la suite de ces tests, nous avons choisi un modèle permettant de caractériser les performances du service (obtenu par le couplage Transport et IP) entre deux routeurs de bordure d'un domaine pour un état donné de charge du réseau. Ce modèle, basé sur la fonction de répartition des délais des paquets d'un flux, a été validé dans un contexte étendu grâce à une plate-forme émulant le comportement d'un environnement DiffServ multi domaines.

1.3 Architecture de communication

Sur les bases d'une proposition d'architecture de communication (l'architecture @IRS) conçue pour offrir des garanties de QdS par flux applicatif, nous avons étendu la proposition en définissant un algorithme de sélection automatique des services Transport et IP soustrayant le programmeur d'application du choix de ces services dans un environnement multi domaines. Cet algorithme permet, au regard d'une requête applicative, de sélectionner (sous réserve que les ressources du réseau l'autorisent) la classe de service IP et le protocole de Transport (ainsi que son paramétrage pour FPTP) à mettre en œuvre pour satisfaire la requête. Par le principe même de cet algorithme choisissant le meilleur couple de services, nous optimisons ainsi les réservations de ressources.

1.4 Architecture de signalisation

Afin d'assurer la continuité du service offert aux applications sur l'ensemble des domaines traversés par une communication, nous avons défini, spécifié et implémenté une architecture de signalisation multi domaines. Cette architecture comporte quatre protocoles :

- un protocole de création de canal qui permet aux applications émettrice et réceptrice d'échanger la QdS souhaitée et, par la suite, de paramétrer correctement le canal pour l'échange des données ;
- un protocole de gestion de la QdS établi entre l'hôte émetteur et le BB sur lequel l'émetteur est déclaré comme client, appelé BB *local*. Il permet à l'émetteur de formuler auprès de ce BB une requête de QdS et d'obtenir une réponse de celui-ci ;
- un protocole de signalisation inter BB. Il permet d'obtenir l'état des ressources ainsi que les caractéristiques des services et de faire les réservations dans chaque domaine traversé ;
- un protocole de configuration des routeurs de bordure établi entre le routeur de bordure du site émetteur et le BB *local*.

Ces différents protocoles ont été spécifiés en UML 2.0 à l'aide de l'outil TAU de la société Telelogic, puis implémentés en JAVA.

1.5 Tests de déploiement du système de communication

La dernière étape a été de tester le système de communication sur une plate-forme expérimentale émulant le comportement de plusieurs domaines DiffServ (via le logiciel *Dummynet*). Pour cela, nous avons adapté deux applications multimédias reposant chacune sur des choix de conception différents (vis-à-vis de l'accès aux protocoles de Transport sous-jacents). Nous avons également défini comment permettre à ces applications d'accéder au système sans modification importante de leur code, une IHM graphique permettant d'injecter les paramètres de QdS. Parallèlement à ces tests, nous avons réalisé une étude préliminaire sur le principe de modules

d’adaptation, proposant aux programmeurs d’application des API *ad hoc* plus simples d’accès car intégrant des paramètres de QdS adaptés à chaque famille d’application.

2 Perspectives

Nous dégageons de ces travaux deux types de perspectives. Des perspectives immédiates dérivent directement de ces travaux et sont parfois déjà initiées. Des perspectives à moyen et plus long terme s’inscrivent dans la volonté d’étendre notre proposition d’architecture tant aux niveaux Transport et Application qu’au niveau des technologies réseaux.

2.1 Perspectives immédiates

Les perspectives immédiates des travaux présentés dans ce mémoire sont :

- d'étudier les problèmes de mise à l’échelle de notre système de communication, en particulier, vis-à-vis du nombre de connexions par BB nécessaires pour établir les réservations. En l’état actuel, les requêtes de QdS nécessitent qu’une connexion TCP soit maintenue entre deux équipements consécutifs. Il semblerait a priori plus judicieux d’utiliser pour certains PDU (par exemple pour les PDU relatifs au maintien des réservations) le protocole UDP. Cependant, ce changement ne pourra se faire qu’en reconsidérant les mécanismes actuels de traitements d’erreur (notamment ceux basés sur la détection de pertes de connexion) ;
- d’étudier l’impact du contrôle de congestion appliqué au niveau Transport sur notre modèle de caractérisation des services ;
- de finaliser l’étude et d’implémenter les modules d’adaptation qui permettent de proposer des API *ad hoc* pour chaque famille d’application ;
- de compléter le modèle de caractérisation des services en proposant non plus un algorithme de sélection basé sur une caractérisation unique des services mais sur une caractérisation intégrant d’autres critères pour prendre en compte l’état effectif de congestion du réseau à un instant donné ; par exemple, en considérant l’heure, le jour, … de la requête afin de se baser sur une modélisation plus réaliste ;
- d’étudier comment substituer (ou adapter) nos protocoles par (ou à) des protocoles standard (tels que COPS, BGRP, …), dont l’implémentation est disponible.

2.2 Perspectives à plus long terme

La première de ces perspectives concerne l’intégration dans notre système de communication des nouveaux services offerts par les protocoles de Transport comme DCCP, SCTP, FPTP/TFRC. En effet, la caractérisation de ces services lorsqu’ils sont mis en œuvre dans un environnement DiffServ n’est pour l’instant que très peu étudiée. Dans l’avenir, une souche du protocole FPTP intégrant les

nouveaux mécanismes de contrôle de congestion de type *TCP FRiendly Connection* (TFRC) sera disponible.

La deuxième de ces perspectives concerne l'intégration de protocoles applicatifs à notre système de communication, en particulier le protocole SIP, qui permettrait d'établir une première connexion entre les entités communicantes et de négocier des paramètres de communication de haut niveau.

La troisième perspective concerne la validation formelle de chacun des protocoles de l'architecture de signalisation. En effet, si une validation fonctionnelle de l'architecture a été effectuée en UML2.0, ses différents protocoles n'ont pas été validés individuellement.

La quatrième perspective concerne la prise en compte d'un contexte multicast 1 vers *m*. Plusieurs cas peuvent être étudiés dans ce cadre, du plus simple (mais déjà complexe !) considérant que chaque destinataire souhaite la même QdS, au plus évolué prenant en compte que les QdS réceptrices puissent être différentes.

La cinquième perspective concerne les problèmes de sécurité qui peuvent avoir un impact sur notre système de communication. Notons par exemple qu'à l'heure actuelle, aucune vérification sur l'origine (à part l'adresse IP) d'une requête ou d'une libération des ressources n'est faite.

La dernière de ces perspectives concerne la prise en compte, dans l'étude sur la caractérisation des services, de l'hétérogénéité des technologies physiques supports de la communication : réseaux de capteurs, sans fils (Wifi, Bluetoogth), GSM/GPRS, …, et de la généralisation de nos résultats dans un tel contexte multi technologies.

Bibliographie

La typographie employée dans cette bibliographie est la suivante :

– les références avec le nom en minuscule sont issues d'articles, revues, manuels,….
– les références avec le nom en petite majuscule sont issues de sites WEB.

Articles, revues, manuels

(Auriol, 2003) Auriol G, Larrieu N. « Introduction à l'utilisation de l'outil de simulation ns-2 », Séminaire LAAS-CNRS, Toulouse, février 2003

(Aurrecoechea, 1998) Aurrecoecha C., Campbell A.T., Hauw L., « A Survey of QoS Architectures », Multimedia Systems, vol. 6, num. 3, p138-151,1998.

(Banerjee, 2004) Banerjee S., Griffin T.G., Pias M., « The Interdomain Connectivity of PlanetLab Nodes », The 5th anuual Passive & Active Measurement Workshop PAM2004, Antibes Juan-les-Pins, France, avril 2004.

(Benameur, 2002) Benameur N., Ben Fredj S., Oueslati-Boulahia S., Roberts J.W., « Quality of service and flow level admission control in the Internet », *Computer Networks*, vol. 40, 2002, p57-71.

(Bianchi, 2002) Bianchi G., Blefari-Melazzi N., Femminell M., « Per-flow QoS support over a stateless Differentiated Services IP domain », *Computer Networks*, vol. 40, pp. 73-87, 2002.

(Blake, 1998) Blake S., Black D., Carlson M., « An Architecture for Differentiated Services », *RFC 2475*, 1998.

(Bolot, 1994) Bolot J.C., Turletti T., « A rate control scheme for packet video in the Internet », *Proceeding of IEEE Infocom*, 1994, pp.1216-1223.

(Bonald, 2000) Bonald T., May M., Bolot J., « Analytic Evaluation of REDPerformance », *INFOCOM'2000*,Tel Aviv, Israel, 2000

(Braden, 1997) Braden R., Zhang L., Berson S., Herzog S., Jamin S., « Resource ReSerVation Protocol », RFC 2205, septembre 1997.

(Braden, 1998) Braden R., Hoffman D., « RAPI – An RSVP Application Programming Interface », Internet Draft, août 1998.

(Campanella, 2001) Campanella M., Ferrari T., Leinen S., Sabatino R., Reijs V., « Specification and implementation plan for a Premium IP service », rapport d'avancement du projet GEANT, www.dante.org.uk/tf-ngn/GEA-01-032.pdf, avril 2001.

(Campbell, 1994) Campbell A.T., Coulson G., Hutchinson D., « A QoS architecture », *ACM Computer Com*. Review, 1994.

(Campbell, 1996) Campbell A.T., Coulson G., « Implementation and evaluation of the QoS-A transport system », Protocols for High Speed Networks, p201-218, 1996.

(Campbell, 1998) Campbell A.T., « QoA Architecture », Chapitre 3 dans Multimedia Communication Networks, Technologies and Services, 1998.

(Chassot, 1995) Chassot C., Fournier M., Lozes A., « Service Definition of a Multimedia Partial Order Connection », *2nd COST 237 workshop on Teleservices and Multimedia Communications,* Copenhagen, Denmark, novembre 1995.

(Chassot, 1996) Chassot C., Diaz M., Lozes A. « From the partial order concept to partial order connection », *Journal of High Speed Network (JHSN*), vol 5, n°2, 1996.

(Chassot, 2002) Chassot C., Garcia F., Auriol G., Lozes A., Lochin E., Anelli P., « Performance Analysis for an IP Differentiated Services Network », *ICC02*, New York, USA, Mai 2002.

(Cisco, 2004) Cisco Systems « Cisco 7600 Series, Solution and Design Guide – Metro Aggregation », Reference Guide, 2004.

(Clausen, 1999) Clausen H., Linder H., Collini-Nocker B., « Internet over Direct Broadcast Satellite », IEEE Communication Magazine, vol. 37, p146-151, juin 1999.

(Conrad, 1996) Conrad P.T., Golden E., Amer P.D., Marasli R., « A multimedia document retrieval system using partially-ordered/partially-reliable transport service », Proceedings of Multimédia Computing and Networking, 1996.

(Cristallo, 2002) Cristallo G., Jacquenet C. « An approach to inter-domain traffic engineering », Proceedings of XVIII World Telecommunication Congress (WTC2002), Paris, France, septembre 2002.

(Dantine, 1992) Dantine A., Baguette Y., Leduc G., Léonard L., « The OSI 95 Connection-Mode Transport Service – The Enhanced QoS », *HPN'92,* 1992.

(Diaz, 1993) Diaz M., Senac P., « Time Stream Petri Nets : a Model for Multimedia Stream Synchronization », First International Conference on Multimedia Modeling (MMM'93), Singapour, novembre 1993.

(DiffServ, 1998) Blake S., Black D., Carlson M., Davies E., Wang Z., Weiss W., « An Architecture for Differentiated Services », *RFC 2475*, décembre 1998.

(Doldi, 2003) Doldi Laurent « UML 2 Illustrated: developing real-time and communications systems », TMSO, ISBN 2-9516600-1-4, 316 pages, octobre 2003.

(Ebata, 2000) Ebata T., Takihiro M., Miyake S., Koizumi M., Hartanto F., Carle G., « Inter-Domain QoS Provisioning and Accounting », The 10th Annual Internet Society Conference (INET2000), Japan , juillet 2000.

(Eurescom, 1999) Eurescom « A Common Framework for QoS/Network Performance in a Multi Provider Environment », rapport de projet Eurescom P806, septembre 1999.

(Eurescom, 2000) Eurescom « Inter-operator interfaces for ensuring e2e QoS: State of the Art of IP Inter-Domain Management and Supporting Measurements », rapport de projet Eurescom P1008, juillet 2000.

(Eurescom, 2001) Eurescom « Inter-operator interfaces for ensuring end-to-end IP QoS: Specification of Inter-domain Quality of Service Management Interfaces », rapport de projet Eurescom P1008, mai 2001.

(Exposito, 2002) Exposito E., Gineste M., Peyrichou R., Sénac P., Diaz M., « XQoS : a quality of service specification language » IADIS International Conference 2002, Portugal, 2002.

(Exposito, 2003) Exposito E., « Spécification et mise en œuvre d'un protocole de transport orienté Qualité de Service pour les applications multimédias », thèse de Doctorat de l'Institut National Polytechnique de Toulouse, spécialité Réseaux et Télécommunications, décembre 2003.

(Fernandez, 2001) Fernadez M.P., De Castro A., Pedroza P., De Rezende J.F. « QoS Provisioning across a DiffServ Domain using Policy-based Management », Globecom 2001, novembre 2001.

(Floyd, 2003) Floyd S., « Profile for DCCP Congestion Control ID 3 : TFR Congestion Control », Internet Draft, 2003.

(Füzesi., 2003) P. Füzesi, K. Németh, N. Borg, R. Holmberg, I. Cselényi « Provisioning of QoS enabled inter-domain services » Computer Communications 26, page 1070-1082, 2003.

(Garcia, 2001) Garcia F., Auriol G., Chassot C., Lozes A., Lochin E., Anelli P. "Conception, implementation and evaluation of a QoS based architecture for an IP environment supporting differentiated services", *IDMS'01*, Lancaster, UK, Sept. 2001.

(Garcia, 2002) Garcia F., « Conception, implémentation et mesures des performances d'une architecture de communication de bout en bout à QdS garantie en environnement Internet de nouvelle génération », thèse de Doctorat de l'Institut National des Sciences Appliquées de Toulouse, spécialité Systèmes Informatiques, décembre 2002.

(Gopalakrishma, 1995) Gopalakrishna, Parulkar : « A Framework for QoS Guarantees for Multimedia Applications within an End system », *Tutorial at « Jahrestagung der deutschen Gesellschaft fr Informatik und der Schweizer Informatiker Gesellschaft (GI/SI 95) »*, 1995.

(Gross, 1992) Gross P. « Choosing a "Common IGP" for the IP Internet », RFC 1371.

(Hamma, 1997)Hamma S., Atmaca T., Piotr P., Czachorski T, « Modèle analytique du mécanisme partial buffer sharing pour des arrivées de paquets de taille variable », Rapport de Recherche LIP6, 1997.

(Handley, 1998) Handley M., Jacobson V., « SDP: Session Description Protocol », RFC 2327, avril 1998.

(Handley, 2000) Handley M., Perkins C., Whelan E., « Session Announcement Protocol », RFC 2974, octobre 2000.

(Heinanhen, 1999) J. Heinanhen, F. Baker, W. Weiss, and al. « An Assured forwarding PHB », *RFC 2597.*

(Hwang, 2003) U. Hwang, R. Revuru « Inter-domain Diffserv dynamic provisioning and interconnection peering study using Bandwidth Management Point - A Simulation Evaluation », The 2003 International Conference on Information Systems and Engineering (ISE 2003), Quebec, Canada juillet 20 - 25 2003.

(IntServ, 1997) J. Wroclawski, *RFC 2210*, Septembre 1997.

(ITU, 2001) ITU-T G.1010 « Quality of service and performance: End-user multimedia QoS categories », End-user multimedia QoS categories, novembre 2001.

(Jacobson, 1999) V. Jacobson, K. Nichols, K. Poduri. « An Expedited Forwarding PHB », *RFC 2598.*

(Jacquenet, 1999) Jacquenet C., « Providing Quality of Service Indication by the BGP-4 Protocol: the QOS_NLRI attribute », Internet Draft, juillet 1999.

(Jacquenet, 2004a) Jacquenet C., « A COPS Client-Type for Traffic Engineering », Internet Draft, février 2004.

(Jacquenet, 2004b) Jacquenet C., Cristallo G., « The BGP QoS_NLRI Attribute », Internet Draft, février 2004.

(Kohler, 2002) Kohler E., Handley M., Floyd S. « Datagram Congestion Control Protocol (DCCP) », Draft Internet, octobre 2002.

(Lochin, 2004) Lochin E. Anelli P., Fdida S., Garcia F., Auriol G., Chassot C., Lozes A. « Évaluation de la différenciation de services dans l'Internet », Revue TSI numéro spécial Services et Protocoles, 2004.

(Mantar, 2001) Mantar H.A., Hwang J., Iokumus I., Chapin S.J. « Inter domain Ressource Reservation via a Third Party Agent », The 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2001),USA, juillet 2001

(Mills, 1984) Mills D.L., « Exterior Gateway Protocol Formal Specification », RFC 904.

(Moffet, 1994) Moffet J.D., Clark J.A., « An Introduction to Security in Distributed Systems », High Integrity Systems, vol. 1, num. 1, 1994.

(Moy, 1998) Moy J., « OSPF Version 2 », RFC 2328.

(Nahrstedt,1996) Nahrstedt K., Smith J., « Design, Implementation and experiences of the OMEGA end-point architecture », *IEEE JSAC,* vol.14, 1996.

(Nichols, 1999) Nichols K., Jacobson V., Zhang L., « A Two-bit Differentiated Services Architecture for the Internet », RFC 2638 IETF, juillet 1999.

(Okomus, 2001) Okomus I.T., Hwang J., Mantar H.A., Chapin S.J., « Inter Domain LSP Setup Using Bandwidth Management Point », IEEE Globecom 2001 San Antonio, USA, novembre 2001.

(Owesarsky, 1996) Owezarsky P. « Conception et formalisation d'une application de visioconférence coopérative. Application et extension pour la téléformation », thèse de Doctorat de l'Université Paul Sabatier, décembre 1996.

(Owezarski, 1998) P. Owezarski, M. Diaz, C. Chassot. « A Time Efficient Architecture for Multimedia Applications » *IEEE JSAC*, April 1998, vol.16, n°3, pp 383-396.

(Pan, 2000) Pan P., Hahne E., Schulzrinne H., « BGRP : Sink-Tree-Based Aggregation for Inter-Domain Reservations », Journal Of Communications and Netxorks, Vol. 2, N. 2, juin 2000.

(PEI, 1992) Protocol Engines Incorporated, « XTP protocol definition », Revision 3.6., PEI 92-10, Mountain View, janvier 1992.

(QBone, 2001) Qbone Signaling WorkGroup « QBone Signaling Design Team », Final Report, 2001.

(Reichmeyer, 1998) Reichmeyer F., Ong L., Terzis A., Zhang L., Yavatkar R., « A Two-Tier Ressource Management Model for Differentiated Service Network », Internet Draft, novembre 1998.

(Rekhter, 1995) Rekhter Y., Watson T.J., Li T., « A Border Gateway Protocol 4 (BGP-4) », RFC 1771 mars 1995.

(Rizzo, 1997) Rizzo Luigi, « Dummynet: a simple approach to the evaluation of network protocols », ACM Computer Communication Review 27, 1, janvier 1997.

(Rojas, 1999a) Rojas L., Sénac P., Dairaine L., Diaz M. « Towards a new generation of transport services adapted to multimedia applications », *Annals of Telecommunications*, Tome 54, n°11-12, November-December 1999.

(Rojas, 1999b) Rojas L., Sénac P., Dairaine L. Chaput E., Diaz M., « Video Transport over partial order connections », *Computer networks*, Vol. 31, Issue 7, Elsevier, pp. 709-725.

(Rosenberg, 2002) Rosenberg J., Schulzrinne J., Camarillo G., Johnston A., Peterson J., Handley M., Schooler E., « SIP: Session Initiation Protocol »,RFC 3261, juin 2002.

(Salsano, 2001) Salsano S., Winter M., Miettinen N., « The BGP Plus Architecture for Dynamic Inter-Domain IP QoS », Intermon IST, août 2001.

(Schulzrinne, 1999) Schulzrinne H., Casner S., Frederick R., Jacobson V., « RTP: A Transport protocol for real time applications », RFC 1889, janvier 1999.

(Semeria, 2001) Semeria, Supporting Differentiated Service Classes ; Queue Scheduling Disciplines, White Paper, Junipers Networks Inc., janvier 2001.

(Senac, 1996) Senac P., « Contribution à la modélisation des systèmes mulitmédias et hypermédias », Doctorat, Institut National des Sciences Appliquées, Toulouse, juin 1996.

(Senac, 2001) Senac P., Exposito E., Diaz M., « Towards a new generation of generic transport protocols », 2001 Tyrrhenian International Workshop on Digital Communications (IWDC'2001), Taormina (Italie), 17-20 Septembre 2001, Lecture Notes in Computer Science 2170, Evolutionary Trends of the Internet, Eds.S.Palazzo, Springer, ISBN 3-540-42592-6, 2001, pp.492-506.

(Stewart, 2000) Stewart R., Xie Q., Morneault K., Sharp C., Scwarzbauer H., Taylor T., Rytina I., Kalla M., Zhang L., Paxson V., « Stream Control Transmission Protocol »,, *RFC 2960*, octobre 2000.

(Stewart, 2003) Stewart R., Ramalho M., Xie Q., Tuexen M., Conrad P « SCTP Partial Reliability Extension », Draft Internet, juin 2003.

(Terzi, 1999a) A. Terzis, J. Ogawa, S. Tsui, L. Wang, L. Zhang « A prototype Implementation of the Two-Tier Architecture for Differentiated Services », Fifth IEEE Real-Time Technology and Applications Symposium (RTAS99) Vancouver, Canada, juin 2-4, 1999.

(Terzi, 1999b) A. Terzis, L. Wang, J. Ogawa, L. Zhang « A two-tier resource management model for the Internet », Proceedings of Global Internet 99, décembre 1999.

(Trimintzios, 2001) Trimintzios P., Andrikopoulos I., Pavlou G., Cavalcanti C.F., Goderis D., T'Joens Y., Georgatsos P., Georgiadis L., Griffin D., Jacquenet C., Egan R., Memenios G., « An Architectural Framework for Providing QoS in IP Differentiated Services Networks », 7th IFIP/IEEE International Symposium on Integrated Network Management, 2001.

(Uruena, 2002) Uruena M., Larrabeiti D., Calderon M., Azcorra A., Kristensen J.E. and L.K., Exposito E., Garduno D., Diaz M.., « An active network approach to support multimedia relays », Joint International Workshop on Interactive Dsitributed Multimedia Systems / Protocols for Multimedia Systems (IDMS/PROMS),Portugal, novembre 2002.

(Ziegler, 2000) Ziegler T., Fdida S., Brandauer C., Hechenleitner B., « Stability of RED with two-way TCP traffic », ICCN, las Vegas, octobre 2000.

Sites Web

(@IRS, 2002) http://www.telecom.gouv.fr/rnrt/projets/res_3_ap00.htm

(AQUILA, 2000) http://www-st.inf.tu-dresden.de/aquila

(CADENUS, 2000) http://www.cadenus.org/

(GCAP, 2000) http://www.cadenus.org/

(ETHER, 1999) http://www.techfest.com/networking/lan/ethernet.htm

(LIP6, 2002) http://www-rp.lip6.fr/ns-2/

(TEQUILA, 2000) http://www.ist-tequila.org/

(TF-TANT, 1999) http://www.dante.net/tf-tant

(POLITO, 2002) http://netgroup-serv.polito.it/netgroup/hp/qos/index.html

(ULG, 2002) http://www.run.montefiore.ulg.ac.be

Bibliographie de l'auteur

Revues nationales

G. AURIOL, C. CHASSOT, M. DIAZ, « Architecture de communication à gestion automatique de la QdS en environnement IP à services différenciés », revue Technique et Science Informatiques (TSI), numéro courant, à paraître fin d'année 2004.

E. LOCHIN, P. ANELLI, S. FDIDA, F. GARCIA, G. AURIOL, C. CHASSOT, A. LOZES, « Évaluation de la différenciation de services dans l'Internet », revue Technique et Science Informatiques (TSI), numéro thématique « Réseaux et protocoles », vol. 23, n°5-6/2004.

Conférences internationales avec comité de lecture et actes

G. Auriol, C. Chassot, M. Diaz, « Toward a signalling architecture in a DiffServ multi-domain environment for a per flow guaranteed end to end QoS », Australian Telecommunication Networks and Applications Conference (ATNAC 2004), Swiss Grand Resort and Spa Bondi Beach, 8-10 décembre 2004.

G. AURIOL, C. CHASSOT, M. DIAZ « Architecture de communication à gestion automatique de la QdS et validation en simulation ns-2 », 10ème Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'2003), Paris (France), 7-10 Octobre 2003, pp.495-510.

C. CHASSOT, G. AURIOL, M. DIAZ « Automatic management of the QoS within an architecture integrating new transport and IP services in a DiffServ internet », 6th IFIP/IEEE International Conference on Management of Multimedia Netrworks and Services (MMNS'2003), Belfast (Irlande du Nord), 7-10 Septembre 2003 Lecture Notes in Computer Science 2839, Eds. A.Marshall, N.Agoulmine, ISBN 3-540-20050-9, Springer, 2003, pp.286-299.

C. CHASSOT, G. AURIOL, A. LOZES « QoS management protocol for an end to end communication architecture implemented over a differentiated IPv6 network », 18th International Teletraffic Congress (ITC'18), Berlin (Allemagne), 31 Août - 5 Septembre 2003, Vol.5b, pp.1251-1260.

F. GARCIA, G. AURIOL, C. CHASSOT, A. LOZES, E. LOCHIN, P. ANELLI « Conception, implémentation et mesure des performances d'une architecture de communication à QdS garantie dans un domaine IPv6 à services différenciés », 9ème Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'2002), Montréal (Canada), 27-30 Mai 2002, pp.381-394.

C. CHASSOT, F. GARCIA, G. AURIOL, A. LOZES, E. LOCHIN, P. ANELLI « Performance analysis for an IP differentiated services network », 2002 IEEE International Conference on Communications (ICC'2002), New York (USA), 28 Avril - 2 Mai 2002, pp.976-980.

Conférences nationales avec comité de lecture et actes

G. AURIOL « Architecture de bout en bout à QdS garantie dans un environnement DiffServ multi domaines : protocoles de signalisation », Colloque de l'École Doctorale Informatique et Télécommunications (EDIT'04), Toulouse (France), 29-30 Mars 2004, pp.177-181.

G. AURIOL, C. CHASSOT « Architecture de communication de bout en bout gérant la QdS dans un environnement DiffServ : étude en simulation ns-2 », Colloque de l'École Doctorale Informatique et Télécommunications (EDIT'03), Toulouse (France), 14-15 Avril 2003, pp.103-109.

G. AURIOL, C. CHASSOT « Conception d'un protocole de gestion de la Qualité de Service de bout en bout en environnement IPv6 à services différenciés », 5èmes Journées Doctorales Informatique et Réseaux (JDIR'2002), Toulouse (France), 4-6 Mars 2002, pp.31-40.

Rapport interne

G. AURIOL, C. CHASSOT, A. LOZES, M. DIAZ « QoS management mechanisms for an IP differentiated service network », Rapport LAAS N°02336, Août 2002, 14p.