Une Ingénierie d'Architecture pour une évolution trans-génération aux NGNs : Intégrations verticale et horizontale

Tarek Nadour

Thèse

présentée pour obtenir le grade de docteur
de l'Ecole Nationale Supérieure des Télécommunications

Spécialité : **Informatique et Réseaux**

Tarek NADOUR

Une Ingénierie d'Architecture pour une évolution trans-génération aux NGNs : Intégrations verticale et horizontale

Présentée le 13 décembre 2005 devant le jury composé de :

Pr. Bijan Jabbari (U. Georges Mason, USA)	Rapporteurs
Pr. Raouf Boutaba (U. Waterloo, Canada)	
Pr. Jean-Pierre Claudé (U. Versailles SQ, France)	Examinateurs
Mr. Antoine Boutignon (SFR R&D, France)	
Pr. Noémie Simoni (ENST Paris, France)	Directeur de thèse

Résumé

Le domaine des réseaux, au cœur des technologies de l'information et de la communication, a été marqué ces dernières années par la croissance et le développement explosif des demandes de communications et accompagné par l'évolution multidimensionnelle des offres à valeur ajoutées ainsi que la demande de plus en plus exigeante et variées des utilisateurs.

Devant ces évolutions de plus en plus rapides, les solutions proposées aujourd'hui traitent les problèmes qui surgissent dans un niveau architectural donné de manière isolée sans considérer l'architecture globale. En effet, les solutions du niveau réseau d'acheminement étudient l'évolution de l'infrastructure des plates-formes de connectivités (UMTS, xDSL, IP/MPLS, etc.) et les problématiques de QoS qui en découlent (comme la réservation de ressources réseau.) en ne considérant que ce niveau et le bout en bout transport. Il en est de même des solutions du niveau services applicatifs qui solutionnent le développement et le déploiement des plates-formes des services (RI, OSA, serveurs d'applications, etc.) et les problématiques de QoS qui en découlent (comme la réservation de ressources applicatives) en ne considérant que le niveau et le bout en bout applicatifs.

L'objectif de la thèse est de proposer une vue et une solution globale en mettant en œuvre un concept d'intégration globale des architectures de télécommunications en incluant tous les acteurs des niveaux superposés de la chaîne de bout en bout, à savoir, équipements matériels, le réseau de transport, les service et les usagers de ces services. Nous avons baptisé ce concept *Ingénierie d'Architecture* qui se base sur : (*i*) un modèle de *cohabitation* entre les niveaux pour s'assurer de la consistance de la structure d'ensemble, c'est-à-dire, que la superposition des blocs architecturaux entre les niveaux soit fonctionnellement complémentaire et non redondante pour la QoS de bout en bout, , et (*ii*) un modèle de *coopération* pour superviser et évaluer en temps réel la QoS offerte à travers les différents réseaux constituant les niveaux horizontaux (équipement, réseau, service ou usagers). L'objectif étant d'automatiser les processus de déploiement de nouveaux services et de mettre en œuvre la QoS demandé de manière efficace.

Pour le modèle de coopération du niveau réseau nous proposons un algorithme d'auto-organisation des éléments du réseau pour maintenir la QoS horizontale. Nous avons nommé cet algorithme Path-Based Clustering.

Pour la mise en œuvre de l'Ingénierie d'Architecture et des modèles associés (*cohabitation* et *coopération*), nous avons conçu une plate-forme baptisée Pilote Organisationnel (POrg) en définissant les trois dimensions suivantes : organisationnelle, fonctionnelle et architecturale. *(i)* La dimension organisationnelle qui définit le rôle, la portée et l'emplacement de chaque entité composant la plate-forme *POrg* dans l'architecture globale. Nous avons défini *POrg* comme une plate-forme d'agents distribués, potentiellement intégrés dans chaque nœud de l'architecture (routeurs, serveurs, proxies, etc.), activables à tous moments si besoin. Ils prennent en charge la déclinaison et l'agrégation de la QoS ainsi que la réorganisation des domaines. *(ii)* La dimension fonctionnelle qui définit l'ensemble des fonctions et actions à mettre en œuvre par *POrg* pour assurer l'ajout de nouveaux services et maintenir la QoS de bout en bout de manière automatisée. Pour la modélisation des traitements nous avons eu recours à UML : l'ensemble des fonctions des agents *POrg* est défini à travers des diagrammes des Use-Cases UML. *(iii)* Et la dimension architecturale qui définit la structure et la coordination des modules composant *POrg*. Par soucis de cohérence et de généricité, nous avons divisé ces modules architecturaux en deux grandes parties : La première partie est celle qui permet à l'agent de communiquer avec son environnement. La deuxième partie est le *Noyau* qui englobe tous les traitements internes et propres à l'agent. Cette séparation entre le traitement et la communication facilite la jonction entre un agent *POrg* et n'importe quelle architecture assurant une certaine évolutivité et adaptabilité de la plate-forme *POrg*.

L'implémentation d'un prototype *POrg* nous conforte sur la faisabilité et la pertinence de notre proposition.

Table des Matières

Table des figures

Liste des Tableaux

Table des Equations

INTRODUCTION GENERALE

Contexte d'étude

Le domaine des réseaux, au cœur des technologies de l'information et de la communication, a été marqué ces dernières années par la croissance et le développement explosif des demandes de communications et du trafic des données véhiculées par l'Internet, outil de communication incontournable aujourd'hui, économique et très puissant pour l'accès et le partage de l'information et de la communication à l'échelle mondiale. Il a permis d'évoluer vers la convergence des services de télécommunications induisant une nouvelle génération de réseaux et de services (NGN et NGS), ce nirvana évasif des télécommunications, vision d'une seule solution qui prend en charge les services réseau (VPN, fixe, mobile, xDSL, etc.) et les services applicatifs (voix, vidéo et données, RI, OSA/Parlay, etc.) de manière transparente et homogène, à n'importe quel moment et n'importe où.

Cette convergence a commencé à se faire ressentir à travers quatre (4) évolutions préoccupant les communautés académique et industrielle:

- augmentation significative du volume de trafic acheminé,
- évolution de la nature du trafic acheminé (des données vers la voix) et la vidéo devenant l'essentiel du trafic Internet.
- évolution des services à valeurs ajoutées qui deviennent de plus en plus innovants.
- et évolution des besoins utilisateurs eux-mêmes en termes de diversité et de personnalisation de services.

Devant ces évolutions de plus en plus rapides, des solutions ont été proposées, chacune essayant de faire face à plusieurs contraintes et défis qui apparaissent au fur et à mesure que

les besoins évoluent, sans oublier que chacune ajoute son coût de déploiement et de maintenance.

Par exemple, fournir une bande passante suffisante afin de répondre aux besoins des applications nécessitant une bande passante fixe à leurs flux, puis optimiser l'allocation de la bande passante (différentes règles d'ingénierie de trafic) pour les différents flux ou agrégat de flux, afin de répondre aux besoins de rentabilité des réseaux d'opérateurs, puis préserver le comportement des flux applicatifs qui sont de nature hétérogène (sporadiques, para-sporadiques, etc.) en proposant des architectures comme DiffServ et IntServ, installer des plates-formes de services applicatifs prenant en charge les besoins utilisateurs (SLA, etc.), etc. Chaque technologie et solution en cascade palliant chacune une partie du problème sans forcément considérer la vue globale de bout en bout.

Problématique

Le caractère ad-hoc des solutions proposées répondant à une partie du problème sans, forcément, considérer la vue de bout en bout, a conduit à des architectures hétérogènes, figées et peu évolutives.

En effet, afin de faire face aux évolutions de plus en plus rapides, les solutions proposées traitent les problèmes qui surgissent sur un niveau architectural donné sans considérer l'architecture globale. On peut noter des solutions du niveau réseau d'acheminement qui étudient l'évolution de l'infrastructure des plates-formes de connectivité (UMTS, xDSL, IP/MPLS, etc.) et les problématiques de QoS qui en découlent (différentiation des flux, routage, réservation de ressources réseau, etc.) en ne considérant que ce niveau et le bout en bout transport. Ou encore des solutions du niveau services applicatifs qui concerne le développement et le déploiement des plates-formes de services (RI, OSA, serveurs d'applications, etc.) et les problématiques de QoS qui en découlent (différentiation des messages, routage sémantiques, routage des messages, réservation de ressources applicatives, etc.) en considérant le niveau et le bout en bout applicatifs.

Ces vues isolées de l'architecture vont à contre courant des besoins de *réactivité* et de *flexibilité* exigées par les architectures d'aujourd'hui pour deux raisons :

La première raison est que l'organisation de l'architecture est faite de manière « overlay », c'est-à-dire que les deux niveaux coexistent sans interaction dynamique entre eux.

La deuxième raison est que le développement des services dépend d'une pile protocolaire pour chaque nouveau service, c'est-à-dire que pour chaque nouveau service, une pile protocolaire est développée. Cette solution est limitée dans le temps surtout en constatant que la tendance actuelle est vers la proposition de services personnalisés où chaque utilisateur pourra définir son propre service. Dans un tel environnement, la variété de services proposés peut être très importante et le cycle de vie des services peut être très court.

Pour répondre au besoin d'optimisation du Revenu Moyen par Utilisateur (ARPU), les opérateurs et fournisseurs de services doivent avoir les moyens leur permettant d'évaluer les impacts d'un niveau architectural (réseau, service, etc.) sur l'architecture globale, afin de faire face à l'évolution rapide des besoins à travers une rapidité de déploiement de nouveau services à valeurs ajoutées et la fourniture de ces services conformes aux attentes des utilisateurs (QoS de bout en bout).

Cette vision nécessite l'intégration de l'architecture des fournisseurs de service à tous les niveaux (les équipements matériels, le réseau de transport, les plates-formes de services -RI, OSA, SIP AS, etc.- et les plates-formes de prise en charge de la personnalisation des services [1]), et à toutes les vues, à savoir : l'intégration verticale, c'est-à-dire, qu'une décision ne peut être prise à un niveau sans une *cohabitation* avec les autres niveaux et sans pouvoir évaluer les impacts d'une telle décision sur les autres niveaux et notamment sur la perception des usagers finaux (QoS de bout en bout), et l'intégration horizontale, c'est-à-dire, que sur un niveau donné, quelque soit les sous-domaines traversés (exemple, pour le niveau réseau de transport, les sous-domaines peuvent être des sous-réseaux dans le sens adressages, des zones de routage dans le sens routage intra-domaine, des systèmes autonomes dans le sens routage inter-domaines, ou simplement une juxtaposition de réseaux hétérogènes (IP, ATM, Wifi, etc.)), une *coopération* de ces sous-domaines doit être maintenue afin que la QoS (horizontale) de ce niveau soit respectée.

Contributions

L'objectif global de cette thèse est de donner des clés à cette problématique en mettant en œuvre un concept d'intégration globale des architectures de télécommunications en incluant tous les acteurs superposés de la chaîne de bout en bout : équipements matériels, le réseau d transport, les service et les usagers de ces services. Nous avons baptisé ce concept *Ingénierie d'Architecture* qui se base sur des modèles de *cohabitation* et *coopération* pour rendre cette

superposition la plus transparente et réactive possible aux nouveaux besoins qui régissent le monde des télécommunications aujourd'hui, notamment les besoins aux nouveaux services personnalisés et à valeurs ajoutées qui par conséquent nécessitent le maintien de la QoS de bout en bout tout au long de leur cycle de vie.

Pour une intégration verticale, nos avons proposé un modèle de *cohabitation* qui s'assure de la consistance de la structure d'ensemble, c'est-à-dire, que la superposition des blocs architecturaux entre les niveaux soit fonctionnellement complémentaire et non redondante pour la QoS de bout en bout, l'objectif étant d'automatiser les processus de déploiement de nouveaux services et traduire leurs besoins en QoS. Notre approche repose sur deux principes : principe de la *déclinaison* de la QoS et principe de *l'agrégation* de la QoS.

- La déclinaison de la QoS fait référence au processus d'analyse de l'architecture globale de haut en bas (top-down) en visant deux objectifs :
    - trouver une solution pour une QoS de bout en bout,
    - maintenir la cohérence de la structure d'ensemble.
- L'agrégation fait référence au processus d'analyse de l'architecture globale de bas en haut (bottom-up), c'est à dire les couches supérieures doivent recevoir des informations de rétroaction des couches inférieures afin que les niveaux supérieurs s'assurent du bon déroulement de la déclinaison et éventuellement relancer le processus de déclinaison suivant ces informations de rétroaction.

Pour une intégration horizontale, nous avons proposé un modèle de coopération pour superviser et d'évaluer en temps réel la QoS offerte à travers les différents « sous-réseaux » (domaine) constituant le niveau horizontal (équipement, réseau, service ou usagers). Chaque domaine est autonome, c'est-à-dire, responsable de la supervision de sa propre QoS (SLA entre domaines). Si une dégradation de la QoS est constatée sur un (ou plusieurs) domaine(s), et donc une non-conformité au contrat de QoS, nous avons proposé de résoudre le problème en trois (3) étapes:

1. re-négocier le contrat de QoS (SLA) entre les domaines de telle sorte de rattraper la dégradation constatée dans le domaine *i* dans le domaine suivant (*i*+1).
2. changer l'itinéraire des flux traversant le domaine sur lequel on constate une dégradation pour les faire passer par un autre domaine qui aurait les ressources nécessaires aux besoins

du flux, ceci est assuré à travers l'interaction avec le protocole de routage en vigueur (routage réseau, routage applicatif, etc.).

3. réorganiser les domaines de telle sorte que des ressources soient disponibles de bout en bout horizontal.

Si ces trois étapes n'ont pas aboutit à une solution, un feedback négatif au niveau supérieur indiquant la non possibilité d'accomplir le contrat vertical pour qu'il prenne des décisions (modifier les besoins, réorganiser le niveau *N*+1, etc.) à travers le principe d'agrégation.

Afin de réorganiser les domaines de telle sorte que des ressources soient disponibles de bout en bout horizontal, nous avons proposé un algorithme de clustering basé sur la disponibilité des ressources dans les domaines. Nous l'avons appelé *Path-Based Clustering*.

Afin de concrétiser notre concept d'Ingénierie d'Architecture à travers ces deux modèles (*cohabitation* et *coopération*) et de les mettre en application, nous avons conçu une plate-forme d'ingénierie d'architecture baptisée Pilote Organisationnel (POrg) qui prend en charge l'architecture des opérateurs et fournisseurs de services de télécommunications, notamment, l'introduction de nouveaux services et le maintien de la QoS de bout en bout.

Afin de définir *POrg* le plus rigoureusement possible, nous l'avons spécifié suivant trois dimensions : organisationnelle, fonctionnelle et architecturale

- dimension organisationnelle qui définit le rôle, la portée et l'emplacement de chaque entité composant la plate-forme POrg dans l'architecture. Nous avons défini POrg comme une plate-forme d'agents distribués, potentiellement intégrés à chaque nœud de l'architecture (routeurs, serveurs, proxies, etc.), activables à tous moments si besoin. Ils prennent en charge la déclinaison et l'agrégation de la QoS ainsi que la réorganisation des domaines.

- la dimension fonctionnelle qui définit l'ensemble des fonctions et actions à mettre en œuvre par POrg pour assurer l'ajout de nouveaux services et maintenir la QoS de bout en bout de manière transparente. Par soucis de rigueur et d'objectivité, l'ensemble des fonctions des agents POrg est défini à travers des diagrammes d'Uses Cases d' UML.

- la dimension architecturale qui définit la structure et la coordination des modules composant *POrg*. Par soucis de cohérence et de généricité, nous avons divisé ces modules en deux grandes parties : La première partie est celle qui permet à l'agent de communiquer avec son environnement. La deuxième partie est le *Noyau* qui englobe

tous les traitements internes et propres à l'agent. Cette séparation entre le traitement et la communication facilite la jonction entre un agent POrg et n'importe quelle architecture, l'évolutivité et l'adaptabilité de la plate-forme *POrg*.

L'implémentation d'un prototype POrg nous conforte de la faisabilité de notre proposition.

Cadre de travail

Cette thèse s'est déroulée au sein de l'équipe AIRS (Architecture et Ingénierie des Réseaux et Services) du département Informatique et Réseaux de l'Ecole Nationale Supérieure des Télécommunications (ENST Paris). La spécification et le développement de la plate-forme Pilote Organisationnel (*POrg*) ont été effectués notamment dans le cadre du projet de recherche IA/PO en collaboration avec la direction R&D de la Société Française de Radio-télécommunication (SFR). Ce projet avait pour objectif d'étudier et mettre en œuvre les moyens (modèles, méthodes et plates-formes) permettant à un opérateur/ISP de rendre son architecture la plus flexible possible aux nouveaux services innovants proposés tout en assurant leur QoS. L'objectif étant d'optimiser le CAPEX et TTM.

Plan du document

Pour répondre à notre problématique, nous avons consigné nos travaux, nos réflexions et nos résultats à travers trois (3) parties.

Partie I : Evolution de la QoS des réseaux et services de télécommunication

Intéressés par une vision globale de la problématique de l'opérateur qui doit assurer la flexibilité et la réactivité de son architecture aux nouveaux besoins émergeants, notre contribution dans cette partie est d'analyser les différentes propositions qui essayent de satisfaire les besoins en termes de nouveaux services et de contraintes de QoS. L'objectif étant d'étudier ces solutions en termes de fourniture de QoS de bout en bout, d'optimisation du coût de déploiement et d'évolutivité. Nous avons divisé cette partie en trois (3) chapitres :

Chapitre 1 : Ce chapitre présent l'évolution architecturale des réseaux de télécommunication d'un point de vue opérateurs et fournisseurs de services. Afin d'avoir la vue la plus générale

possible, nous présentons l'évolution des réseaux à commutation de circuits et les réseaux à commutation de paquets.

Chapitre 2 : Afin de compléter l'état de l'art présenté dans le chapitre 1, nous présentons, dans ce chapitre les propositions et solutions qui prennent en compte les usagers et leur perception de la qualité des services offerts. Ces propositions concernent les solutions fonctionnelles et protocolaires de fourniture de QoS de bout en bout.

Chapitre 3 : Que doit-on ajouter à ces propositions pour atteindre nos objectifs ? Ce chapitre donne un aperçu des éléments de réponse en explicitant les objectifs à atteindre pour répondre à notre problématique, les objectifs atteints par les propositions existantes et les manques qu'il faut combler.

Partie II : A la recherche d'une solution …

Cette partie concerne la présentation de notre proposition pour résoudre notre problématique et les défis auxquels nous faisons face pour mettre en œuvre cette proposition. Nous avons divisé cette partie en deux (2) chapitres :

Chapitre 4 : Ce chapitre développe notre proposition du concept d'Ingénierie d'Architecture. Il présente notre modélisation de ce concept, et les défis auxquels nous avons fait face pour mettre en œuvre ce concept.

Chapitre 5 : Afin de mettre en œuvre le modèle d'ingénierie d'architecture, nous proposons dans ce chapitre la matérialisation de ce concept à travers une plate-forme baptisée Pilote Organisationnel (POrg). Afin d'être le plus rigoureux possible, nous définissons cette plate-forme à travers 3 dimensions : organisationnelle, fonctionnelle et architecturale.

Partie III : Simulations et expérimentations au service de l'ingénierie d'architecture

Cette troisième, et dernière, partie, est consacrée au développement du Pilote Organisationnelle ainsi qu'à la présentation des différentes plates-formes qui nous ont servi de support expérimental pour la définition des règles d'ingénierie utiles pour le Pilote Organisationnel. Cette partie est divisée en deux (2) chapitres :

Chapitre 6 : Ce chapitre présente la plate-forme de QoS que nous avons mise en place pour étudier le comportement des applications faces aux changements dans le réseau de transport.

L'étude de ces comportements nous permet d'extraire des règles dites d'ingénierie utilisées par la plate-forme POrg.

Chapitre 7 : Ce chapitre présente les détails d'implantation de POrg et les choix que nous avons effectué pour cette implémentation.

Conclusion générale

Cette partie conclut ce document en synthétisant les contributions essentielles de notre travail et en évoquant les perspectives qu'il ouvre.

Partie I EVOLUTION DE LA QOS DES RESEAUX ET SERVICES DE TELECOMMUNICATION

Introduction

Intéressés par une vision globale de la problématique de l'opérateur qui doit assurer la flexibilité et la réactivité de son architecture aux nouveaux besoins émergeants, notre contribution dans cette partie est de donner une vue sur l'évolution des propositions essayant de satisfaire au mieux les contraintes de QoS, d'innovation, d'optimisation des coûts de déploiement et de concurrence.

L'architecture d'un opérateur peut être vue comme un ensemble de ressources mises en place pour offrir un ensemble de services de télécommunication. Nous entendons par *service de télécommunication* « toute fonction mise à disposition d'un utilisateur par un fournisseur de service». Ceci inclut les services, dits, *réseau* (routage, réservation de ressources, contrôle d'accès, etc.) et les services dits applicatifs (transfert d'appel, compression, etc.). C'est l'évolution de ces services et des flux qui en découlent qui a piloté, ces dernières années, l'évolution des architectures supports vers une séparation des plans architecturaux que nous consignons et analysons dans le premier chapitre (§Chapitre 1).

La qualité d'un service rendu par un fournisseur de services, c'est-à-dire, la conformité du comportement de ce service au contrat souscrit (QoS), est également perçue par le client à travers l'approche de mise en œuvre des mécanismes de QoS par le fournisseur de services. Les mécanismes de mise en œuvre de la QoS, qui suivent l'analyse de l'évolution du premier chapitre, seront analysés dans le deuxième chapitre selon un filtre que nous proposons afin d'identifier leurs contributions à rendre l'architecture plus flexible et réactive aux nouveaux besoins (§Chapitre 2).

Reste alors à savoir, comme le souligne le troisième chapitre qui est consacré à l'analyse de toutes ces solutions (Chapitre 1 et Chapitre 2), si cela est suffisant et efficace pour la problématique que nous avons soulevée dans l'introduction.

Chapitre 1

Evolution des réseaux et services de télécommunication

L'objectif de ce chapitre est de donner une vue globale de l'évolution architecturale des réseaux et services de télécommunication et d'évaluer cette évolution notamment par rapport aux usagers. C'est l'évolution de ces services et des flux qui en découlent qui ont piloté, ces dernières années, l'évolution des architectures supports vers une séparation des niveaux architecturaux, à savoir, les services demandés et le réseau d'acheminement des flux de ces services. Cette séparation vient pour répondre aux nouveaux besoins en termes de services à valeurs ajoutées (*Value Added Services*) et de personnalisation de ces services par les utilisateurs.

Afin d'avoir la vue la plus globale possible, nous allons présenter notre vue de l'évolution des architectures de réseaux à commutation de circuits, et leur évolution vers l'architecture *Réseaux Intelligents* (§1.1) et les réseaux à commutation de paquets et leur évolution vers l'architecture *IP/MPLS* (§1.2). Cette étude ne se veut, évidement, pas exostive. L'objectif et de représenter les étapes les plus importantes de ces évolutions.

1.1. Services à commutation de circuits

Nous entendons par un service à commutation de circuit tout service visant à mettre en relation deux ou plusieurs interlocuteurs en vue d'échange d'informations parlées. Le support de transmission de la conversation (*bearer*) étant un réseau à commutation de circuits, même si le réseau de transmission de la signalisation[1] est un réseau à commutation de paquets. Le réseau téléphonique traditionnel (§1.1.1) a été l'un des premiers réseaux téléphoniques modernes[2] supportant le service de téléphonie où la logique du service et le traitement d'appel étaient câblés sur les commutateurs. Ce réseau a évolué vers un réseau où la logique du service a été programmable et non câblée sur les commutateurs (§1.1.2), les fonctions de traitement d'appels sont mises en liaison à travers un réseau de signalisation, séparant ainsi la logique du traitement d'appel et le réseau de signalisation du réseau de commutation, nommé « réseau sémaphore » (§1.1.3). Les nouveaux besoins en terme de rapidité de déploiement de nouveaux services à moindre coût et par n'importe quelle organisation (opérateur ou fournisseurs de service) a mené à la séparation de la notion de service du transport et à la naissance des réseaux intelligents (§1.1.4).

1.1.1 Plain Old Telephone Service (POTS/PSTN)

Un des premiers services de télécommunication existant (avant les années 60) est le service de téléphonie traditionnel (*Plain Old Telephone Service ou RTC*), mixant les flux voix et signalisations dans le même circuit [1], supportant ainsi tout le service téléphonique par le réseau. L'architecture du service POTS est illustrée dans la Figure I-1. Les abonnés sont reliés à un système de commutation d'appels.

Un système de commutation est composé de :

- une unité de contrôle : L'unité de contrôle est un composant matériel (câblé) exécutant la logique du service et donc qui fait le traitement d'appel.
- commutateur de données.

[1] Comme le cas des réseaux à canaux de signalisation commune (réseaux sémaphore) qui séparer physiquement le réseau de contrôle (signalisation) qui est un réseau à commutation de paquets du réseau de données qui est un réseau à commutation de circuits (voir §1.1.3 pour plus de détails).

[2] Où le traitement d'appel est automatique et pas à base d'opérateurs humains

Les systèmes de commutation d'appels sont reliés entre eux full mesh afin d'assurer la connectivité physique entre les abonnés distants.

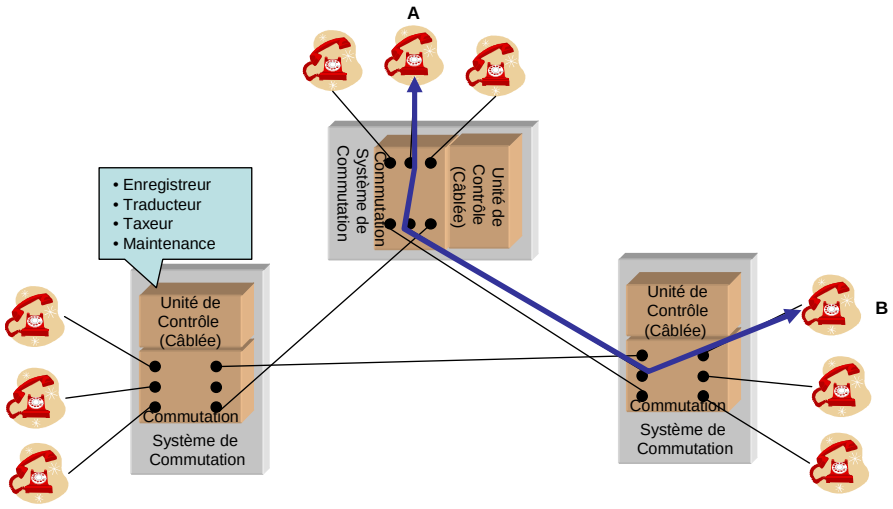


Figure I-1: Le RTC

Le service téléphonique POTS était supporté complètement par le réseau téléphonique, en l'occurrence, par les systèmes de commutations. De ce fait, la logique du service (Figure I-2) était câblée sur les systèmes de commutation. Typiquement, le service et le support étaient liés. Cinq fonctions caractérisaient la logique du service de téléphonie traditionnel, tous câblés sur le commutateur: Présélection, Traduction, Sélection, Taxation et Supervision

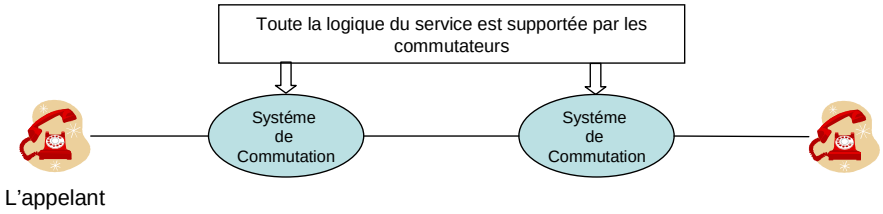


Figure I-2: Service téléphonique traditionnel

L'inconvénient majeur du réseau POTS est que les services et la logique de service étaient câblés dans les systèmes de commutation. De ce fait, l'ajout de nouveaux services fut très difficile. C'est là que les systèmes à base de programmes de contrôles stockés (SPC) ont vu le jour.

1.1.2 Réseaux à programmes de contrôle stockés (SPC)

Comme le service et la logique du service étaient câblés dans le système de commutation, le développement de nouveaux services comme le transfert d'appel et la mise en attente était très coûteux nécessitant le remplacement de tous les systèmes de commutation existants.

Les systèmes à base de *SPC* (*Stored Program Control*), que nous présentons dans cette section, furent, de ce fait, une avancée technologique importante.

En effet, au milieu des années 1960, les programmes de contrôle stockés (*Stored Program Control* : *SPC*) sont introduits [4], [5] pour la fourniture du service téléphonique. SCP est un terme large désignant un commutateur où l'Unité de Contrôle (voir §1.1.1) est programmable (non câblée).

Il y a quatre (4) éléments de base dans le système de commutation SPC : la matrice de commutation, la zone d'appel, la zone de programme et le processeur central.

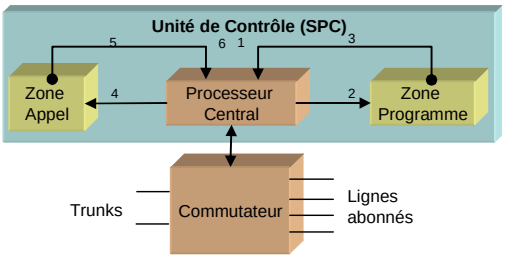


Figure I-3: Architecture simplifiée d'un SCP

A chaque détection d'un nouvel appel (action n°1 Figure I-3), le processeur de contrôle instancie le programme de traitement d'appel à partir de la « zone programme » (action n°3) et ouvre une zone mémoire pour recevoir le numéro d'appel (action n°4). Il y a autant d'instanciations simultanées et de zone mémoires d'appel que d'appels simultanés.

SCP fut une avancée technologique majeure car la logique du service n'est plus câblée sur les commutateurs mais programmable (Figure I-4). Par conséquent, il a été plus facile d'introduire de nouveaux services (reconfigurer les services). En effet, ils ont permis de palier le problème d'incorporation de la logique du service dans l'équipement de commutation directement. Néanmoins, le concept de la logique de service n'était pas « décorrélé » du service lui-même. Il était donc toujours aussi difficile d'introduire de nouveaux services à cause de la dépendance entre le service et la logique du service.

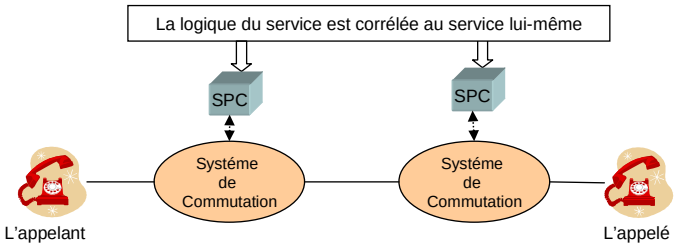


Figure I-4: Introduction du SPC

1.1.3 Réseaux à canaux de signalisation commune (Réseaux Sémaphore)

Bien que les SPCs aient apporté une certaine décorrélation du service du support de transmission ils furent limités par rapport aux évolutions des besoins en télécommunications. En effet, à partir des années 70, une nette évolution des besoins en télécommunication s'est fait sentir, avec l'apparition des nouvelles demandes de services [1]. Tous ces services sont différents les uns des autres. A cette diversité de besoins, s'ajoute une complexité croissante de sémantique des services.

Les solutions qui existaient (§1.1.1, §1.1.2) s'avéraient très limitées pour répondre à ces nouveaux besoins pour les raisons suivantes :

- Impossibilité d'interactivité durant la conversation. La signalisation dans les réseaux précédents se fait avant ou après l'établissement de l'appel. Il n'y a pas moyen de signaler des informations durant la conversation. Ceci est dû au multiplexage des messages de signalisation et de la voix sur le même réseau de transmission (signalisation In-band).
- Traitement coûteux des appels non aboutis : Dans la phase de sélection, les ressources sont réservées tout au long du chemin de l'appelant vers l'appelé même si l'appelant est occupé ou en dérangement.
- Contenu sémantique limité de messages de signalisation suite à un alphabet réduit.

Afin de prendre en charge les nouveaux besoins et faire face aux limitations des réseaux traditionnels, des systèmes de signalisation ont été proposés à partir des années 70 consistant à séparer physiquement le plan de contrôle (signalisation) et le plan de données et de multiplexer tous les messages de signalisation correspondant à une multiplicité de circuits sur un canal commun de transmission de données (Figure I-5). Cette signalisation est appelée « *signalisation par canal sémaphore* » et le réseau de transport associé cette signalisation (réseau à commutation de paquets) est appelé « *Réseau Sémaphore* » [1], [6].

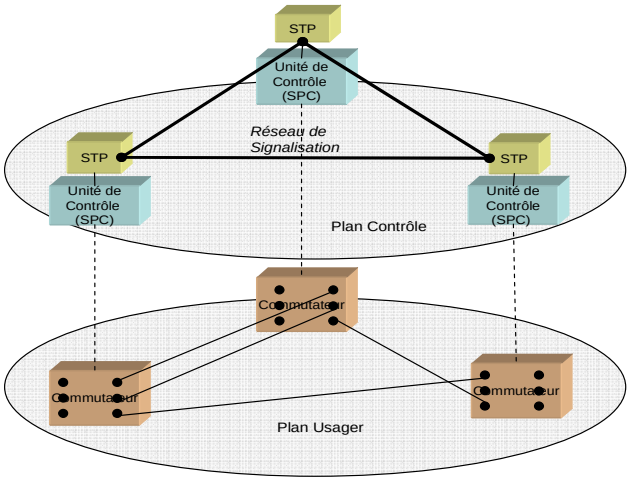


Figure I-5 : Séparation plan de contrôle (réseau sémaphore) et plan Usager (Commutateurs)

L'élément principal dans cette architecture est le *STP* (*Signaling Transfer Point*) qui représente le nœud de traitement dans le réseau sémaphore. Il est responsable de l'acheminement des messages de signalisation entre plusieurs Unités de Contrôle (*Points Sémaphore*).

En 1979, l'UIT-T a normalisé les réseaux sémaphore sous la norme « *Signalisation Sémaphore n°7 (SS7)* » en définissant ([7], [8]) :

- Le réseau de transmission des messages de signalisation (réseau sémaphore)
- Les protocoles de signalisation (applicatifs) sur le réseau sémaphore (TUP, ISUP, MAP et INAP)

L'ensemble du réseau sémaphore et les applications sémaphores sont structurés en couches comme illustré dans la Figure I-6.

Dans cette structure, le réseau commuté de transmission de données occupe les 3 premiers niveaux dont l'ensemble est appelé sous-système de transfert de messages MTP (Message Transfer Part). Les signalisations ISUP [12], [13], [14], [15], MAP, INAP, etc. constituent des applications au-dessus du MTP [9], [10] .

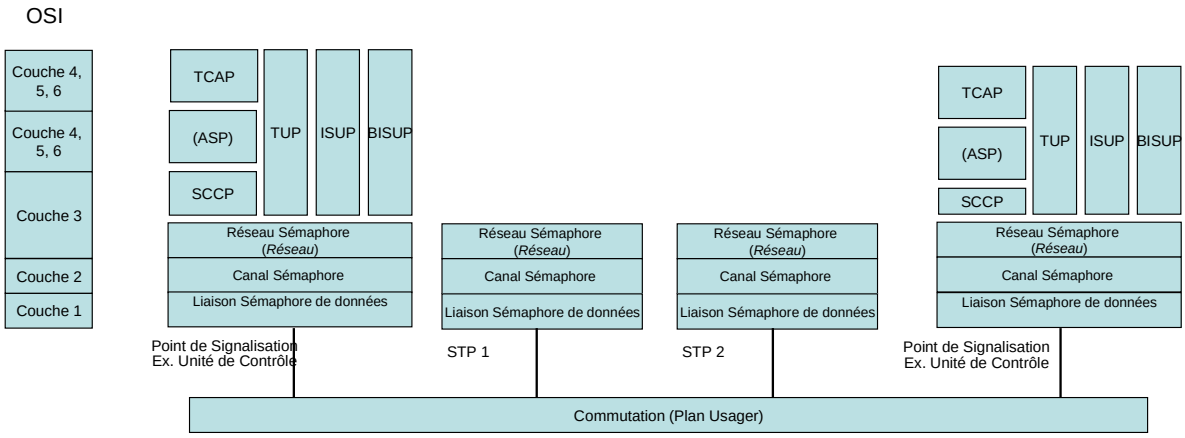


Figure I-6: Architecture Signalisation Sémaphore n°7 (SS7)

1.1.4 Réseaux intelligents (IN)

Au milieu des années 1980, un nouvel environnement de « déréglementation » des télécommunications est caractérisé par une mise en concurrence des opérateurs et l'apparition de fournisseurs de services tiers [16]. La conséquence de cette situation est une sorte de course à la fourniture de services nouveaux. Dans un tel environnement, une certaine « modularité » de l'architecture devait être assurée de telle sorte que les fournisseurs tiers puissent remanier la logique du service indépendamment du réseau support (opérateurs). En effet, les seuls services que pouvait fournir un réseau téléphonique traditionnel (solutions présentées ci-dessus) sont les services programmés dans le logiciel de traitement d'appel appartenant à l'opérateur. Afin d'obtenir un service d'un fournisseur tiers, il fallait, donc, séparer la notion de *service* de la notion d'*établissement de la connexion*. D'où la naissance du concept des *Réseaux Intelligents (RI)*. Il est défini dans la recommandation Q1290 comme un concept architectural pour la création et la fourniture de services de télécommunication [16].

Cette architecture est basée sur quatre éléments (Figure I-7) : les *SSPs* (*Signaling Switching Points*), les *STPs* (*Signaling Transfer Points*) présenté dans la section précédente (§1.1.3), les *SCP* (*Signal Control Points*) et les *SMPs* (*Service Management Points*).

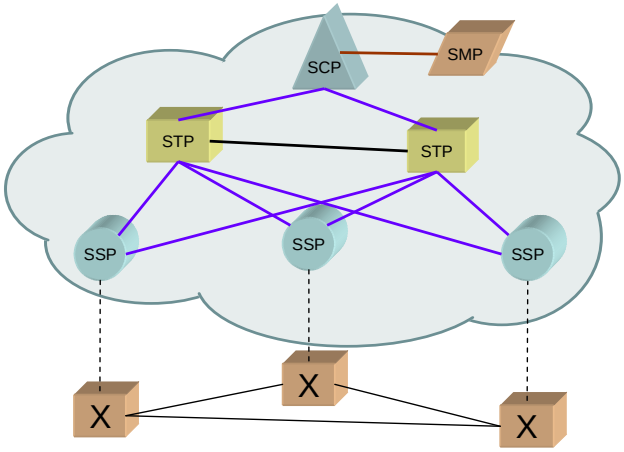


Figure I-7: Architecture du Réseau Intelligent

- SSP (Signaling Switching Point) : les SSPs sont des commutateurs de signalisation (centres de transit) traitant les demandes des abonnés (signalisations d'appels) pour les aiguiller vers le réseau sémaphore (vers les STPs) s'il s'agit de demande de services intelligents (numéros spéciaux, cartes prépayées, etc. en utilisant TCAP) pour vers le réseau téléphonique traditionnelle (vers le SSP destinataire en utilisant ISUP) s'il s'agit d'un appel « ordinaire ».
- STP (Signaling Transfer Point) : les STPs sont des commutateurs de paquets, ils jouent de passerelles entre les SSP et les SCP.
- SCP (Service Control Point) : les SCPs sont des stations (computer based) est exécute la logique du service. Les SCP sont en liaison avec les bases de données (profiles, facturation, etc.)
- SMP (Service Management Point) : les SMPs sont des service de gestion de services, réseau, utilisateurs, etc.

En vue de décrire les éléments du réseau intelligent, l'UIT-T a introduit un modèle conceptuel qui sert de cadre à la spécification et la description de l'architecture illustrée ci-dessus (Figure I-7). Il est définit à travers quatre (4) plans : plan service (SP), plan fonctionnel global (GFP), plan fonctionnel réparti (DFP) et plan physique (PP).

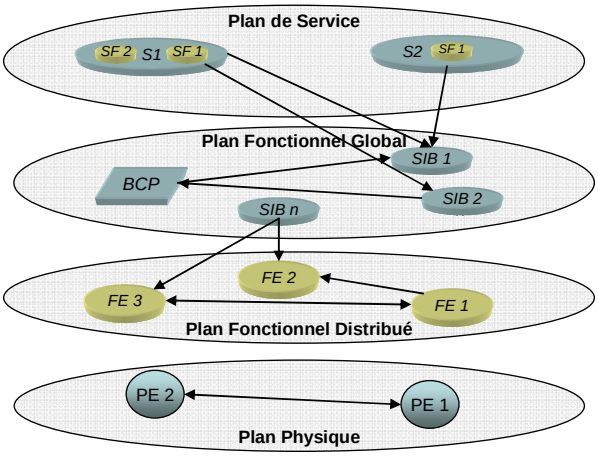


Figure I-8: Modèle Conceptuel du Réseau Intelligent

- Plan de Service: Ce plan contient les descriptions des services tels qu'ils peuvent être vus par un usager du service (numéro vert, etc.).
- Plan Fonctionnel Global: Ce plan modélise la fonctionnalité du réseau intelligent comme un seul réseau. L'ensemble des fonctions est défini par des *SIBs* (*Service Independent Buildings*) qui représentent le traitement élémentaire effectué.
- Plan Fonctionnel Distribué: Ce plan modélise le réseau intelligent comme un ensemble d'entités fonctionnelles réparties qui exécutent des actions (*FEA, Functional Entity Action*).
- Plan physique: Il modélise les différents éléments physiques supportant le réseau intelligent. Les différentes entités fonctionnelles réparties sont localisées dans les différentes entités physiques.

Nous avons vu l'évolution des services à commutation de circuit, nous allons avoir dans ce qui suit les services à commutation de paquets.

1.2. Services à commutation de paquets

Nous entendons par le service à commutation de paquets tout service manipulant comme flux utilisateurs des données utilisant comme support de transmission un réseau à commutation de paquets. Nous présentons dans cette section l'évolution de ces réseau à travers la description des réseaux: X.25 (§1.2.1.), puis Frame-Relay (§1.2.2.), puis IP (§1.2.3.), ATM (§1.2.4.) et finalement MPLS (§1.2.5.).

1.2.1. Le réseau X.25

La norme X.25 a été spécifiée suivant les constats faits à l'époque que (a) les équipements utilisateurs étaient limités en intelligence, et que (2) les systèmes de communication étaient non fiables [19]. A partir de ces suppositions, X.25 a été développé pour prendre en charge les équipements communicants inintelligents en supportant tous les services tels que le contrôle des flux et vérification des erreurs. Ainsi, la conception du réseau X.25 est un système orienté-connexion où les liaisons entre les extrémités communicantes étaient réservées tout au long du chemin. Il s'agissait de reproduire le même schéma que celui utilisé dans la commutation de circuit et de l'adapter à la commutation de paquets. Cependant, la notion de circuits virtuels est le fondement de l'architecture X.25.

La spécificité du réseau à commutation de paquets X.25 par rapport au réseau téléphonique traditionnel est qu'un DTE (*Data Terminal Equipment*) peut communiquer avec plusieurs autres DTEs simultanément. Ceci est possible grâce à la notion de circuits virtuels. Un circuit virtuel est une liaison logique bidirectionnelle créée pour assurer une communication fiable entre deux DTEs. X.25 prévoit deux types de circuits : Circuit Virtuel Permanent, Circuits Virtuels Commutés.

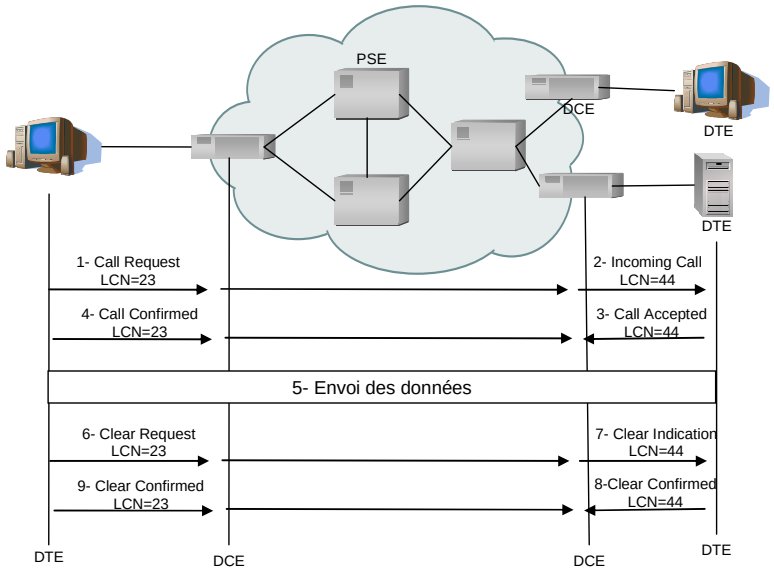


Figure I-9: Architecture et composants d'un réseau X.25

Services X.25

La recommandation X.25 est structurée en trois (3) couches (Figure I-10), elles résident dans les trois premières couches du modèle OSI. Chaque couche définit un protocole spécifique :

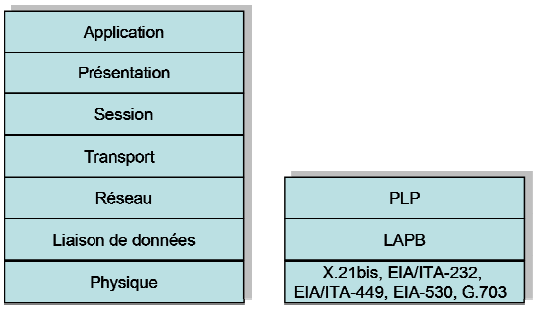


Figure I-10: X.25 et le modèle OSI

- Couche physique : elle définit la signalisation physique et connexion entre le DTE et le DCE. Plusieurs standards sont proposés : X.21bis, EIA-232 et G.703.
- LAPB (Link Access Procedure, Balanced) est le protocole de liaison de données qui gère les communications et la mise en trames des paquets entre le DTE et le DCE. LAPB est un protocole orienté flux (octets). Les services rendus par cette couche (protocole) incluent [19]: FCS (Frame Checking Sequence), retransmission, contrôle de flux, détection et correction des erreurs, acquittements, ouverture et fermeture de connexions et error reporting.
    - PLP est le protocole X.25 de niveau 3. Il gère les échanges de paquets entre les DTE à travers les circuits virtuels.

De ce fait, X.25 offre une panoplie de fonctions qui ne sont pas toute forcément requises par les applications. De plus, des fonctions comme la retransmission est présente aussi bien dans LAPB que dans PLP. Ces fonctions génèrent elles-mêmes un trafic de contrôle qui pouvait atteindre jusqu'à 12% du trafic total. Ceci est appliqué pour toutes les applications de la même façon sans laisser une « liberté » aux applications d'extrémité de personnaliser ces fonctions suivant leurs besoins. Ces inconvénients furent les raisons principales pour lesquelles *Frame-Relay* a été proposé.

1.2.2. Frame-Relay

Contrairement à X.25, *Frame-Relay* est spécifié à partir des principes que (*a*) les équipements d'extrémité sont assez intelligents pour supporter les services comme le contrôle d'erreurs, intégrité des données, etc. ; (*b*) les infrastructures physiques (équipements et liens) sont assez

fiables et ce n'est pas optimal de consommer d'avantage de ressources pour corriger les erreurs. Frame-Relay est orienté-connexion [19]. Cependant, la notion de circuits virtuels (permanent et commuté) est le fondement de l'architecture Frame-Relay comme X.25.

Services Frame Relay

Frame-Relay élimine tous les services offerts par X.25 en adaptant deux services principaux : notification de congestion et rejet sélectif des trames (Figure I-11) :

- *Circuits virtuels*
- *Notification de congestion* : FR offre le service de notification des congestions et laisse le contrôle effectif des flux par circuit virtuel aux extrémités (contrairement à X.25). Il implémente deux mécanismes de notifications de congestion : BECN (*Backward-Explicit Congestion Control*) et FECN (*Forward-Explicit Congestion Control*).
- *Rejet sélectif* : FR adopte l'approche de rejeter des trames afin d'éviter les problèmes de congestion. L'approche actuelle utilisée par FR est d'implémenter un bit de marquage DE (Discard Eligibility) (Figure I-11) indiquant, lorsqu'il est à 1, qu'en cas de congestion, cette trame est « plus éligible » au rejet que les autres trames.
- *Détection des erreurs* : FR utilise CRC (Cyclic Redundancy Checking) comme mécanisme de détection des erreurs de transmission. FR se contente de détecter les erreurs et de les signaler aux couches supérieures (utilisateurs) pour prendre qu'ils prennent des décisions.

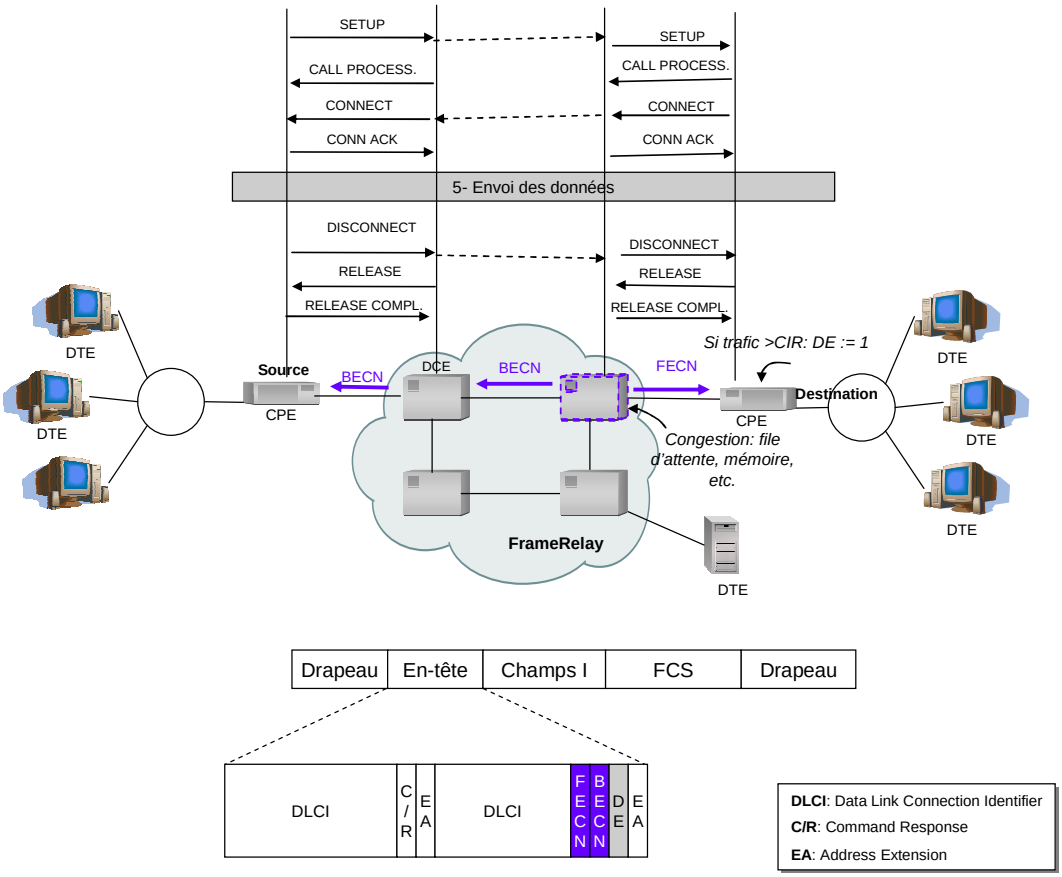


Figure I-11: Notification des congestions et format de trames Frame-Relay

1.2.3. Internet Protocol (IP)

Le développement de protocole de l'internet a été initié au milieu des années 70, quand la DARPA (*Defense Advanced Research Projects Agency*) était intéressés par l'établissement d'un réseau à commutation de paquet qui facilite la communication entre des ordinateurs de différents systèmes [20]. Devant l'hétérogénéité des différents réseaux reliant ces ordinateurs, la DARPA a fondé un groupe de recherche avec l'université de Stanford et BBN (Bolt, Beranek and Newman). Le résultat de ces efforts fut la suite des protocoles Internet (TCP/IP).

Services IP

Afin de rendre transparent l'hétérogénéité des réseaux de connectivité, IP préconise la suppression des services réseau offerts par *Frame-Relay* et *X.25* et les remplace par deux services primaires :

- délivrance des paquets en mode *non-connecté* et *au-mieux* (*best-effort*) à travers le réseau en utilisant un schéma d'adresses de taille fixe. IP traite chaque paquet comme une entité

à part indépendante de tous les autres paquets. Par conséquent, il n'y a pas de notion de connexion ou circuit logique (virtuel ou autre).

- *fragmentation* et *défragmentation* des paquets afin de s'adapter aux différents réseaux sous-jacents avec différentes tailles de MTU (Maximum Transmission Unit).

De ce fait, tous les services relatifs au traitement des paquets (correction d'erreurs, contrôle de flux, numérotation, etc.) se feront aux extrémités communicantes identifiées par une adresse de taille fixe.

IP et la QoS

IP n'assure aucune garantie de livraison des paquets, et encore moins des garanties de livraison suivant des contraintes de QoS. Bien que la notion de QoS n'ait pas été explicitement prévue dans le monde IP, le besoin de différencier le traitement des paquets IP a été considéré très tôt dans la conception du protocole IP, et le TOS est prévu pour cet usage [18]. Mais la difficulté de mise en œuvre, le manque de clarification des besoins, la vocation non commerciale du premier Internet et son usage quasi exclusif pour le transport de données ont longtemps contrarié cette intention. Les réseaux IP se sont, donc, contentés d'assurer les services de base du niveau 3 avec une politique best-effort laissant tous les services relevant de la garantie de QoS aux extrémités utilisant IP. Ce n'est que plus récemment, depuis l'entrée de l'Internet dans le monde des opérateurs commerciaux que les ambitions naissantes de prise en charge des applications multimédia de nature temps réel par ces opérateurs ont amené à remettre l'ouvrage sur le métier. D'où la proposition de modèles cherchant à copier d'une certaine façon le mode orienté connexion et la naissance de l'*ATM* (*Asynchronous Transfer Mode*) (§1.2.4.) dans le monde de l'IP.

1.2.4. Asynchronous Tranfer Mode (ATM)

L'idée de réaliser un réseau puissant avec une architecture analogue à X.25 et Frame-Relay, susceptible de prendre en charge les applications multimédia, a vu le jour vers les années 80 [21]. De là est né le protocole ATM et sa cellule, d'une longueur constante de 53 octets (48 octets de données et 5 octets d'entête)[3].

[3] La longueur de la zone de données, de 48 octets, est le résultat d'un compromis passé entre les Européens, qui souhaitaient 32 octets, et les Américains, qui désiraient 64 octets [21].

ATM a été développé dans un souci de transport de flux de différentes natures, avec des contraintes différentes. C'est ainsi que cinq catégories de niveaux de services ont été définies par l'ATM forum [22] : CBR, rt-BMR, nrt-VBR, ABR et UBR:

- CBR (Constant Bit Rate ou Deterministic Bit Rate): pour le trafic temps réel à débit constant,
- rt-VBR (real-time Variable Bit Rate): pour le trafic temps réel à débit variable,
- nrt-VBR (non-real-time Variable Bit Rate): pour le trafic sans contraintes de temps,
- ABR (Available Bit Rate) : pour les applications type « best-effort » demandant un contrôle de flux,
- UBR (Unspecified Bit Rate) : pour le trafic nécessitant des débits crêtes sans contrôle de flux.

IP sur ATM

Afin d'associer le monde Internet, et notamment IP, à ce protocole, deux modèles « Peer Model » et le « Overlay Model » sont apparus au milieu des années 90.

- *Peer Model*

Dans ce modèle, la couche ATM est considérée comme un nœud IP et propose d'utiliser le même schéma d'adressage comme pour IP pour les nœuds de bordures du réseau ATM. Ce modèle a été rejeté parce que, bien qu'il simplifie le schéma d'adressage des éléments finaux, il complique la conception des commutateurs ATM en exigeant que les commutateurs doit avoir, en plus des fonctions d'un commutateur ATM, les fonctions d'un routeur IP.

- Overlay Model

Dans ce modèle, la couche ATM est vue comme une couche de liaison de données sur laquelle IP s'exécute. Un réseau ATM, dans ce modèle, a son propre schéma d'adressage et protocoles de routage. L'espace d'adressage ATM d'est pas logiquement couplé avec celui d'IP et il n'a pas de mappage arithmétique entre les deux.

Afin de palier à cette complexité, l'évolution de l'Internet a vu apparaître des solutions de commutation multicouches (multi-layer switching) où les routeurs et les commutateurs sont intégrés et possède la même topologie du système.

1.2.5. MultiProtocol Label Switching (MPLS)

Les premières solutions de commutation multicouche, comme IP-over-ATM *Peer Model* ont donc maintenu les composants de contrôle conventionnels de l'IP, et ont utilisé la permutation d'étiquette de la couche liaison (un couple VPI/VCI) comme composante d'acheminement.

En plus de la séparation des composants de contrôle des composants d'acheminement, une caractéristique fondamentale commune à toutes ces solutions de commutation multicouche consiste en l'utilisation d'un algorithme de permutation d'étiquette (label swapping).

Mais ces solutions étaient propriétaires et la plupart d'entre elles devaient refaire appel à un transport ATM parce qu'elles ne pourraient pas fonctionner au-dessus d'infrastructures média mixtes (Frame Relay, PPA, SDH et LANs). Si la commutation multicouche devait être largement déployée par les fournisseurs de services, il devait y avoir un standard multi-constructeurs c'est pourquoi, un nouveau groupe de travail, MPLS WG [25], fût établi en 1997 pour produire un standard de commutation multicouche unifié et interopérable.

Le principe fondamental de MPLS consiste donc à acheminer les paquets selon des chemins prédéfinis en fonction d'une étiquette (label) et non de les router saut après saut à base de l'en-tête du paquet [26], [27]. MPLS fonctionne donc sur le mode VC comme ATM bien MPLS commute des paquets et pas des cellules.

1.2 Conclusion

Dans ce chapitre, nous avons soulevé les points qui nous paraissent important dans l'évolution des services proposés et la manière de déployer ses services du le réseau de transport. Nous avons relevé que la tendance architecturale dans la proposition des services est vers la séparation entre les services et le réseau d'acheminent, la séparation entre la signalisation et les données ainsi que la séparation entre la gestion et la signalisation. Nous avons identifié aussi que la motivation de cette tendance est, en plus de la rapidité de déploiement de nouveaux services de plus en plus diversifiés, la flexibilité et dynamicité de l'architecture.

Cette analyse motive la première question que nous nous posons dans cette thèse qui est : Est ce que la séparation des plan architecturaux est suffisante pour atteindre l'objectif de flexibilité et réactivité d'une architecture ? Des éléments de réponse seront donnés dans le Chapitre 3.

Chapitre 2

La QoS dans les réseaux et services de télécommunication : Supports existants

Le premier chapitre était consacré à l'analyse de l'évolution des réseaux et services de télécommunication du point de vue opérateur. Cette analyse nous a permis de dégager les tendances vers lesquelles les fournisseurs de services de télécommunication conçoivent les nouvelles architectures. Cette tendance architecturale est vers une séparation des plans et niveaux architecturaux. Mais comment on a pris en compte la QoS de ces nouvelles architectures? En d'autre terme, comment la perception des usagers de ces services est prise en compte dans cette évolution ?

Dans ce chapitre, nous analysons les différentes solutions mettant en œuvre les moyens qui permettent aux usagers de percevoir les services demandés conformément au contrat prescrits. L'ensemble de ces solutions suivent le courant de l'évolution des réseaux et services de télécommunications (§Chapitre 1). Deux approches sont possibles aujourd'hui : faire supporter la QoS par le réseau ou faire supporter la QoS par le service (application). La première approche préconise la mise en œuvre des mécanismes dits de qualité de service au niveau du réseau de transport (§2.1). La deuxième approche suppose que le réseau n'est pas en mesure d'assurer la QoS nécessaire aux flux des utilisateurs et essaie, donc, d'améliorer la QoS perçue en « corrigeant » les dégradations causés par le réseau (§2.2). Dans chacun des niveaux cités ci-dessus, nous distinguons deux vues : une vue *horizontale* où l'approche consiste à assurer la QoS de bout en bout en se base sur une mise en relation entre deux ou plusieurs composants appartenant au niveau considéré (réseau ou service). La deuxième vue étant une vue *verticale* où l'approche se base sur la mise en œuvre de mécanismes agissant de manière ponctuelle (sur le nœud d'entrée au réseau par exemple) suivant un besoin de niveau applicatif (application par exemple) et sans recourir à une collaboration entre composants.

2.1 Solutions de niveau réseau

2.1.1 Propositions verticales

La QoS au niveau réseau couvre un ensemble de mécanismes verticaux de délivrance des paquets avec des niveaux de service garantis ou différenciés. Les deux modèles utilisés pour assurer la QoS dans le niveau réseau sont, le Service Intégré (Integrated Service), et le Service Différencié (Differentiated Service).

Le Service Intégré (architecture IntServ)

La philosophie derrière le service intégré est que les nœuds du réseau (routeurs) doivent pouvoir réserver des ressources pour chaque flux afin de fournir une QoS garantie pour les flux utilisateurs. Le *Service Intégré* supporte trois classes de service [28] : a) la classe « Best effort », b) Service *Garanti* et c) service *Charge contrôlée*.

Le service *Garanti* [29] garantit que le délai maximum des datagrammes soit borné. L'application donne la taille et la fréquence de ses bursts, ainsi que son débit crête. Le flux peut être refusé, mais si le routeur l'accepte, le routeur offre une véritable garantie qu'un délai maximum sera respecté pour sa traversée.

Le service *Charge Contrôlée* [30] est prévu pour soutenir une large classe d'applications qui sont sensibles au débit en leur assurant un minimum de bande passante exigée. L'application donne la description de son trafic (Traffic Specifications, T-Specs) en même steps que sa demande de QoS. Le routeur garanti que la majeure partie du trafic aura un délai proche du délai demandé.

Le *Service Intégré* est assuré par différentes composantes architecturales:

- Contrôleurs ponctuels dans les nœuds du réseau (par exemple CAC, ordonnanceur, policer, etc.), ces composants sont implémentés dans chaque nœud à travers le réseau;
- Méthodes qui permettent aux applications d'exprimer et de communiquer leur besoin en termes de QoS (par exemple Unix RAPI [31], Winsock2 QoS API [32]).

Le service différentié (architecture DiffServ)

Dans ce modèle, la QoS n'est plus traitée pour chaque flux mais plutôt pour un agrégat de flux. Il supporte les exigences de divers flux d'applications en employant un mécanisme de classification [33], [34]. L'idée fondamentale derrière le modèle de services différenciés est de

transporter l'information de QoS dans le paquet ou dans l'en-tête de la trame. La différentiation de service peut être appliquée dans la couche trois [34] aussi bien que la couche deux [35] du modèle OSI:

- *Différentiation du service couche-3:* Le champ Type Of Service (TOS) et le champ Differentiated Service (DS) dans l'en-tête du paquet IP sont utilisés pour indiquer le besoin de QoS en termes de délai, de débit ou de pertes. Les routeurs de bordure sont responsables de la classification, marquage et ordonnancement des flux. Exemple de tels nœuds est le Provider Edge Router (PE) et le SGSN *(Serving GPRS Support Node)* [36]).
- *Différentiation du service couche-2***:** Le champ *Tag* dans les trames de la couche IEEE MAC détermine la priorité d'accès de cette trame au support de transmission. Le champ *priorité* dans la couche MAC (3 bits) définit huit niveaux de priorité [35].

  La première tentative de traiter la QoS dans la couche MAC de façon normalisée apparaît dans la version originale de IEEE 802.1D, qui définit l'architecture pour des ponts et les commutateurs de niveau 2. La version 1998 de 802.1D (désigné sous le nom de 802.1p dans la littérature) introduit la notion de classes de trafic dans lesquelles un buffer est réservé pour chacune des huit classes de trafic. Token Bus (802.4) et Token Ring (802.6) utilisent le même nombre de classes de trafic.

2.1.2 Propositions horizontales

La mise en relation des nœuds du réseau pour une QoS de bout en bout réseau est un moyen mettant en œuvre le processus de routage et le processus de réservation de ressources. L'ensemble de ces moyens rentre dans le concept d'Ingénierie de Trafic (IT). L'IT est le processus de contrôle de la façon dont l'ensemble du trafic traverse un réseau afin d'optimiser la consommation des ressources et les performances du réseau tout en supportant les flux exigeant une certaine QoS. L'IT s'occupe donc essentiellement de l'optimisation du «layout» et, donc, la réalisation d'un bon compromis entre les deux objectifs contradictoires suivants:

- Assurer la QoS : Il s'agit de répondre conformément aux attentes des flux entrants en terme de QoS (Délai, *Fiabilité, Disponibilité et Capacité, DFDC*) ;
- Utiliser efficacement les ressources du réseau : Cet objectif est indispensable à un opérateur dans le contexte commercial et concurrentiel, de manière rentabiliser au maximum ses investissements.

Routage supportant la QoS

Le processus de routage est un moyen de communiquer la disponibilité des ressources réseau à des fins de calculs de routes de bout en bout satisfaisant les besoins QoS d'un flux (ou agrégat de flux). L'ensemble des protocoles de routage réseau peuvent être divisés en deux catégories avec différents objectifs et visibilités: Inter et Intra domaines.

- *Routage Intra-domaine*

Les protocoles de routage Intra-domaines sont déployés au sein d'un réseau appartenant à une seule autorité administrative, dénommé *Système Autonome (AS)* (Figure I-13). Les protocoles les plus connus utilisés dans l'Internet aujourd'hui sont OSPF (Open Shortest Path First) [39] et ISIS (Intermediate System to Intermediate System) [40]. OSPF (ISIS) est basé sur l'algorithme SPF (Shortest Path First) de Dijkstra [41]. L'échange des informations de routage se fait par la diffusion des « Link State Advertissments (LSAs) » par chaque routeur vers son voisinage. Chaque routeur OSPF maintient une base de données topologique du *Système Autonome*. A partir de cette base, chaque routeur calcule une table de routage en construisant un arbre à plus court chemin en se basant sur des poids administratifs affectés aux liens sans notion de QoS. OSPF met à jour la base de données topologique suivant les changements dans la topologie du réseau (lien défectueux, routeur en panne, etc.). Plusieurs extensions viennent pour compléter OSPF de base pour prendre en compte les besoins QoS des flux traversant le réseau (bande passante et délai). Nous citons parmi eux : Q-OSPF (extensions QoS pour OSPF) [42] et OSPF/ISIS-TE (extension d'OSPF/ISIS pour l'ingénierie de trafic) [43], [44].

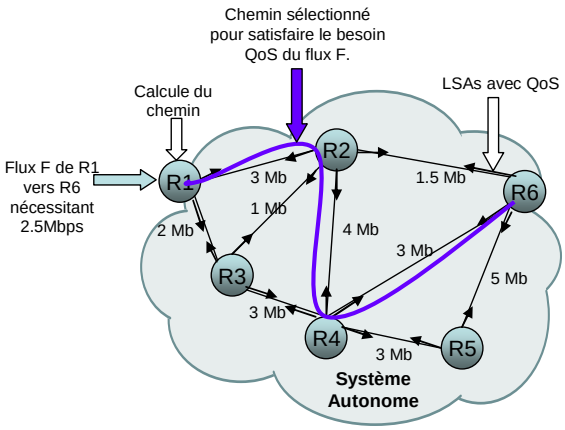


Figure I-12: Routage Intra-domaine avec QoS

- Q-OSPF : Q-OSPF se base sur l'extension des LSAs de base par ajout d'information de bande passante disponible et le délai du lien. Cependant, le calcul des chemins ne considère que la bande passante disponible [42].
- OSPF-TE : Contrairement à Q-OSPF qui essaie d'ajouter des attributs aux LSA existants, OSPF-TE améliore OSPF de base en se servant d'une nouvelle classe de LSA, dénommée *Opaque-LSA* [45]. Un *Opaque-LSA* est composé de l'en-tête d'un LSA standard suivi de quatre (4) octets transportant des informations propres à une application donnée. OSPF-TE personnalise les informations sur 4 octets pour transporter les informations de TE sous forme de TLV (Type/Length/Value). Ces informations incluent les informations de QoS comme bande passante maximale disponible et bande maximale de réservation ainsi que des informations de politique comme la classe du lien qui sert à favoriser un certain type de trafic à passer par un lien donner

- *Routage Inter-domaine*

Les protocoles de routage Inter-domaine sont déployés pour l'interconnexion de systèmes autonomes (Figure I-13). Un protocole de routage Inter-domaine vise à communiquer les informations d'accessibilité des réseaux afin de sélectionner la meilleure route vers chaque destination (AS) en fonction des *politiques* spécifiées par chaque domaine. Le protocole de routage Inter-domaine n'est au courant que des interconnexions entre des domaines distincts; il n'a aucune information sur le détail de ces domaines (nœuds intra-domaine). Le protocole de routage inter-domaine le plus connu est le *Border Gateway Protocol* (*BGP*) [46].

BGP est un protocole à vecteur de chemins ou les informations échangées entre les routeurs BGP (annonces) contiennent une séquence des systèmes autonomes traversés par ces annonces. L'algorithme à vecteur de chemin est une généralisation de l'algorithme à vecteur de distance dans le sens où BGP utilise les séquences des AS pour détecter les boucles. L'annonce d'une route, qui est formalisée à travers des messages UPDATE, indique l'accessibilité à un réseau. Ces informations échangées entre les nœuds BGP sont des informations de nature *politique* (contrairement à OSPF qui échange des informations de nature *métrique*) sous forme d'attributs. Ce sont des grandeurs décrivant les caractéristiques d'un réseau distant sur lesquelles se base l'algorithme de calcule des routes [47]. BGP définit plusieurs attributs : *ASPath, LocalPreference, NextHop, Multi-exit discriminator,* etc.

La notion de QoS n'était pas prise en compte par BGP de base (protocole de base pour l'Internet d'aujourd'hui). Cependant, plusieurs travaux se sont focalisés sur l'extension de BGP pour supporter la sélection de chemins suivant des contraintes de QoS.

Bonaventure [47], [48] s'est focalisé sur la distribution flexible des informations de QoS par BGP dans différentes topologies de réseaux. Le projet tequila [49] propose un nouvel attribut pour les messages BGP, QoS_NLRI, pour transporter les informations de QoS. Abarbanel et Venkatachalam [50] utilisent BGP pour distribuer les informations d'ingénierie de trafic qui représentent l'abstraction (résumé) de l'état de l'AS (bande passante, délais, etc.).

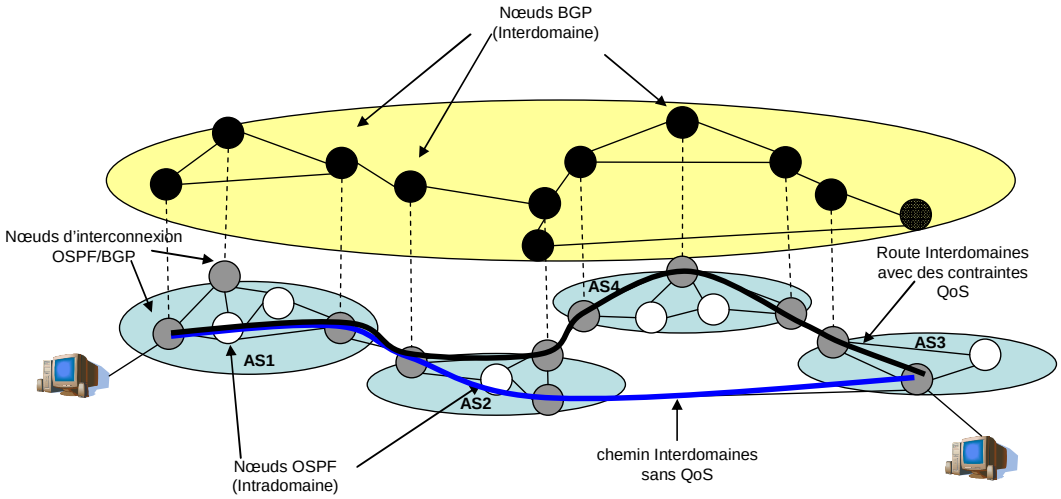


Figure I-13:Intra-domaine vs. Inter-domaine

Réservation de ressources

La deuxième « famille » de propositions horizontales pour assurer un niveau de service conforme aux besoins des flux est à travers un processus de réservation de ressources réseau. La notion de ressource est large, elle peut désigner aussi bien une bande passante qu'une étiquette (MPLS par exemple) ou interface.

Le protocole le plus connu pour la réservation de ressources (bande passante) au réseau est *RSVP* (*Resource Reservation Protocol*) [52]. C'est un protocole de réservation de ressources de bout en bout pour flux unidirectionnels en se basant sur deux types de messages : *Resv* et *Path* (Figure I-14).

Une station réceptrice envoie un message *Resv* à la station émettrice. Ce message traverse tous les nœuds du chemin de proche en proche, à chaque nœud, l'adresse du nœud traversé est insérée dans le message jusqu'à l'arrivée (source). La source envoie alors un message *Path*

pour finaliser la réservation. Ce message suit le même chemin que le message *Resv*. RSVP maintient l'état du chemin réservé à travers l'envoie périodique des messages *Resv* et *Path* entre les nœuds. L'état de la réservation maintenu par RSVP peut être changé à n'importe quel moment à travers l'envoie d'un message *Path* de changement d'état (demander plus ou moins de bande passante). Plusieurs autres protocoles de réservation de ressources réseau ont été proposés comme YESSIR [53] et BOOMANG [54]. Afin de supporter la distribution des labels MPLS, RSVP a été étendu pour l'ingénierie de trafic (RSVP-TE) [55].

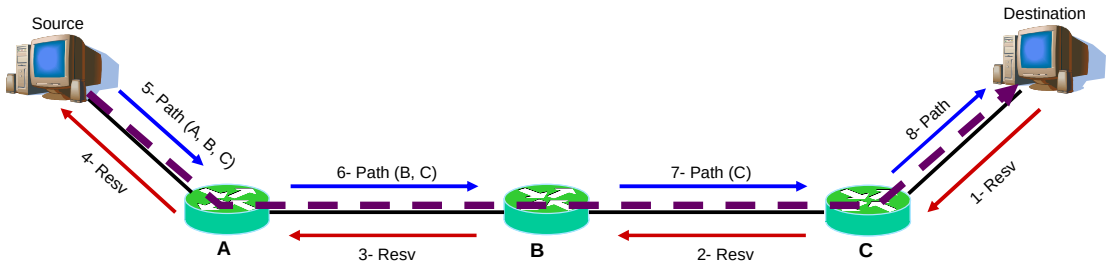


Figure I-14: Le protocole RSVP (Solution horizontale de niveau réseau)

2.2 Solutions de niveau service/application

Dans la section précédente (§2.1) nous avons vu que les solutions proposées font supporter toute la QoS sur le réseau de transport. Ces méthodes nécessitant la mise à jour de l'implémentation des nœuds du réseau pour mettre en œuvre les mécanismes et moyens décris ci-dessus. Avec l'avènement de l'Internet, réseau IP natif ne garantissant aucune qualité de service, ces applications se doivent elles-mêmes assurer cette QoS. Dans cette section, nous présentons les différentes méthodes utilisées par les applications actuelles pour atteindre cet objectif. Nous classifions ces méthodes suivants les vue verticale (§2.2.1) et horizontales (§2.2.2).

2.2.1 Propositions verticales

Gestion de Tampons

L'impact des variations de la gigue de délai produis par la traversée du réseau peut être réduit par des mécanismes de gestion de tampons (Figure I-15). Il s'agit de mécanismes de stockages d'une partie des données reçues afin d'ajuster le point de diffusion et ainsi amortir l'impact de la gigue sur la QoS perçue [55].

Correction d'erreurs

Il s'agit de mécanismes fondés sur la reconstitution, à destination, des informations contenues dans les perdus. Cette reconstitution nécessite l'ajout de redondance au flux de données. L'approche principale de mécanismes de compensation des pertes est l'approche FEC (*Forward Error-correcting Codes*) [58] qui est fondée sur la reconstitution, à la destination, des informations contenues dans les paquets perdus. Il existe deux classes de mécanismes de correction d'erreurs : *Media-Independent Correction* et *Media-Dependent Correction.*

- *Media-Independent Correction*

Ce schéma est indépendant du obtenu des flux de données, il calcul à travers algorithme d'encodage la redondance ajoutée au flux de données. Cette redondance permettra à la réception de recalculer les paquets éventuellement endommagés ou perdus. Les schémas d'encodage les plus connus sont les codes *Reed Solomon* [59] et les codes fondés sur les matrices de *Cauchy* [60]. L'utilisation de ces schémas de correction d'erreur a été normalisée par l'IETF en les incluant dans les messages RTP [61].

- *Media-Dependent Correction*

Ce mécanisme est fondé sur la représentation hiérarchique de la qualité d'une source (qualité du codage des données). En utilisant deux niveaux de qualité par exemple, deux représentations de la source à émettre sont construites. La première représentation (S_1) atteint une qualité (D_1) pour un débit (R_1) et la seconde (S_2) atteint représentation atteint une qualité (D_2) pour un débit (R_2). L'idée de base est de permette d'extraire la version dégradée dans le cas où la version originale est indécodable. Dans ce schéma, un paquet $n + 1$ contient, en plus de la représentation de premier niveau (S_1), une représentation de deuxième niveau des informations contenues dans le paquet *n*. l'utilisation de ce schéma de correction a été normalisée par l'IETF [62].

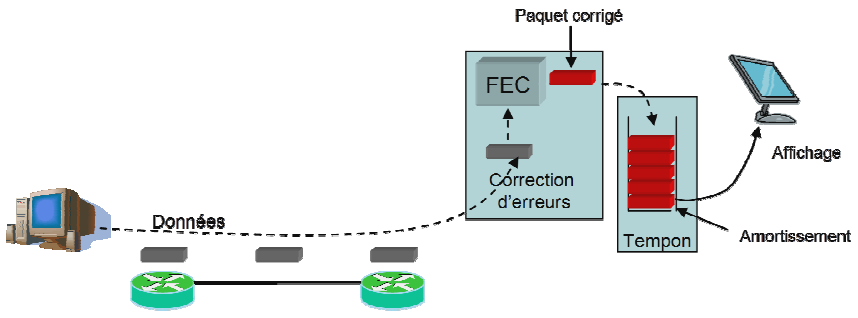


Figure I-15: Solutions verticales de niveau service

2.2.2 Propositions horizontales

La mise en relation de deux ou plusieurs éléments applicatifs (éléments de service) est un autre moyen envisagé par les services actuels pour assurer une QoS conforme aux besoins des flux applicatifs. Ces moyens peuvent être classifiés en trois catégories : adaptation des extrémités, routage et réservation de ressources

Adaptation des extrémités (contrôle de congestion)

Ces solutions s'apparentent au processus adaptatif par rapport à l'état du *réseau* impliquant les nœuds de service (applicatif) d'extrémité. Dans cette configuration, le contrôle de la QoS se fait aux extrémités communicantes. Le principe que nous avons pu déduire de ces méthodes est celui d'un rétrocontrôle (*feedback*) entre émetteur et récepteur. Dans une telle approche, les extrémités surveillent l'état du réseau et s'échangent des informations de QoS (que nous qualifions d'informations de gestion). Ces informations sont utilisées pour réguler le débit d'émission et, éventuellement, changer codage des flux utilisateurs (audio, vidéo ou données). Deux approches sont à citer, approches se basant sur le délai d'aller-retour RTT (*Round-Trip Time*) et autre utilisant le protocole RTP/RTCP (*Real Time Protocol/ Real Time Control Protocol*).

- RTT : Cette méthode part du principe que les files d'attente des éléments réseau (routeurs), pouvant conduire à une congestion du réseau, vont entraîner un accroissement du RTT [63]. Un mécanisme d'observation des variations de RTT implémenté au niveau de l'élément de service (émetteur) permet une réaction de ce dernier et ainsi éviter la perte de paquets.
- RTP/RTCP : Contrairement à la première méthode qui ne modifie pas le paquet applicatif, cette méthode ajoute un en-tête au paquet IP (en-tête RTP) an ajoutant des marqueurs temporels et des numéros de séquence des paquets transportés permettant au récepteur de reconstruire la base de temps et ainsi le flux applicatif [61], [64]. Le récepteur, quant à lui, fournit périodiquement un retour d'information sur la qualité de réception (gigue, pertes, etc.) par le biais du protocole RTCP [63]. Ces informations sont exploitées par la source pour ajuster le débit d'émission ou adapter le type de codage au niveau de ressources disponibles.

Routage applicatif

Une nouvelle classe de schémas de routage et d'adressage désignés sous le nom de Routage et Adressage basés sur le contenu (*Content-based routing and addressing*) est entrain d'émerger dans le domaine des systèmes distribués [65], [66]. Cette approche diffère du schéma de routage réseau dans le fait que la décision de routage des messages applicatifs entre éléments de service est basée sur les données transportées dans ces messages plutôt que sur un adressage spécialisé attaché, ou associé, au message.

Utilisant ce schéma de routage, les destinataires potentiels déclarent leurs besoins à travers des prédicats de sélection. Les expéditeurs potentiels publient leurs messages, c'est ainsi que les destinataires des messages sont déterminés.

Ce schéma de routage est largement utilisé dans les systèmes à notification d'événements et Publish/Subscribe [67] afin de fournir les données appropriées aux clients suivants leurs besoins. Il présente deux avantages principaux :

- Par rapport aux fournisseurs de contenus : ils ne sont pas obligés de maintenir une liste des clients potentiels et il suffit juste d'injecter des messages dans le réseau (applicatif)
- Par rapport aux clients : ils peuvent définir le type et la tailles des données qu'ils veulent recevoir et ainsi limiter le volume du trafic entrant.

Plusieurs systèmes de routage applicatif ont été proposés comme Gryphon [68], Siena [69], Jedi [70], XRoute [71] et *i*Bus [72].

Réservation de ressources applicatives

Pour les applications multimédia distribuées, exigeant une négociation de la QoS et de réservation de ressource pour les modules répartis de l'application, une classe de protocoles de réservation de ressources applicatives est entrain d'émerger. Une ressource applicative est une notion abstraite incluant aussi bien les numéros de ports, buffers, frames et points de diffusion. Le protocole de réservation de ressources applicatives le plus connus est NRP (Negociation and resource Reservation Protocol) [73].

NRP est un protocole assurant la négociation basée sur des gammes de valeur de QoS indiquées par le client et la disponibilité des ressources dans les systèmes d'extrémité (finaux) et des liaisons applicatives.

NRP est exécuté par un ensemble d'agents protocolaires (PA) (Figure I-16). Chaque entité de l'application (composants et liens) est matérialisée à travers un agent PA ou les Pas sont interconnectés suivant la topologie des composants de l'application.

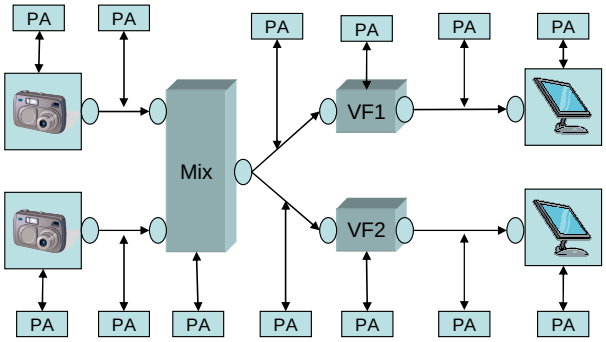


Figure I-16 : Architecture NRP : Exemple d'application multimédia

L'architecture NRP nécessite que chaque composant fournisse une interface générique de négociation. Cette interface est composée d'un ensemble de méthodes où chaque méthode correspond à l'une des quatre (4) méthodes suivantes :

- *Reserve* : pour réserver les ressources à la source. Les ressources sont présentées sous forme d'ensemble de résolutions d'image, frames/s, …supportés par le composant
- *PropagateBack* : pour l'envoie et la réception des messages NRP
- *Relax* : pour sélectionner, parmi les l'ensemble des ressources réservées, celle qui convient le mieux pour cette session (après négociation de tous les composants).
- *Free* : pour indiquer à un composant la terminaison d'une session de réservation.

2.3 Conclusion

Dans ce chapitre, nous avons fait un balayage des différentes méthodes et moyens permettant aux usagers des services de télécommunication de percevoir leurs services conformément à leurs attentes exprimées par un contrat de QoS. Nous avons vu que deux approches superposées sont dominantes dans les solutions d'aujourd'hui: faire supporter toute la QoS par le réseau de transport sans considérer les applications ou l'inverse, c'est-à-dire les applications supportent toute la QoS en supposant que le réseau est incapable d'assurer une partie de cette QoS. Dans les deux cas de figure, nous avons classifié les solutions proposées en deux catégories : solution verticales et solutions horizontales. Ces deux filtres nous permettent d'analyser (dans le Chapitre 3) la portée des solutions proposées et poser la deuxième question qui motive notre thèse qui est : Est ce qu'on maîtrise cette superposition ?

En d'autre terme, est ce qu'on a la possibilité d'évaluer l'impact d'un choix architectural (activer la correction d'erreur au niveau applicatif par exemple) sur le comportement de l'architecture globale identifié par sa QoS perçue de bout en bout? Des éléments de réponse seront fournis dans le chapitre suivant.

Chapitre 3

Analyse et discussion

Dans les deux chapitres précédents, nous avons fait un tour d'horizon de l'évolution de la QoS des réseaux et services de télécommunication pour cerner notre problématique. Nous nous sommes efforcés de présenter un état de l'art à travers une vision des moyens et solutions proposés pour absorber l'évolution des services de télécommunication accompagnée de l'évolution des besoins des usagers de ces services. Nous avons vu que la tendance est vers la séparation des vues horizontales, c'est-à-dire, une vue réseau qui engobe le moyen de transport des flux applicatifs, et une vue du service supporté ou rendu par ce moyen de transport. Nous avons vu aussi que, suite à cette évolution, le respect du contrat de QoS est confié à un niveau architectural donné sans forcément considérer son impact sur l'architecture globale et sans s'assurer que la superposition des blocs architecturaux entre les niveaux soit fonctionnellement complémentaire et non redondante pour la QoS de bout en bout. Dans ce chapitre, nous discutons ces différentes propositions suivant la problématique exposée dans l'introduction. Pour ce faire, nous analysons les propositions suivant les trois (3) filtres : Support de QoS de bout en bout (§3.1), temps et coût de déploiement des services (§3.2) et l'évolutivité de ces solutions (§3.3). Nous terminons ce chapitre par une conclusion où nous introduisons notre proposition pour compléter ces propositions et répondre aux trois critères (§3.4).

3.1 Support de QoS de bout en bout

Le premier critère d'évaluation des solutions proposées est le support de la QoS de bout en bout. C'est-à-dire l'aptitude de ces propositions à assurer la conformité du comportement des flux usagers suivant le contrat souscrit. Mais que recouvre le bout en bout (§3.1.1)? Et dans les propositions existantes, quels éléments impactant la QoS de bout en bout ont été pris en compte (§3.1.2)?

3.1.1 Quel bout en bout ?

Dans les propositions que nous avons présentées, deux « bout en bout » sont considérés. Il y a tout d'abord le bout en bout qui couvre le réseau d'acheminement (§2.1), champs d'action de l'ingénierie de trafic (§2.1). La QoS du réseau d'acheminement est impactée par la mise en relations des éléments de service (applications) que l'on peut offrir aux utilisateurs. Ces éléments de service forment le deuxième niveau de bout en bout considéré par les propositions existantes (§2.2). Mais est ce que ces deux niveaux forment la chaîne complète du bout en bout ?

La Figure I-17 illustre notre première vue des acteurs de la chaîne impactant la QoS de bout en bout. Le premier niveau à considérer pour compléter les propositions existantes est les plates-formes matérielles. Ces équipements physiques, c'est-à-dire, les nœuds et les liens physiques, composants de l'infrastructure du réseau qui héberge les modules logiciels de l'architecture (commutateurs, routeurs, …) assurant la transmission des flux réseau les traversant par la capacité de leurs ressources (CPU, mémoire, interfaces) sont un élément très important à prendre en compte dans la chaîne de bout en bout.

Les services offerts à un utilisateur potentiel

Les résultats présentés dans la section II.1 montrent que les plates-formes matérielles de connectivité et les plates-formes d'accès possibles aux services sont, donc, des tronçons à prendre en compte pour avoir une maîtrise de la chaîne de bout en bout (Figure I-17).

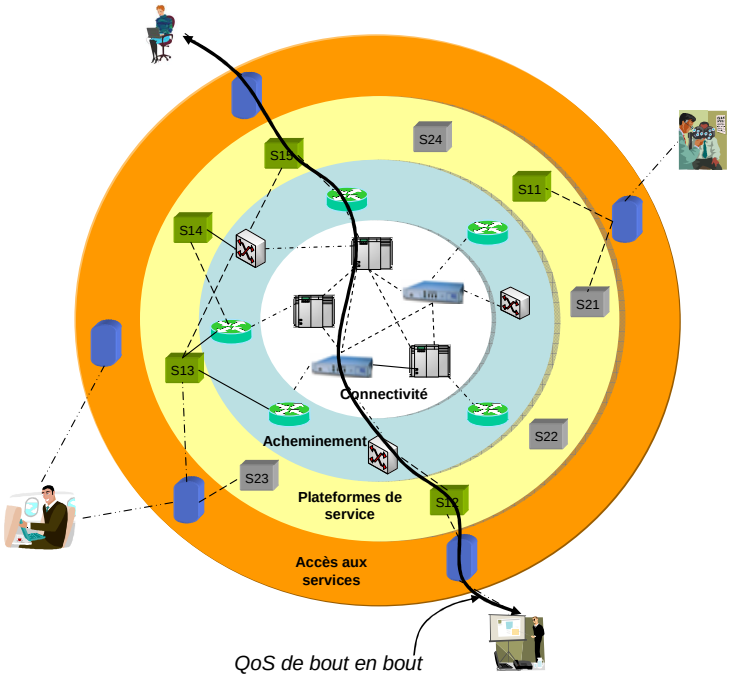


Figure I-17: La chaîne d'acteurs de bout en bout

3.1.2 Interaction verticale

Comme présenté ci-dessus (§3.1.1), la QoS de bout en bout est impactée par toute la chaîne, des équipements de transmission jusqu'aux plates-formes d'accès aux services. Et nous avons vu dans les deux chapitres précédents que les propositions existantes ne prennent en considération que le niveau du réseau d'acheminement et le niveau des services. Mais quelle est la vue des propositions existantes des services et leurs impacts ?

Nous avons vu que ce qui est considéré au niveau service (gestion de tampons, correction d'erreurs, et contrôle de congestion) comme élément impactant la QoS est le réseau d'acheminement, leur comportement et réactions restent assimilés au comportement du réseau, c'est-à-dire qu'ils sont conçus pour s'adapter au réseau et pas aux changements qui « arrivent » à la plate-forme de services elle-même comme la saturation d'accès au service ou la saturation des tampons, causes impactant la QoS perçue par l'usager final du service.

Les mécanismes basés sur l'observation des variations du RTT (§2.2), par exemple, peuvent conduire à une sous-utilisation du réseau [77], et une pénalisation face aux flux TCP. En plus l'observation de la réponse du réseau doit être lissée par la calcule de la moyenne sur une fenêtre temporelle glissante, afin d'éviter des phénomènes transitoires ou oscillatoires [83] qui augmentent le temps de réaction de l'architecture globale (Réaction suivant le comportement du réseau et non de l'ensemble).

Les propositions présentées pour le routage et la réservation de ressources applicatives franchissent ce pas et considèrent le service comme un niveau à part entière, décorrélé du réseau. Cependant, ces propositions voient ce niveau comme un niveau indépendant du niveau du réseau et sa QoS n'est impacté que par les éléments composants ce niveau. Considération qui n'est pas tout à fait juste car les services applicatifs utilisent le réseau d'acheminement comme moyen de transmission.

Plusieurs travaux ont essayé d'étudier l'interaction entres les différents niveaux à des fins de QoS, notamment, les travaux qui portent sur le Cross-Layer QoS. Ces travaux essayent de proposer un cadre architectural pour la fourniture de la QoS sur les réseaux fixes et mobiles impliquant les cinq couches du modèle OSI (application, transport, réseau, MAC et physique). Parmi ces travaux, nous citons le travail de Chuan [74], où l'auteur propose un cadre de délivrance de QoS de « client à serveur » pour les applications Internet (IP fixe) sensibles au délai. Zhang et al. [75] propose une estimation dynamique de l'état du canal de transmission mobile, la compression d'en-tête et une architecture de cache proxy adaptatif aux conditions du réseau de transmission 3G. Chen et al. [76] propose un ordonnancement basé sur la QoS et un schéma d'adaptation à la puissance du signal afin de coordonner le comportement de la couche basse (physique) pour une consommation optimale des ressources.

Cependant, bien que ces propositions introduisent la notion d'interactivité entre les niveaux architecturaux (application, réseau), ils proposent une organisation statique et figée des architectures et sont focalisés sur l'adaptation du réseau de transmission aux besoins des flux applicatifs. Ces architectures sont adaptées à un type spécifique de services (voix, vidéo, FTP et e-mail), ainsi il est très difficile d'introduire un nouveau service dans ces architectures proposées.

Dans notre démarche et suite aux études et tests que nous avons effectué (§Chapitre 6), nous sommes, donc, convaincu que le niveau service doit être vu comme un niveau à part entière. Certes, la QoS est impactée par le comportement du réseau d'acheminement des flux entre les éléments de ces services mais il faut considérer de façon distincte (pour pouvoir y remédier) les impacts induits par le comportement des éléments composants ce niveau et par la mise en relation des éléments de service que l'on peut offrir aux utilisateurs finaux.

De ce fait, une interaction verticale est nécessaire afin d'apporter une solution intégrée de QoS de bout en bout au lieu d'avoir des solutions isolées, chacune essayant de résoudre une partie du problème sans considérer l'architecture globale.

3.2 Temps et coût de déploiement

Une première séparation entre le niveau des services et le niveau du réseau d'acheminement s'est fait ressentir au cours de l'évolution des architectures de réseaux de télécommunication. Cette première séparation permet sans doute d'être plus réactif aux besoins du marché en termes de services à valeur ajoutée. Cependant, les services actuellement développés restent dépendants des réseaux d'acheminement (Figure I-18). En d'autre terme, un réseau d'acheminement par type de service, par conséquent, l'ajout d'un nouveau service nécessite le redéveloppement de toute la pile protocolaire (services intelligents sur LAN par exemple), induisant un temps et un coût de déploiement non maîtrisable de nouveaux services.

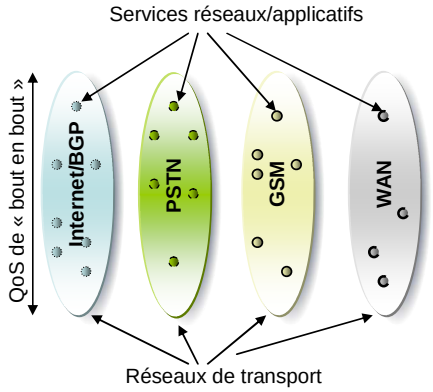


Figure I-18: Modèle existant de fourniture de services

De ce fait, une modularité de l'architecture globale de l'opérateur, que nous proposons dans notre solution, peut être une solution qui prend en charge la réalité du TTM et du CAPEX. Pour cela, les composants d'une architecture doivent être automatiquement réutilisables avec un minimum (voire aucune) d'intervention externes.

3.3 Evolutivité

La question qui se pose suite à la vision des deux critères exposés ci-dessus, à savoir le support de la QoS de bout en bout et le temps/coût de déploiement de services à valeurs ajoutées, est la question d'évolutivité de ces solutions ? Nous entendons par une solution évolutive une solution supportant de nouveaux besoins et technologies sans changement important de sa structure interne.

En termes de dynamicité trans-génération, comment supporter les évolutions technologiques qui ne se font pas en même rythme que les concepts ?

3.4 Synthèse

L'architecture de télécommunication d'un fournisseur de services doit être la plus réactive possible aux nouveaux défis qui apparaissent au fur et à mesure que les besoins des usagers de ces services évoluent. Cette première partie du rapport a permis de faire le tour d'horizon des différentes solutions proposées pour supporter l'évolution des besoins en termes de QoS ainsi qu'en termes de diversité des services. Elle montre également l'ampleur des problématiques auxquelles doivent faire face les opérateurs et fournisseurs de services de télécommunication.

Nous avons vu que les différentes solutions proposées ont apporté des réponses isolées à la problématique de réactivité et flexibilité de l'architecture à des fins de QoS de bout en bout.

Ces vues isolées de l'architecture vont en contre courant avec les besoins de réactivité et flexibilités exigées par les architectures d'aujourd'hui pour deux raisons : la première est que l'organisation de l'architecture est faite de manière « overlay », c'est-à-dire que les deux niveaux coexistent sans interaction dynamique entre eux. La deuxième raison est que le développement des services dépend d'une pile protocolaire pour chaque nouveau service. Cette solution est limitée dans le temps surtout quand on constate que la tendance actuelle est vers les services personnalisés où chacun peut définir son propre service. Dans ce cas, la variété de services proposés peut être très importante et le cycle de vie du service peut être très court. Afin d'optimiser le Revenu Moyen par Utilisateur (ARPU), l'impact d'un niveau architectural sur l'architecture globale doit être quantifiable, les architectures des opérateurs doivent être flexibles par rapport aux nouveaux besoins et adaptables aux changements de l'environnement.

Cette vision nécessite l'intégration de l'architecture des fournisseurs de service à tous les niveaux (les équipements matériel, le réseau de transport, les plates-formes de services (RI, OSA, SIP AS, etc.) et les plateformes de prise en charge des la personnalisation des services (Userware [réf])). Cette intégration englobe :

- l'intégration verticale, c'est-à-dire, qu'une décision de ne peut prise à un niveau sans une *cohabitation* des autres niveaux et sans pouvoir évaluer les impacts d'une telle décision sur les autres niveaux et notamment sur la perception des usagers finaux (QoS de bout en bout).
- l'intégration horizontale, c'est-à-dire, que sur un niveau donné, quelque soit les sous-domaines traversés (exemple, pour le niveau réseau de transport, les sous-domaines peuvent être des sous-réseau dans le sens adressages, des zones de routage dans le

sens routage intra-domaine, des systèmes autonomes dans le sens routage inter-domaines, ou simplement des réseaux hétérogènes (IP, ATM, Wifi, etc.)), une *coopération* de ces sous-domaines doit être maintenue afin que la QoS (horizontale) de ce niveau doit être respectée.

Notre objectif dans cette thèse est de donner une première solution qui se veut globale à la problématique d'intégration des architectures de télécommunication. Nos réflexions pour la résolution de cette problématique nous ont menés à proposer un nouveau concept que nous avons baptisé « Ingénierie d'Architecture » que nous nous proposons de présenter dans la deuxième partie de ce rapport.

Partie II A LA RECHERCHE D'UNE SOLUTION ….

Introduction

La première partie de ce rapport a été consacrée à la présentation de notre périmètre d'investigation, où nous avons soulevé des questions relatives à l'évolutivité et la réactivité des architectures de télécommunication aux nouveaux besoins émergents, notamment, les services à valeurs ajoutées et la QoS de bout en bout. Pour y répondre, nous proposons dans cette partie de présenter le cœur de notre travail de thèse en deux chapitres.

Le chapitre 4 développe notre proposition du concept d'Ingénierie d'Architecture. Il présente notre modélisation de ce concept, et les défis auxquels nous faisons face pour mettre en œuvre ce concept.

Afin de mettre en œuvre ce concept d'ingénierie d'architecture, nous proposons dans le chapitre 5 la concrétisation de ce concept à travers une plate-forme baptisée Pilote Organisationnel (POrg).

Chapitre 4

L'Ingénierie d'Architecture

Nous avons abordé dans les chapitres précédents les différentes approches de conception et de mise en œuvre des architectures de télécommunication. Nous voulions, à travers ces chapitres, faire valoir l'idée que la séparation des niveaux architecturaux constitue une première solution efficace aux problématiques des architectures d'aujourd'hui, à savoir, la réactivité, la flexibilité et la QoS de bout en bout. Cependant, une séparation totale de ces niveaux ne semble pas être la solution la plus puissante du fait que le comportement d'un niveau donné est influencé par les décisions prises dans les autres niveaux, impactant, ainsi, le comportement global de l'architecture qui est ressenti par les usagers finaux.

Afin de pouvoir évaluer et gérer ces impacts, nous avons développé un concept que nous avons baptisé : Ingénierie d'Architecture. Ce chapitre est essentiellement consacré à la présentation de ce concept.

L'Ingénierie d'Architecture repose sur une modélisation du monde réel indépendamment de sa gestion. Elle propose une vue homogène des différents composants de l'architecture et des niveaux architecturaux. Elle préconise ensuite la gestion et l'ingénierie de cette architecture à travers la QoS, c'est-à-dire, que chaque composant, chaque niveau architectural est piloté et invoqué en fonction de son comportement en cours d'exploitation.

Pour ce faire, nous commencerons par définir la notion de QoS et préciser les propositions d'évaluation auxquelles ont abouti les travaux de l'E.N.S.T. (section §4.1). Puis, pour avoir une vue de l'architecture globale, nous présentons un modèle des réseaux composant l'architecture permettant d'identifier les points d'observation clés. Ainsi nous obtenons une représentation du comportement la plus fiable et la plus homogène possible §4.2, permettant d'identifier les points d'observation clés afin d'avoir la représentation du comportement la

plus fiable et la plus homogène. Le concept d'Ingénierie d'Architecture consolide les deux sections précédents et propose une modélisation du monde des télécommunications à des fins de réactivité et d'intégration à travers la QoS (§4.3). Afin de mettre ces concepts en œuvre, plusieurs défis doivent être surmontés, la section §4.4 présente ces défis et la solution que nous proposons pour en faire face.

4.1 La Qualité de Service

4.1.1 Définition de la QoS

La QoS est une notion très usitée et très focale actuellement sur le monde des réseaux de télécommunication. Cette notion offre aux différents acteurs contractants, une manière d'inclure des clauses supplémentaires dans leurs accords, ou de formaliser plus en détail les prestations attendues ou fournies. Cependant, les notions de QoS restent confuses et génèrent de multiples définitions. Cette rançon est due aux énormes enjeux économiques qu'elle laisse entre apercevoir chaque domaine proposant sa propre vision.

Comme nous l'avons vu dans le chapitre 2, chaque constructeur/ « implémenteur » avance ses mécanismes de QoS, ou son implémentation de ces mécanismes. De plus, chaque niveau : réseau, service, etc. propose une vision différente de la QoS. Cet ensemble de définitions rend difficile son étude et sa compréhension.

Parmi toutes ces définitions, celle que nous trouvons la plus globale est celle de la norme E800 [77] : « La QoS se réfère à l'effet global produit par l'exécution (qualité de fonctionnement) d'un service qui détermine le degré de satisfaction de l'usager du service ».

D'après cette définition, l'usager final d'un service est le centre de la problématique. Ce qui nous permet de préciser que tout service est le résultat d'un contrat de prestation (explicite ou implicite) par un fournisseur au profit d'un usager. Mais comment caractériser cette QoS (§4.1.2) ? Et avons-nous un moyen cohérent et homogène d'évaluer le degré de satisfaction de l'usager d'un service donné (§4.1.3)?

4.1.2 Caractérisation de la QoS : le modèle DFDC

Le but de ce paragraphe est d'identifier, à travers les caractéristiques d'un service donné, des critères cohérents et homogènes pour traduire le comportement de ce service. En effet, lorsqu'un utilisateur fait appel à un service donné, il désire que ce service réponde à ces

besoins de la manière la plus transparente possible. Nous avons défini trois (3) classes de transparences : temporelle, spatiale et sémantique :

Transparence temporelle

Cette transparence implique que

- le service doit être capable de traiter les informations à chaque fois que l'utilisateur de ce dernier en produit et aussi longtemps que dure leur génération,
- le service ne doit pas changer la relation temporelle intrinsèque aux informations générées, c'est-à-dire maintenir le comportement de la source.

Afin de traduire cette transparence, nous avons défini deux critères : *Disponibilité* qui traduit la première caractéristique, et *Délai* qui traduit la deuxième.

Transparence spatiale

Cette transparence implique que le service doit traiter les informations générés par l'utilisateur instantanément quelque soit leur volume. Nous traduisons cette transparence à travers le critère de *Capacité.*

Transparence sémantique

Cette transparence implique que le traitement des informations générées par l'utilisateur se fasse sans les altérer. Nous traduisons cette transparence à travers le critère de *Fiabilité.*

A partir de ces définitions, nous constatons que la Qualité de Service traduit les caractéristiques non-fonctionnelles de tout service, et non pas la *meilleure qualité*, caractérisant, ainsi, tout service de manière unifiée et homogène.

4.1.3 Evaluation pertinente de la QoS

L'évaluation de la QoS est une étape fondamentale pour s'assurer du comportement du service offert à l'utilisateur potentiel conformément au contrat souscrit. Le modèle d'évaluation de la QoS que nous préconisons à l'ENST, comme nous venons de le définir dans §4.1.2 est celui qui traduit de manière homogène la qualité de n'importe quel service, à travers les quatre (4) critères *génériques*: *Disponibilité*, *Fiabilité*, *Délai* et *Capacité.*

Disponibilité

Représente l'accessibilité d'un service en tenant compte des conditions contractuelles temporelles et spatiales.

Fiabilité

Représente l'aptitude d'un service à être exécuté sans détérioration de l'information traitée et en respectant les conditions contractuelles.

Délai

Représente l'aptitude d'un service à être exécuté en respectant le temps spécifié par les conditions contractuelles.

Capacité

Représente l'aptitude à avoir les moyens nécessaires pour accomplir un service suivant les conditions contractuelles.

Chaque critère est évalué à différents niveaux d'abstraction de l'architecture globale et pour chaque composant qui participe à la chaîne de bout en bout. Il se décline à travers des paramètres *mesurables* à chaque niveau de visibilité.

4.2 Modèle abstrait de service

Plus la complexité et la diversité des services augmentent, plus le besoin de l'automatisation de l'autogestion des services devient plus fort. Notre solution repose sur une méthode permettant l'analyse d'une demande de service et la définition des flux nécessaires pour rendre ce service. L'intérêt de notre méthode est de faire apparaître tous les composants architecturaux impactant la QoS de bout en bout de ce service. Ces entités ne sont visibles qu'à travers leur QoS, c'est-à-dire, qu'à travers les quatre critères génériques DFDC (voir §4.1). Les composants architecturaux peuvent être des entités protocolaires, des mécanismes de contrôle d'accès, de classification, de compression, etc. et la communication entre ces composants.

La méthode doit apporter une décomposition rigoureuse du service global rendu à l'usager final à travers les différents niveaux d'abstraction (décomposition verticale). Dans un niveau donné (réseau par exemple), une décomposition horizontale définit les moyens mis en œuvre

pour assurer une QoS dans le niveau considéré (QOSPF et RSVP par exemple). Nous nous appuyons pour cette décomposition sur une modélisation suivant les principes ci-dessous :

- Toute décomposition d'un service est représentée par le modèle « Nœud/Lien/Réseau » tel que c'est défini dans le paragraphe §4.2.1.
- Chaque service et sa décomposition horizontale identifient un niveau de visibilité. Un niveau de visibilité est tel que c'est défini dans le paragraphe §4.2.2.

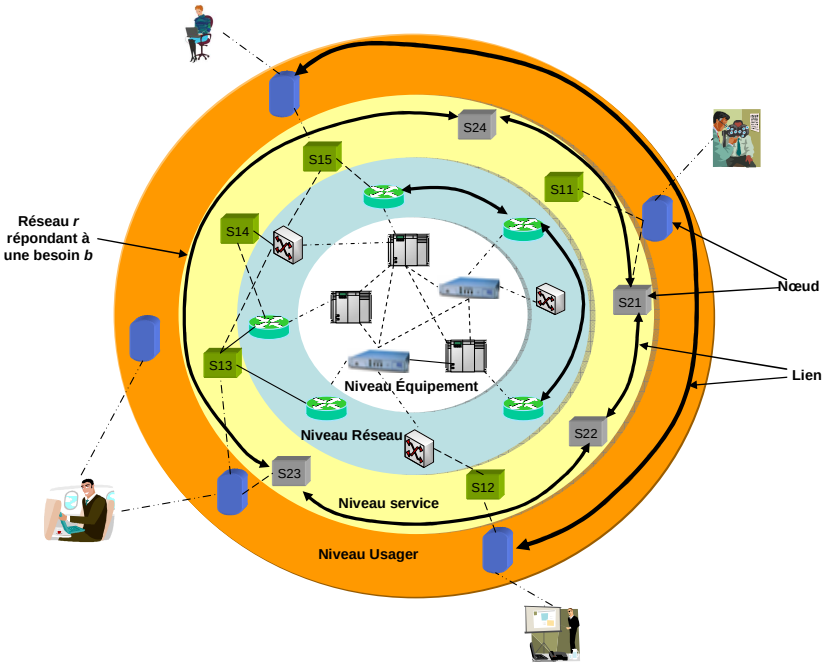


Figure II-1: Modélisation du monde des télécommunications

Nous présentons par la suite la description détaillée de ces définitions :

4.2.1 Le modèle N/L/R

Ce concept traduit notre volonté à modéliser le monde réel (réseaux de télécommunications) de la manière la plus objective possible, sans considérer la gestion du système ni son comportement ni aucun autre filtre, si ce n'est celui d'un ensemble d'entités communicantes. Nous le modélisons comme un ensemble de nœuds de traitement liés à travers des liens, formant ainsi un réseau destiné à rendre un service prédéfini. Nous définissons chacun de ces composants dans ce qui suit.

Nœud

Le Nœud, objet abstrait, définit le composant élémentaire du niveau considéré (routeur, switch, etc.), un élément de son architecture logique. Chaque nœud contribue, par son

comportement propre et par les services qu'il rend, à réaliser les fonctions du niveau de visibilité auquel il appartient. Chaque nœud associé au service possède des caractéristiques propres et une autonomie d'exécution mais il est responsable de la QoS qu'on lui a confié (QoS du nœud). Le service global est réalisé par le traitement et l'échange d'informations entre les nœuds impliqués. Les interactions entre ces nœuds sont assurées par les liens.

Lien

Le Lien entre deux nœuds représente le service de transfert entre ces nœuds et abstrait les ressources du niveau considéré impliquées dans ces échanges. Il est responsable de la transmission, quelle que soit sa complexité, de l'information entre les nœuds. Il représente donc le support des flux échangés entre nœuds appartenant au même niveau d'abstraction.

Dans le cas où la mise en relation entre deux nœuds nécessite l'utilisation de service de niveau inférieur, la décomposition du service de transmission, est donc du lien, est demandée au niveau inférieur.

Réseau

Les nœuds et les liens de même niveau d'abstraction constituent le Réseau (Figure II-2) à travers lequel un service de communication est rendu. Le réseau de niveau *N* peut être décomposé en sous réseaux du niveau inférieur et, de la même façon, peut faire partie du réseau de niveau supérieur. Ainsi, le réseau abstrait du niveau *N* offre au niveau *N*+1 un service de communication vu de ce niveau (*N*+1) comme un lien établi entre les composants du réseau *N*.

La définition de ces trois éléments génériques, à savoir, nœud, lien et réseau permet d'avoir une vue homogène et unifiée des différents éléments composant l'architecture de n'importe quel fournisseur de service indépendamment du niveau considéré, c'est-à-dire que nous voyons les services applicatifs de la même manière que le réseau d'acheminement et il en est de même pour les équipements qui supportent ces deux niveaux ainsi que les plates-formes d'accès aux services applicatifs.

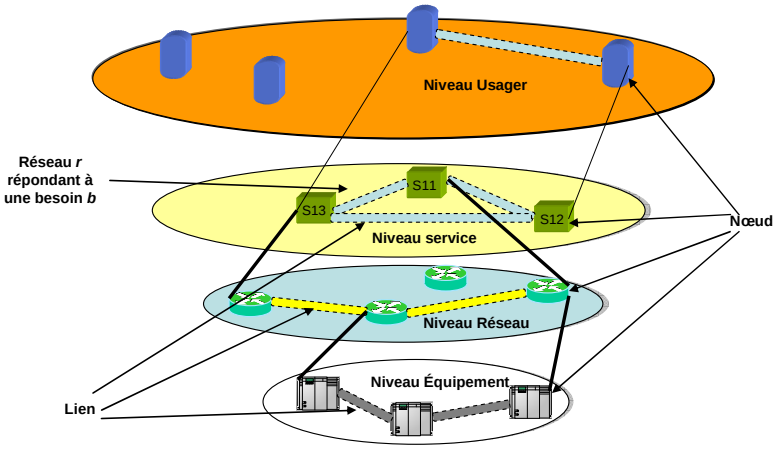


Figure II-2: Décomposition de service

L'introduction de la modélisation NLR pour représenter une architecture complexe a un double objectif :

- celui de séparer les blocs architecturaux en niveaux décisionnels homogènes, que nous définissons dans le paragraphe suivant
- et celui *d'intégrer* ces niveaux (les rassembler) à des fins de transparence pour l'utilisateur final.

4.2.2 Niveaux de visibilité

Un niveau de visibilité est un niveau d'abstraction permettant de différentier les services rendus. Un niveau de visibilité propose une image abstraite d'un réseau (suivant le modèle NLR) autonome capable d'analyser et de traiter les problèmes dans les limites de ses responsabilités. L'ensemble des niveaux de visibilité à définir n'est pas défini de façon générale et unique. Il traduit les niveaux de décisions qui peuvent être prises pour réaliser et maintenir un service donné. On peut limiter le nombre de niveaux de visibilités pour ne retenir que les plus pertinents dans un contexte donné. Le contexte traduit donc les niveaux de visibilité considérés. A des fins de QoS de bout en bout, nous avons retenu quatre (4) niveaux de visibilité (comme nous l'avons introduit dans la section §4.1) :

Niveau de l'équipement

Les nœuds dans ce niveau d'abstraction représentent les équipements physiques, au sens matériel, qui assument une partie de la fonction du système à travers la capacité de leurs ressources physiques. Quant aux liens, ils représentent l'ensemble des câbles reliant les

équipements et encapsulant l'ensemble des signaux (flux) électriques reçus ou émis par ces équipements.

Niveau du réseau

Le niveau réseau identifie la capacité de transmission du système et caractérise l'architecture du réseau d'acheminement. Les Nœuds dans ce niveau représentent les entités protocolaires. Les Liens sont les voies virtuelles qui supportent les flux entre deux nœuds. Le Réseau utilisateur, à un instant t, sera constitué de l'ensemble des nœuds et des liens se trouvant sur le chemin choisi en fonction du flux utilisateur (circuits virtuels, LSPs, etc.). Le Réseau fournisseur étant constitué de tous les chemins potentiels pour l'ensemble des flux acheminés.

Niveau du service

Les Nœuds de ce niveau représentent les entités de services applicatifs permettant de construire et d'effectuer les services rendus par cette couche (qui s'est fait à un instant donné et pour un but précis. Selon sa granularité, il abstrait un processus, une application ou tout élément auquel fera appel un service de granularité plus importante (le service de Streaming fait appel à un service de compression par exemple). Quant aux Liens, ce sont les interactions entre ces éléments de services (les bus d'échanges des middlewares par exemple). Le réseau dans ce niveau, représente la logique du service rendu par le service global.

Niveau de l'usager

Les Liens de ce niveau représentent les mises en relations temporaires des Nœuds qui représentent les utilisateurs et fournisseurs de service formant ainsi des *organisations virtuelles*. Ce niveau fait abstraction du niveau du service, c'est-à-dire, les clients dans une organisation virtuelle (client/fournisseurs) ne voient pas les services mais voient les fournisseurs de ce service à travers leurs offres d'un service donné. Cette abstraction permet la personnalisation des services de manière plus efficace.

A partir de ces deux concepts de base (modèle NLR §4.2.1 et les niveaux de visibilités §4.2.2), nous pouvons modéliser l'architecture globale d'un fournisseur de services Internet comme illustré dans la Figure II-1.

4.3 Modèle d'Ingénierie d'Architecture

A partir des définitions et modélisations que nous avons définies dans le paragraphe précédent, nous pouvons dans ce paragraphe introduire notre concept baptisé : *Ingénierie d'Architecture.*

L'ingénierie d'architecture désigne *l'assemblage dynamique des composants d'une architecture à des fins de QoS de bout en bout*. Son objectif principal est d'assurer l'adéquation entre les besoins des utilisateurs et l'exploitation du réseau de l'opérateur pour une QoS de bout en bout, tout en tenant compte des contraintes de l'existant afin d'avoir une architecture flexible plutôt que plusieurs architectures (piles protocolaires) rigides.

Nous entendons par *architecture* la structure d'ensemble, c'est-à-dire, les règles de consistance d'un système (équipements, réseau, service, usagers) d'intégration de ses composants et de leur coopération, qui permettent d'atteindre un double objectif : optimisation et efficacité.

La modélisation proposée pour l'Ingénierie d'Architecture d'un réseau (opérateur, ISP, etc.) est une superposition de couches architecturales englobant différents niveaux en fonction du service à fournir (voir §4.2.2). A chacune d'elles est associée un réseau autonome, une vue abstraite d'un ensemble de nœuds et de liens capables de traiter les problèmes inhérents à son champ de responsabilités (respect du contrat de QoS horizontale) (voir §4.2.1). La façon dont le service de niveau *N* est rendu au niveau supérieur reste invisible, c'est-à-dire que le service est vu par le niveau supérieur comme un élément indivisible et autonome.

Pour traiter l'ingénierie d'architecture, nous avons retenu les quatre (4) niveaux suivants (§voir 4.2.2) : équipements, réseau, service et usager. Ces quatre niveaux identifient au mieux les différents acteurs de la chaîne de QoS de bout en bout. Cette QoS, comme expliquée précédemment, est impactée par

- les choix de superpositions des blocs architecturaux,
- le respect du contrat de QoS confié à chaque niveau.

Ces deux points constituent les composantes de base de l'ingénierie d'architecture. Nous attribuons au premier point des capacités de *Cohabitation* (§4.3.1) et au deuxième point celles de *Coopération* (§4.3.2).

4.3.1 Cohabitation

Cette composante s'assure de la consistance de la structure d'ensemble, c'est-à-dire que la superposition des blocs architecturaux entre les niveaux soit fonctionnellement complémentaire et non redondante pour la QoS de bout en bout. Le but final étant d'automatiser les processus de déploiement de nouveaux services en intégrant leurs besoins en QoS. Notre approche repose sur deux principes : principe de la déclinaison de la QoS et principe de l'agrégation de la QoS.

Les principes de déclinaison et d'agrégation (Figure II-3), que nous précisons par la suite, sont nécessaires pour répondre aux besoins QoS des utilisateurs en optimisant le déploiement de nouveaux services.

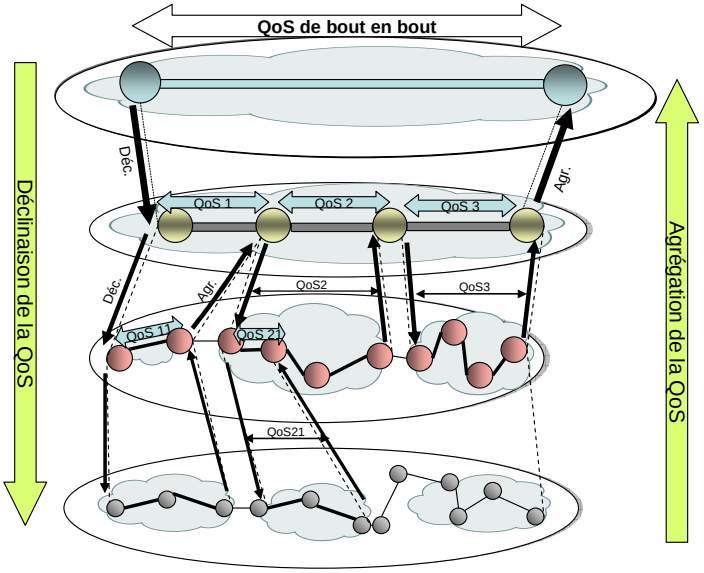


Figure II-3: Composante *cohabitation* de l'ingénierie d'architecture

Le principe de déclinaison

La déclinaison de la QoS fait référence au processus d'analyse de l'architecture globale de haut en bas (top-down) en visant deux objectifs :

- trouver une solution pour une QoS de bout en bout,
- maintenir la cohérence de la structure d'ensemble.

Cette déclinaison sous-entend l'adaptation sémantique du modèle de QoS[4] aux différents niveaux de visibilité. Nous définissons deux types de déclinaisons : informationnelle et fonctionnelle.

- *Déclinaison informationnelle*

La déclinaison informationnelle désigne la partie de la translation des besoins QoS des niveaux supérieurs aux niveaux inférieurs. Une demande de QoS au niveau *N* est d'abord *contextualisée*, c'est-à-dire, exprimée dans le même niveau d'abstraction que ce niveau *N*. Pour donner un exemple, si on parle du niveau usager, les besoins QoS de l'usager pour un service de Streaming sont exprimés en terme « je voudrai accéder au service quand je veux », « sans coupure », « intelligible », « qualité DVD », etc. Ces besoins sont interprétés pour le niveau du service applicatif comme : « taux d'accessibilité à la plate-forme de service », « codage MPEG », etc. Ces derniers sont interprétés pour le niveau réseau comme, « taux de perte », « délai », « gigue », « bande passante », etc. Qui sont à leur tour interprétés comme « vitesse CPU », « nombre d'interfaces », « taille écran » dans le niveau équipement.

- *Déclinaison fonctionnelle*

La déclinaison fonctionnelle désigne la partie de délégation des fonctions (ou une partie des fonctions) par sous-traitance des niveaux supérieurs aux niveaux inférieurs permettant de satisfaire les besoins (trouver une solution) de QoS de bout en bout. Il s'agit ici de trouver une solution (architecture du service) qui répond aux besoins QoS du service. C'est à dire pour assurer une QoS de bout en bout, le niveau supérieur assure certaines tâches selon les possibilités des couches inférieures et délègue à ces couches inférieures les autres tâches et ainsi de suite.

En résumé, la déclinaison de la QoS est un processus nécessitant trois étapes : *contextualisation* de la demande, *interprétation* des besoins et *délégation* des responsabilités à travers un contrat de QoS (SLA) entre le niveau *N* et niveau *N*-1.

Le principe d'agrégation

Le principe d'agrégation fait référence au processus d'analyse de l'architecture globale de bas en haut (bottom-up), c'est-à-dire que les couches supérieures doivent recevoir des informations de rétroaction des couches inférieures afin que les niveaux supérieurs s'assurent

[4] Le modèle DFDC (§4.1.2)

du bon déroulement de la déclinaison et éventuellement relancer le processus de déclinaison suivant ces informations de rétroaction.

Afin de rendre une architecture flexible aux nouveaux besoins et réactive aux changements qui se produisent (pannes, insuffisance de ressources, coupures liens, saturation service, etc.), ces deux processus de déclinaison et agrégation doivent être automatisés et transparents. Nous présenterons dans le Chapitre 5 notre proposition d'automatisation de ces processus.

La première composante de l'ingénierie d'architecture que nous venons de présenter donne une vue sur les liaisons verticales (l'*intégration verticale*) que doit avoir une architecture pour être plus flexible, mais comment assurer le contrat de QoS confié au niveau *N* par le niveau *N*+1 ? La réponse à cette question motive la définition de la deuxième composante de l'ingénierie d'architecture qui est la *Coopération* (§4.3.2).

4.3.2 Coopération

Comme présenté dans l'introduction de cette section (§4.3.2), à chaque niveau architectural est associé un réseau autonome, une vue abstraite d'un ensemble de nœuds et de liens capables de traiter les problèmes inhérents à son champ de responsabilité qui est le respect du contrat de QoS qui lui est confié par le niveau supérieur. Mais comment faire respecter ce contrat de QoS horizontale (l'*intégration horizontale*)?

Nous proposons de superviser et d'évaluer en temps réel la QoS offerte ($QoS_{(n)}$) à travers les différents « sous-réseaux » (domaine) constituant le niveau horizontal *N* (Figure II-4). Chaque domaine *i* est autonome, c'est-à-dire, responsable de la supervision de sa propre QoS ($QoS_{(n,i)}$) (SLA entre domaines). Si une dégradation de la QoS est constatée sur un (ou plusieurs) domaine(s), et donc non-conformité du contrat de QoS, nous proposons quatre (4) étapes pour résoudre le problème:

1. re-négocier le contrat de QoS (SLA) entre les domaines de telle sorte de rattraper la dégradation constatée dans le domaine *i* dans le domaine suivant (*i*+1).
2. changer l'itinéraire des flux traversant le domaine sur lequel on constate une dégradation pour les faire passer par un autre domaine qui aurait les ressources nécessaires aux besoins du flux.

3. réorganiser les domaines de telle sorte que des ressources soient disponibles de bout en bout horizontal.

4. feedback négatif au niveau supérieur (*N*+1) indiquant la non possibilité d'accomplir le contrat vertical pour qu'il prenne des décisions (modifier les besoins, réorganiser le niveau *N*+1, etc.).

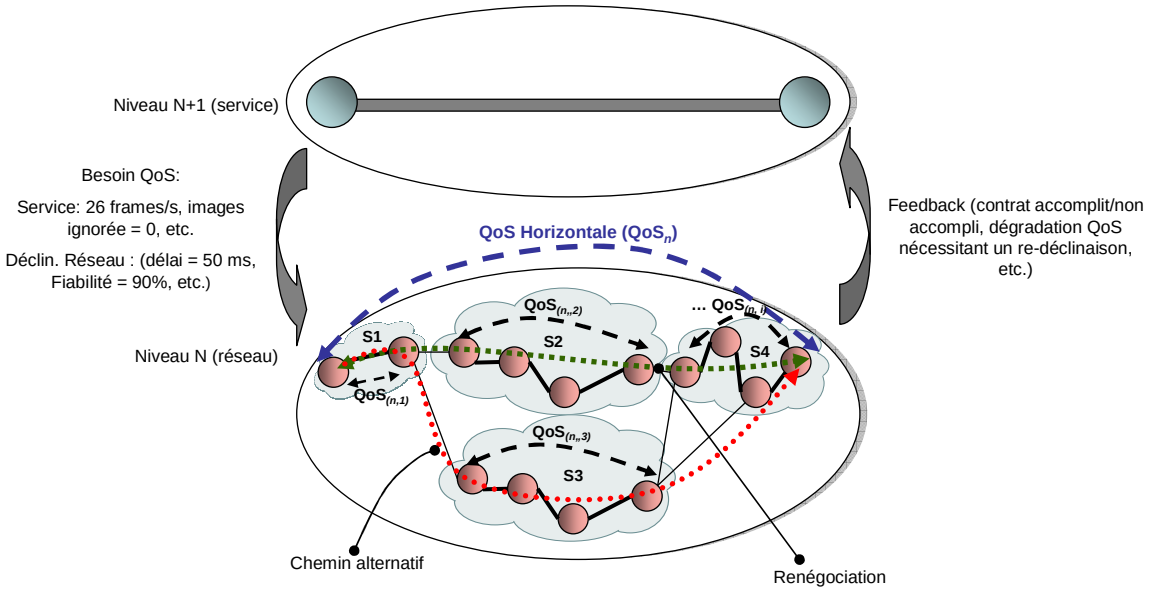


Figure II-4: Composante *coopération* de l'ingénierie d'architecture au niveau N

La définition des deux composantes, cohabitation et coopération, et donc l'ingénierie d'architecture, permettrait d'aboutir à l'objectif que l'on s'est fixé dans cette thèse qui est de trouver les moyens de rendre l'architecture d'un opérateurs la plus flexible possible aux nouveaux besoins en terme de services innovants ou besoins usagers de personnalisation dans le contexte d'une QoS de bout en bout. Mais quel est le degré d'applicabilité de ce concept dans le monde réel.

Dans le paragraphe suivant (§4.4), nous présenterons les contraintes à surmonter pour mettre ces concepts en application.

4.4 Ingénierie d'Architecture : les verrous et les clés

Plusieurs contraintes doivent être surmontées pour mettre en œuvre l'Ingénierie d'Architecture. Ces contraintes peuvent être résumées grossièrement en trois (3) points : prendre en compte l'hétérogénéité de l'architecture (§4.4.1), maintenir la consistance de l'architecture (§4.4.2) et la réduction du temps de réaction de l'architecture (§4.4.3).

4.4.1 Hétérogénéité des niveaux architecturaux

- *Verrou*

Le premier verrou à lever est la prise en compte des différents niveaux architecturaux d'un système. La nature des différents niveaux architecturaux d'une architecture est typiquement hétérogène (nœuds et liens de natures différentes), pour le réseau on a une visibilité sur les routeurs et les flux agrégés traversant ce réseau, le niveau service quant à lui représente les éléments de service (composants logiciels distribués) et les flux applicatifs, le niveau utilisateur représente les communautés virtuelles et les besoins utilisateur (requêtes utilisateur/fournisseur).

- *Clé*

Afin de gérer cette hétérogénéité de manière transparente, nous devons :

- Avoir un même modèle de chaque composant dans le réseau pour ne retenir que la représentation commune de ces composants, cette image définit le comportement générique de chaque composant.
- Pouvoir découvrir chaque composant de manière automatisée

4.4.2 Consistance et cohérence de l'architecture

- *Verrou*

Le deuxième verrou est la préservation de l'intégrité de la structure d'ensemble. Il s'agit de palier les états inconsistants résultant des échecs et des inefficacités opérationnels lors du processus de déclinaison, comme par exemple, mauvais choix du protocole de transport sous-jacent, paramétrage non optimal du classifieur, contradictions fonctionnelles entre les niveaux architecturaux, etc. Une organisation non efficace peut, non seulement, générer une dégradation de la QoS perçue de bout en bout, mais aussi occasionner une panne intermittente du système (nœuds de service, routeur, etc.).

- *Clé*

Afin de gérer cette inconsistance, on doit :

- Etablir et représenter les règles de consistance d'une organisation,
- Avoir un feedback sur le déroulement du processus d'organisation,

- Contrôler/Surveiller le système pour avoir une traçabilité complète et ainsi localiser les points d'inconsistance.
- Restaurer les états de configuration précédents $(t_i - 1)$ en cas d'erreur ou d'inconsistance.

4.4.3 Réduction du temps de réaction de l'architecture

- *Verrou*

Le troisième verrou qui est en même temps un objectif à atteindre est la réduction du temps de réaction d'une architecture aux changements internes nécessitant une réorganisation du système. Le processus d'organisation d'une architecture nécessite la structuration des changements (configuration, activation, désactivation, etc.) dans plusieurs éléments hétérogènes. Par exemple, la migration d'un nœud de service nécessite la réorganisation des mécanismes de QoS sous-jacents de façon à ce qu'ils soient activés dans le nœud réseau le plus proche géographiquement du nouveau nœud de service pour maintenir la QoS de bout en bout.

- *Clé*

Afin d'atteindre cet objectif, on doit:

- Pouvoir découvrir les composants architecturaux (les nœuds),
- Avoir un retour sur les changements survenant sur l'état du système,
- Contrôler/Surveiller le système pour avoir une traçabilité complète et ainsi localiser les points de dysfonctionnement.
- Réagir au plus près du dysfonctionnement
- Répondre aux changements de manière événementielle

4.5 Conclusion

Pour des *cohabitation* et *coopération* flexibles, nous avons proposé un concept permettant l'intégration de toutes les dimensions, et que nous avons désigné comme étant celui de l'Ingénierie d'Architecture. Reste maintenant à valider ces concepts et les mettre en œuvre. Ceci est l'objectif du chapitre suivant

Chapitre 5

De l'Ingénierie d'Architecture au Pilote Organisationnel (POrg)

Dans le chapitre précèdent, nous avons présenté l'ingénierie d'architecture, une modélisation permettant de concevoir, exploiter et maintenir l'architecture de la façon la plus flexible et transparente possible. Reste maintenant à matérialiser ces concepts et modèles et les mettre en application.

Comme le souligne le chapitre précèdent, deux concepts composent l'ingénierie d'architecture : cohabitation entre les niveaux superposés autonomes mais intégrés (pour la QoS de bout en bout), et coopération entre les nœuds et domaines juxtaposés autonomes mais intégrés (pour la QoS horizontale). Ces deux composantes traduisent une volonté de pilotage des entités composants l'architecture globale d'un côté et attribution de rôles et emplacements *(organisation)* pour ces entités dans le maintien de la QoS de bout en bout d'un autre côté. De ce fait, nous avons développé un système (plate-forme) d'ingénierie d'architecture baptisé : *Pilote Organisationnel (POrg)* que nous décrivons dans ce chapitre.

Afin de surmonter les contraintes exposées dans le paragraphe §4.4, nous commençons par caractériser POrg dans la section §5.1. pour être le plus rigoureux possible, nous spécifions, par la suite POrg à travers trois (3) dimensions : la dimension *organisationnelle* qui définit le rôle, la portée et l'emplacement de chaque entité composant la plate-forme POrg dans l'architecture (§5.2), la *dimension fonctionnelle* qui définit l'ensemble des fonctions et actions à mettre en œuvre par POrg pour assurer l'ajout de nouveaux services et maintenir la QoS de bout en bout de manière transparente (§5.3) et la dimension *architecturale* qui définit la structure et la coordination des modules composant *POrg* (§5.4).

5.1 Caractéristiques de *POrg*

Les contraintes exposées ci-dessus (§4.4) nous permettent de dégager des caractéristiques indispensables pour que les entités puissent prendre en charge l'Ingénierie de l'Architecture (POrg). Ces caractéristiques sont : généricité (§5.1.1), distributivité (§5.1.2) et réactivité événementielle (§5.1.3).

5.1.1 Généricité

La première caractéristique de POrg est la généricité qui traduit la capacité d'instanciation de POrg dans différents contextes (ou même indépendamment du contexte). En effet, la plate-forme POrg doit prendre en charge toute une architecture qui est par définition une superposition/juxtaposition de niveaux/domaines de natures hétérogènes. Donc, il doit pourvoir prendre son rôle dans des contextes différents avec des contraintes et natures différentes. La généricité des modèles de conception de POrg est, par conséquent, indispensable.

5.1.2 Distributivité

Afin d'assurer la fonction d'organisation, POrg doit prendre en charge chaque composant de l'architecture. C'est-à-dire, vérifier son contrat de QoS, son impact sur la QoS du réseau, sur les plates-formes de services, etc. réorganiser le réseau ou les plates-formes de service pour faire respecter ce contrat, décliner les besoins QoS, surveiller, etc. Évidement, l'attribution de cette lourde tâche à une seule entité centralisée n'est pas une solution « scalable » et présente beaucoup de risques. La solution de distribution des tâches sur plusieurs entités est une solution à ce problème. Pour cela, nous avons défini POrg comme un système distribué sur les différents composants, domaines et niveaux architecturaux. Chaque entité est responsable de la vue architecturale qui lui est confiée.

5.1.3 Réactivité événementielle

La multitude des tâches que POrg doit assurer, le temps de réaction qui doit être optimisé et le maintien de la consistance de l'architecture ne permettent pas d'avoir des traitements procéduraux. POrg est une architecture event-based.

Munis de ces caractéristiques, nous allons maintenant proposer un modèle organisationnel.

5.2 Le modèle Organisationnel

Nous présentons dans cette section la dimension organisationnelle de POrg. Le modèle organisationnel définit le rôle, la portée et l'emplacement de chaque entité composant POrg dans l'architecture globale. Nous allons définir dans cette section les entités constituant POrg (§5.2.1), leurs rôles (§5.2.2), leurs emplacements (§5.2.3) et leurs responsabilités (§5.2.4).

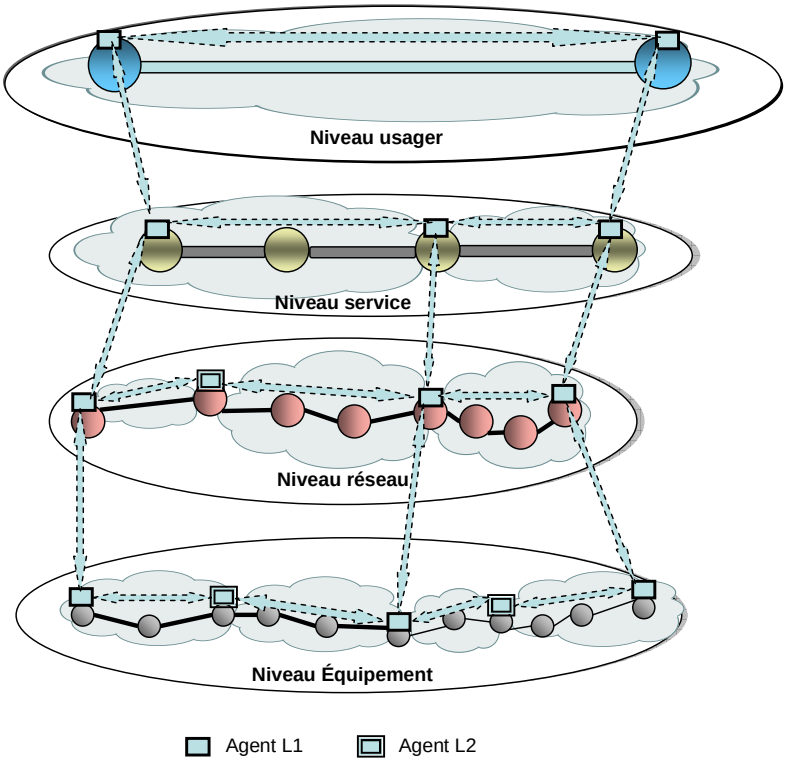


Figure II-5: Modèle Organisationnel de POrg

5.2.1 Composants de *POrg*

Afin de matérialiser les deux concepts : cohabitation (verticale) et coopération (horizontale), nous avons défini deux types de composants dans POrg (Figure II-5). Agents à responsabilité étendue (Agent L1), et agents à responsabilité réduite (Agent L2), sachant que chaque composant a potentiellement un « agent personnel, L3 » qui calcule et contrôle sa QoS.

Agents L1

Ces agents sont conscients de la répartition des composants architecturaux dans un niveau. Ils prennent en charge le processus de déclinaison et d'agrégation des besoins QoS. Ils peuvent communiquer entre eux afin de préserver la cohérence de la structure d'ensemble. Par

ailleurs, afin de maintenir la QoS dans un niveau donné (horizontal), ils peuvent communiquer avec les *Agents L2* (Figure II-5).

Agents L2

Ceux sont les agents appartenant au même niveau architectural qui se chargent de maintenir le contrat de QoS entre les différents domaines.

Donc nous aurons dans un même niveau architectural deux types d'agents : ceux de niveau I, conscients de la répartition des composants architecturaux dans un niveau donné, interactifs avec les niveaux adjacents (cohabitation/coopération) et ceux de niveau II, purement réactifs, ne se chargeant que du maintien du contrat de QoS horizontale (coopération seulement). Ces agents sont intégrés dans les nœuds architecturaux

5.2.2 Rôles de POrg

Les deux agents POrg peuvent avoir plusieurs rôles suivant leur emplacement et l'objectif assigné à la structure. Nous avons défini quatre (4) rôles pour les entités POrg : passif, actif, interactif et proactif.

En fait, les entités POrg sont présentes dans tous les composants architecturaux (agent L3) et ils sont responsables du contrat de QoS de ce dernier. A l'initialisation, ils prennent connaissance des valeurs de conception (valeurs DFDC offerts) et des valeurs seuils (valeurs DFDC seuils à ne pas enfreindre). En cours d'exploitation (transfert), ils surveillent les valeurs courantes et réagissent suivant ses valeurs (en cas de dépassement de seuils).

La partie POrg/QoS native (agent L3) est un rôle de type passif car effectif que pour le composant lui même. Le POrg devient actif (agent L2) quant il agit par délégation, Interactif quant il consulte ses homologues et proactif (agent L1) quand il prend des décisions.

Rôle Passif

Un rôle passif désigne l'état où un agent POrg ne fait que collecter et calculer les informations de QoS (DFDC). Il met à jour les valeurs courantes (bande passante disponible, ports disponibles, etc.). Dans les cas de non respect du contrat de QoS (dépassement de seuils, etc.) il n'envoie aucune notification aux autres entités POrg. Cependant, s'il est sollicité pour communiquer ces informations il doit répondre.

Rôle Actif

A l'initialisation, un agent POrg peut avoir à distribuer par délégation les contrats de QoS pour les autres agents ou toutes autres commandes venant d'une entité de gestion. Il aura alors un rôle actif pour le sous-ensemble confié à sa responsabilité.

Dans ce cas, en cours d'exploitation, il joue le rôle de métrologue et de contrôleur de la QoS. Il notifie à qui de droit (agents de niveau supérieur, du même niveau ou les entités de gestion) les dégradations constatées. Comme dans le rôle passif, il répond aux différentes sollicitations et il met à jour les valeurs courantes.

Rôle Interactif

Un agent interactif est un agent actif doté qu'une capacité d'interaction avec d'autres agents. C'est dire ici, qu'il est dans une relation en pair à pair avec ses homologues et peut négocier les paramètres de QoS à maintenir ou les composants architecturaux à activer pour une solution de bout en bout.

Rôle Proactif

POrg est interactif et possède des connaissances et des règles lui permettant de prendre des décisions « tactiques » pour palier un problème qui lui est propre ou qui relève de son domaine de responsabilité.

Dans l'absolue, toute entité *POrg* peut changer de rôles dynamiquement. Les agents *POrg* distribués sont donc des agents intégrés dans tous les nœuds du système. Nous avons appelé « Agents L1 » ceux ayant un rôle Interactif ou Proactif, et « Agents L2 » ceux ayant un rôle actif, tous les nœuds étant passifs à l'origine. L'emplacement de ces agents définit leurs rôles.

Les différents rôles seront définis avec plus de précisions en fonction de leur emplacement.

5.2.3 Emplacement des agents POrg

Après avoir défini les entités *POrg* et les types de rôles que peuvent avoir ces entités il reste à analyser leurs emplacements. Les emplacements stratégiques des agents *POrg* sont fonction des points de prise et d'application de décision qui peuvent être faites dans l'architecture globale. Nous proposons que ces points soient généralement situés aux extrémités des domaines. Ces extrémités sont des points d'accès que l'on peut classer en deux catégories (Figure II-6) : verticale et horizontale. Les extrémités verticales sont les points d'accès entre deux niveaux architecturaux, qui relient par exemple un nœud réseau à un nœud de service si

nous considérons les niveaux réseau/service. Les extrémités horizontales sont les points d'interconnexion de domaines de même niveau architectural. Typiquement les agents L1 sont intégrés dans les extrémités verticales, à l'intersection des niveaux de visibilité, alors que les agents L2 sont intégrés dans les extrémités horizontales, à l'interconnexion des sous-réseaux ou des domaines.

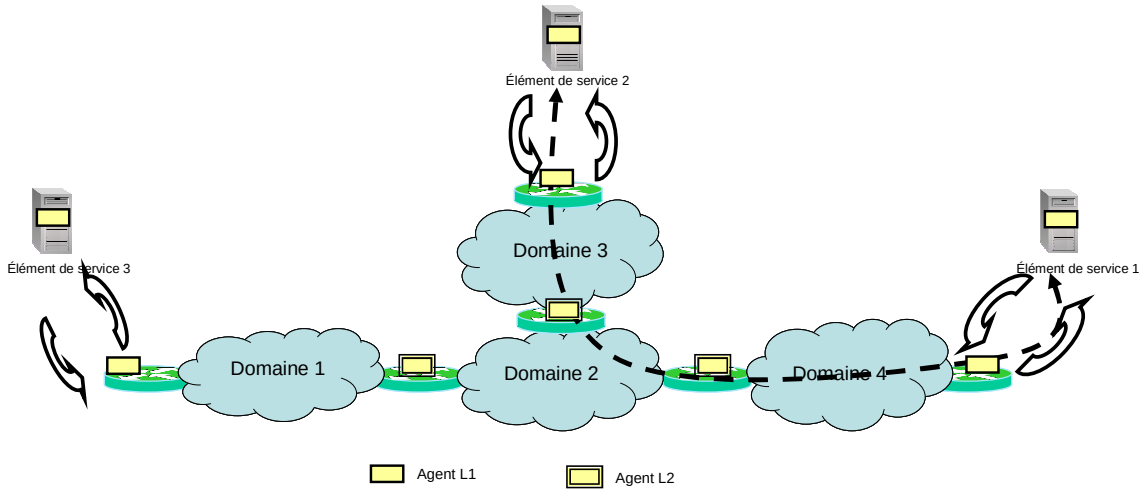


Figure II-6: Emplacement des agents POrg (réseau/service)

Un agent *POrg* est potentiellement intégré dans chaque nœud du système global (comme les agents SNMP) et peut être activé ou désactivé suivant les besoins. Nous présenterons, dans ce qui suit, notre proposition pour maintenir le contrat de QoS horizontale.

5.2.4 Maintien de la QoS horizontale

Comme expliqué dans le paragraphe §4.3.2, nous proposons quatre (4) étapes pour la résoudre la non-conformité au contrat de QoS horizontale :

1. re-négocier le contrat de QoS (SLA) entre les domaines de telle sorte de rattraper la dégradation constatée dans le domaine *i* sur le domaine suivant (*i*+1).
2. changer l'itinéraire des flux traversant le domaine sur lequel on constate une dégradation pour les faire passer par un autre domaine qui aurait les ressources nécessaires aux besoins du flux.
3. réorganiser les domaines de telle sorte que :
    a. des ressources soient disponibles de bout en bout horizontal.
    b. La stratégie de l'opérateur soit prise en compte

4. feedback négatif au niveau supérieur (*N*+1) indiquant la non possibilité d'accomplir le contrat horizontal pour qu'il prenne des décisions (modifier les besoins, réorganiser le niveau *N*+1, etc.).

Les deux premiers points et le dernier point relèvent de la dimension fonctionnelle que nous allons décrire dans le paragraphe §5.3, le deuxième point définit la façon de réorganiser la répartition des domaines afin de satisfaire le contrat de QoS d'un côté, et de l'autre côté, prendre en compte la stratégie de l'opérateur. Le critère de politique de l'opérateur peut être envisagé si la stratégie de l'opérateur est de faire passer le trafic par un chemin (domaine) qui n'est pas forcément le chemin optimal par rapport au plan de routage.

La réorganisation des domaines implique la réaffectation (désactivation des agents sur les nœuds précédemment d'extrémités et activation des agents sur les nouveaux nœuds d'extrémités) des agents (Figure II-7). La problématique ici est de trouver le regroupement optimal des nœuds du réseau pour une QoS de bout en bout. Nous appellerons ce problème *QoS-Based clustering Problem (QCP).*

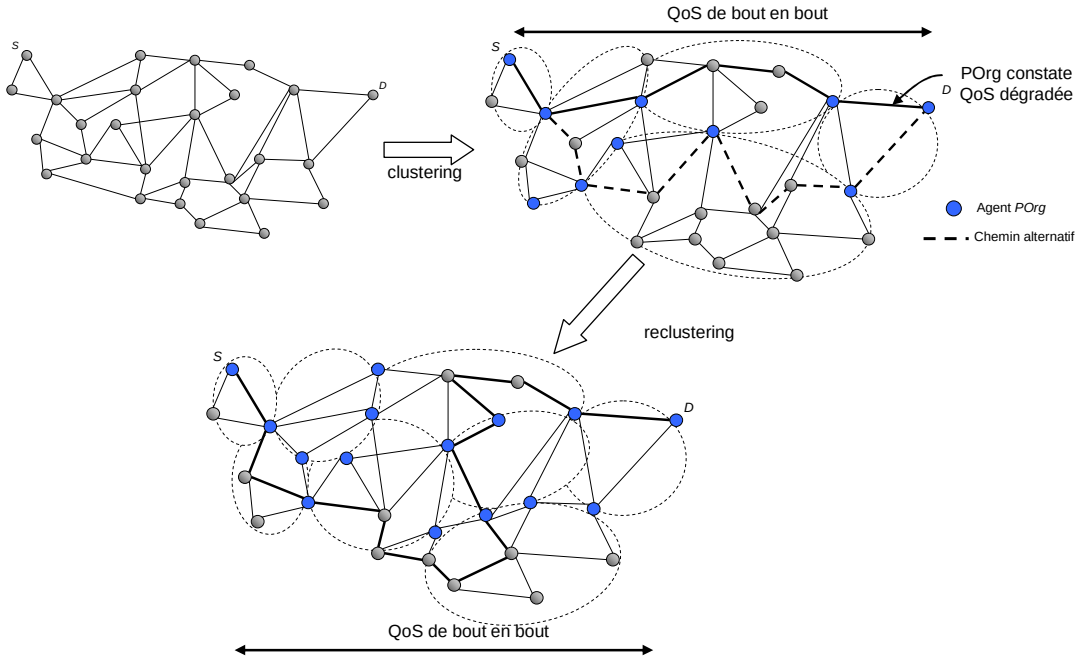


Figure II-7: QoS based clustering

Formalisation du problème

Avant de donner une définition formelle du problème, décrivons les notations utilisées.

Soit $G(M,E)$ le réseau de niveau *N* considéré (équipement, réseau, service et usagers), où *M* est l'ensemble des nœuds du réseau et *E* l'ensemble des liens. Le nombre de paramètres de QoS du modèle DFDC (§4.1.2) surveillés par les entités *POrg* est dénoté par *m*. chaque lien est caractérisé par un vecteur de poids *w* de dimension *m* constitué de *m* paramètres de QoS à surveiller ($(w_i(u,v), i = 1,2,...m, (u,v) \in E)$).

L'objectif est de trouver une répartition de l'ensemble des nœuds *M* en groupes (clusters) de telle sorte que les poids de l'ensemble des chemins à l'intérieur de chaque cluster soit maîtrisable. Lors d'une dégradation de la QoS de bout en bout d'un flux (ou agrégat de flux) constatée par *POrg* ou suite à une décision stratégique de l'opérateur, *POrg* agit sur le routage pour mettre à jour les tables de routage de telle sorte que le flux concerné traverse un cluster alternatif. L'emplacement des entités dépend, donc, de la façon dont le réseau est clusterisé.

La mesure de QoS effectuée sur un lien peut être additive (comme le délai), multiplicative (comme les pertes) ou concave (comme la bande passante).

- Dans le cas de mesures additives, le poids d'un cluster représente le poids maximal des chemins de n'importe quelle source vers n'importe quelle destination à l'intérieur de ce cluster.

$$W(k) = Max(\sum_{(u,v)\in P} w(u,v))$$

Équation 1: Poids d'un cluster (métrique additive)

Où *k* est le cluster, *P* étant le chemin de la source *s* vers la destination *d*.

- Dans le cas de mesures multiplicatives, le poids d'un cluster représente le poids maximal des chemins de n'importe quelle source vers n'importe quelle destination à l'intérieur de ce cluster.

$$W(k) = Min(\prod_{(u,v)\in P} w(u,v)).....(1)$$

Équation 2: Poids d'un cluster (métrique multiplicative)

Où *k* est le cluster, *P* étant le chemin de la source *s* vers la destination *d*.

Le poids multiplicatif *W* (*k*) dans ce cas peut être exprimé en poids additif en passant au logarithme, *W* (*k*) sera noté alors $\tilde{W}(k)$. On aura donc :

$$\tilde{W}(k) = Max(\sum_{(u,v)\in P} \log(w(u,v)).....(2)$$

- Dans le cas de mesures concaves, le poids d'un cluster représente le poids minimal des chemins de n'importe quelle source vers n'importe quelle destination à l'intérieur de ce cluster.

$$W(k) = \underset{(u,v)\in P}{Min}(w(u,v)).....(3)$$

Équation 3: Poids d'un cluster (métrique concave)

Où *k* est le cluster, *P* étant le chemin de la source *s* vers la destination *d*.

Lors de la recherche des clusters qui répondent aux critères de QoS concaves, il suffit de procéder par filtrage en supprimant les liens qui ne répondent pas à la demande.

De (1), (2) et (3), on peut supposer, donc, de manière générale, que toutes les métriques sont additives. Nous pouvons à présent énoncer le problème.

Définition: QoS-based clustering problem (QCP). Soit un réseau $G(M,E)$ tels que *M* est l'ensemble de nœuds, *E* est l'ensemble des liens. Chaque lien $(u,v)\in E$ est pondéré par un vecteur de poids $w_i(u,v)\geq 0, i=1,2,...m$. Soit *B* l'ensemble des nœuds de bordures du réseau *G*. Soit *m* constantes $L_i, i=1,2,...m$ qui représentent les mesures de QoS effectuées par *POrg*. Soit K l'ensemble des clusters potentiels. Soient *s* un nœud source et *d* un nœud destination dans le réseau *G*. *P* étant le chemin de la source *s* à la destination *d*. Le problème est de trouver l'ensemble de clusters *K* qui vérifie la propriété suivante:

$$\forall s,d\in B, \sum_{k\in K} W_i(k)\leq L_i.....(4)$$

Équation 4: QoS-based clustering problem

Où L_i est une mesure effectuée de bout en bout du réseau G.

De manière plus étendue, il s'agit de trouver l'ensemble de clusters *K* qui vérifie la propriété suivante:

$$\forall X\in K, \forall s,d\in X, \sum_{(u,v)\in P} w_i(u,v)\leq L_i.....(5)$$

Équation 5: Extended QoS-based clustering problem

Où L_i est une mesure effectuée de bout en bout du cluster *X*.

Ce problème peut être assimilé au problème de clustering [78], [79] qui définit le processus d'organisation d'objets en des groupes partagent une notion d'équivalence.

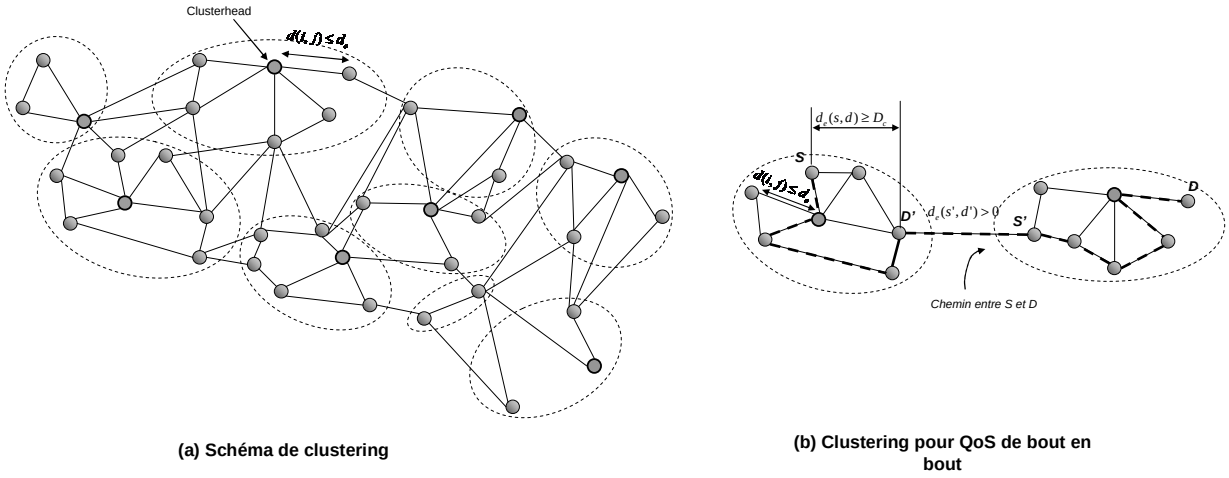
Afin de bien cerné le problème et étudier sa résolution à travers des méthodes de clustering, nous avons dressé un état de l'art des différentes techniques de clustering que vous trouverez en détails dans l'Annexe A. Cependant, nous présenterons une synthèse de celles-ci afin de montrer leurs apports et limites par rapport à notre problématique.

Clustering pour la QoS horizontale

Comme nous l'avons cité ci-dessus, le clustering définit le processus de partitionnement d'un ensemble d'entités de départ en domaines (clusters), tout en ayant (dans un cas idéal) une organisation de chaque cluster composée d'éléments fortement intrinsèques. Cette similitude peut se définir par une notion de proximité par rapport à une (ou plusieurs) métrique(s) particulière (s) comme étant le point commun à partager.

Toutes les méthodes de clustering se basent sur le principe suivant (avec quelques différences que nous présentons dans le Tableau 1) :

Soit un graphe $G(M, E)$ tel que M est l'ensemble de nœuds, E est l'ensemble des liens. On a besoin de le partitionner en k clusters, où $k<|M|$. Soit i le centre d'un cluster C (appelé parfois *clusterhead*), le nœud j est un membre du cluster C si et seulement si $d(i, j) \le D_c$, où $d(\cdot)$ est la distance entre i et j par rapport à une métrique donnée (appelée par fois *distance de couverture* du cluster) et D_c est la distance maximale tolérée pour ce cluster. Un exemple de résultat du clustering d'un réseau G est illustré dans la Figure II-8(a).



(a) Schéma de clustering

(b) Clustering pour QoS de bout en bout

Figure II-8: Schéma du clustering

Le Tableau 1 présente une synthèse des différentes méthodes de clustering susceptibles de répondre à notre problématique.

Rappelons que pour notre problématique de clustering, nous sommes appelés à structurer un réseau en domaines régissant un (ou plusieurs) paramètres de QoS que nous pouvons contrôler et maîtriser aux bordures de ces domaines.

Afin de résoudre cette problématique et maîtriser le bout en bout horizontal, l'algorithme de clustering à utiliser doit surmonter les contraintes suivantes :

- La distance effective à l'intérieur d'un cluster doit être maîtrisée (Figure II-8(b)). Nous appelons distance effective d'un cluster $(D_{c_e}(C))$, la distance maximale entre tout nœud *s* et tout nœud *d* appartenant au même cluster *C*. $D_{C_e}(s,d) = Max(\sum_{(s,d,\in C)} w(s,d))$ où *w(.)* est la distance entre *s* et *d*.
- La distance inter-cluster soit maîtrisée. C'est-à-dire que la distance (QoS) entre les clusters doit être connue et quantifiable.

Le tableau suivant présenté les différentes méthodes de clustering en se basant sur six (6) angles :

- les paramètres d'entrée
- les paramètres de sortie
- le critère d'optimisation
- la fonction objective
- l'utilisation ou non d'un clusterhead
- la maîtrise des chemins à l'intérieur d'un cluster : est ce qu'on peut dire que la distance entre n'importe quels deux nœuds est inférieure à D_c ?
- la maîtrise des distances inter-clusters

		Caractéristiques						
		Paramètres d'entrée	Paramètres de sortie	Paramètre d'optimisation	Objectif	Clusterhead	Maîtrise chemin à l'intérieur	Distance inter-cluster
Méthodes	*Single link clustering*	Graphe, degré de similitude	Les clusters et leur localisation	distance	Minimiser le nombre de cluster	Non	Non	Oui
	Complete link clustering	Graphe, degré de similitude	Les clusters et leur localisation	distance	Minimiser le nombre de cluster	Non	Non	Oui
	K-means	Graphe, degré de similitude	Les clusters et leur localisation	Erreur quadratique	Minimiser le nombre de cluster	Oui	Non	Oui
	MST	Graphe, coût	Les clusters et leur localisation	Coût	Minimiser le nombre de cluster	Non	Non	Oui
	Set Covering	Graphe, la distance d_c	Les clusters et leur localisation	distance	Minimiser le nombre de clusters (k)	Oui	Non	Oui
	K-center	Graphe, le nombre de cluster k	Les clusters et leur localisation	distance	Minimiser la distance	Oui	Non	Oui

Tableau 1: Synthèse des méthodes de clustering

Nous relevons, dans ce tableau, une insuffisance liée intrinsèquement à ces modèles « classiques » de clustering qui se base sur un représentant significatif des clusters dans le réseau (clusterhead).

En effet, par construction même des clusters (§Annexe A), nous remarquons que la relation qui définit un cluster est dépendante du centroide de ce dernier. De ce fait, aucune maîtrise des distances effectives à l'intérieur du cluster n'est prise en compte, ce qui compromet fortement la tâche de contrôle de la QoS et de la fonction d 'équivalence pour le routage. En d'autres termes si nous voulons vérifier un paramètre de QoS à la sortie du cluster nous devons obligatoirement considérer les flux qui passent par le centre et donc les liens reliant les nœuds d'un cluster.

L'autre limite est aussi la visibilité des distances inter-clusters (Figure II-8(b)). La QoS de bout en bout est le résultat de *coopération* des différents domaines composant le réseau. La

distance entre ces domaines est un élément à ne pas négliger pour maîtriser la QoS de bout en bout *horizontal*.

Arrivant à ce stade de l'analyse, nous pensons qu'il est indispensable de revoir la vision traditionnelle du clustering. Il est évident qu'une application directe de ce qui se trouve dans la littérature n'est pas en adéquation avec nos objectifs.

Dans le paragraphe suivant, nous présentons notre proposition d'algorithme de clustering qui peut répondre à notre problématique. Cet algorithme est élaboré pour $m = 1$ de *QCP* (§5.2.4).

Le Path-based Clustering (PBC)

Nous avons constaté, ci-dessus, que parmi les insuffisances rencontrées dans les approches de clustering, est l'assimilation du cluster à un élément central autour duquel tourne toute la construction du cluster et de la fonction d'équivalence. Or dans notre cas de figure nous ne voulons pas instaurer des solutions vérifiables uniquement par rapport au centre du cluster. Ce qui nous incite à orienter notre réflexion vers le regroupement d'entités qui décrivent au mieux ce que nous voulons maîtriser.

Un transfert pour nous n'interagit pas simplement avec des nœuds élémentaires mais s'appuie plutôt sur la notion de chemin et donc de lien formant un réseau[5]. En effet un flux réseau empreinte des chemins qui sont des successions de nœuds et de liens réseau, ce qui nous amène à envisager un clustering qui préconise plutôt le regroupement de nœuds et liens logiques (chemins potentiels) plutôt que le regroupement de nœuds seulement. Cette approche nous semble plus appropriés d'autant plus que si nous nous mettons dans un contexte de contrôle de QoS aux bordures des clusters, la relation d'équivalence que nous chercherons à établir au sein d'un même cluster sera des chemins fortement similaires d'un point de vue QoS (ex : des chemins partageant le même délai de traversée du bout en bout du cluster).

Le regroupement de chemins à l'intérieur de chaque cluster suivant les chemins effectifs nous permet d'avoir une maîtrise locale de la QoS sur les éléments couverts par les clusters.

Quant à l'espace en dehors des clusters nous ne sommes pas en mesure d'affirmer quoi que ce soit sur le contrôle du paramètre QoS choisi. Cet aspect permet de dégager la deuxième spécificité du schéma de clustering que nous devons construire, à savoir un espace inter

[5] D'où la puissance du modèle NLR (§4.2.1)

clusters réduit à l'ensemble vide (Figure II-8(b)). Ce qui renvoie à la juxtaposition des clusters dans le niveau réseau comme le montre la Figure II-7.

La méthode de clustering adéquate à l'objectif de contrôle de la QoS de bout en bout à mettre en œuvre, est une méthode qui définit :

- Un clustering qui se base sur le regroupement de chemins équivalents de point de vue distance effective (QoS).
- Une juxtaposition des clusters afin de réduire l'espace inter clusters à l'ensemble vide.
- Un clustering sensible à la vision des interfaces pour concrétiser la notion de chemin.

Afin de comprendre notre mécanisme de clustering, considérons la figure suivante (Figure II-9). Dans la topologie suivante le nœud *a* représente la source d'un flux de données qui parcoure le réseau pour atteindre le nœud *s*, nœud destination du flux.

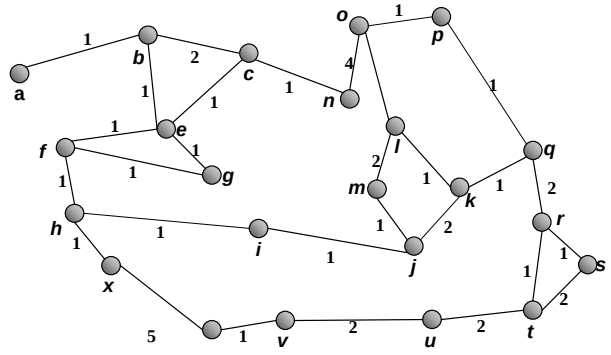


Figure II-9: Nœuds/Liens/Réseau

- *Principe de l'algorithme*

Afin d'avoir le schéma de clustering souhaité, nous allons procéder en trois (3) phases : phase d'expansion, phase de redimensionnement (resizing) et phase de relance.

- **Etape 1 :** Phase d'Expansion

Dans cette étape nous choisissons un nœud de bordure du réseau (nœud périphérique, *NP*). A partir du nœud choisi nous ferons une couverture de tous les chemins d'une distance inférieure ou égale à la distance de clustering choisie (D_c). Nous utiliserons pour cela notre variante de l'algorithme *DFS (Depth First Search)* [80] que nous appelons *Q-DFS* (*QoS-based DFS*). Cette phase va renvoyer l'ensemble des nœuds couverts par cette distance à partir du nœud périphérique. Ces nœuds formeront le cluster *préliminaire*.

Cette démarche aboutit à des nœuds que nous désignerons par « nœuds bordures, *NB* » vu qu'ils bordent le cluster *préliminaire*.

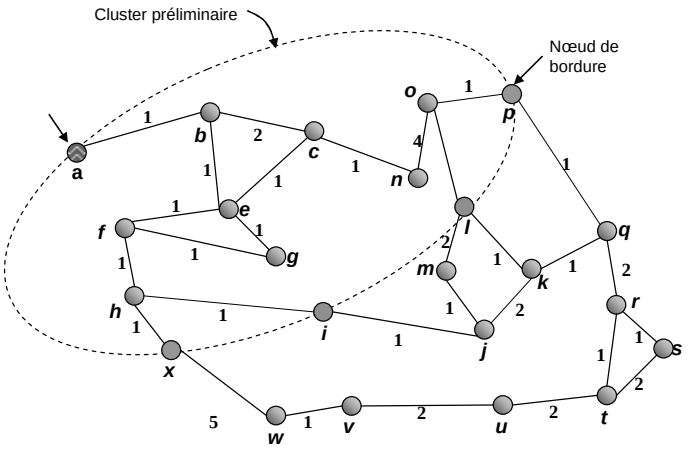


Figure II-10: Formation d'un cluster préliminaire

Ce cluster préliminaire va donc cerner tous les chemins d'un délai inférieur à D_c dans le sens nœud périphérique vers les nœuds de bordures. La Figure II-10 montre cluster préliminaire en prenant comme nœud périphérique de départ le nœud a et comme distance de clustering $D_c = 9$.

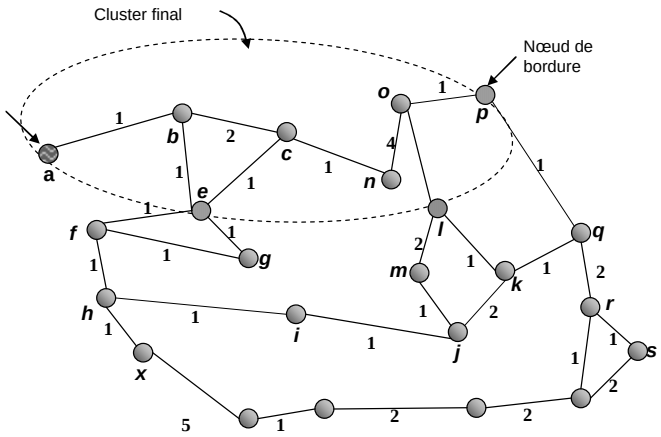
- **Etape 2 :** Phase de Dimensionnement

Durant cette phase nous allons tenter de redimensionner le cluster préliminaire de tel sorte que nous n'ayons plus un nœud périphérique et des nœuds bordures mais plutôt un ensemble de nœuds ES (Entrée / Sortie) d'un cluster final. Le but de cette démarche et de redessiner le cluster pour éviter qu'il existe un chemin dont la distance entre les NBs soit supérieure à D_c.

Pour cette phase de redimensionnement nous procédons de la façon suivante :

- ⇨ Pour chaque nœud bordure établi lors de la phase d'expansion, nous vérifions s'il peut couvrir l'ensemble des autres nœuds NB ($dist \leq D_c$).
- ⇨ Nous déterminons pour chacun l'ensemble des nœuds bordures qu'il peut couvrir.
- ⇨ Afin de redimensionner le cluster, nous éliminons du cluster le nœud qui couvre le moins de nœuds bordures.
- ⇨ Nous réitérons ces actions jusqu'à ce que nous obtenions un ensemble cohérent de nœuds *NBs*. L'ensemble cohérent est un ensemble où tous les NB se couvrent mutuellement, c'est-à-dire que la distance entre les NB ne dépasse pas D_c.

La figure suivante montre le redimensionnement pour le cluster préliminaire précédant :



Une fois le redimensionnement fini, nous passons à la phase de relance que nous présentons dans ce qui suit.

- **Etape 3 :** phase de relance

Dans cette phase nous allons relancer les deux précédentes étapes à partir des nœuds de bordure du cluster final (les nœuds de bordure deviennent des nœuds périphériques).

Les nœuds ES seront donc les jonctions entre les clusters. Le processus de clustering se termine quand nous arrivons à clustériser tous les éléments. La figure suivante montre le résultat final.

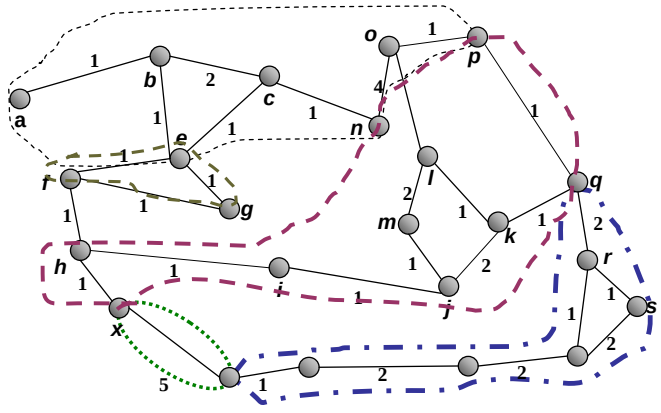


Figure II-11: résultat du QoS-based clustering (QCP)

- *L'Algorithme Q-DFS (QoS-based DFS)*

Afin de résoudre le problème QCP, nous nous sommes basés sur une modification de l'algorithme de recherche en profondeur *DFS* (*Depth First Search*) [80] que nous proposons de présenter dans ce paragraphe. Nous avons nommé cette variante « *QoS-based DFS* (*Q-*

DFS) ». Il s'agit de l'amélioration de DFS pour supporter la recherche de chemins répondant à un critère de QoS à partir d'un nœud de départ (Figure II-12).

Soit $G(M,E)$ le réseau de niveau *N* considéré (équipement, réseau, service et usagers), où *M* est l'ensemble des nœuds du réseau et *E* l'ensemble des liens. Chaque lien est caractérisé par un poids *w* représentant un paramètre de QoS à surveiller ($(w(u,v),(u,v)\in E)$). Soit un nœud *r* représentant le nœud de démarrage de la recherche (sommet de l'arbre).

L'objectif est de trouver un cluster (ensemble de nœuds) *C* tel que $\forall e \in C, w(r,e) \le d$ où *d* est le poids effective du cluster, *e* et *r* sont deux nœuds du cluster *C*.

L'ensemble des nœuds de bordure de ce cluster est représenté par l'ensemble *B*.

```
C := {};
B := {};
Algorithme Q_DFS(G, v, d, r)
Input : graphe G, le sommet de démarrage v de G, la distance
d de clustering souhaitée et le nœud r de référence pour la
calcul de la distance.
Output : ensemble E qui contiendra les noeuds appartenant au
cluster préliminaire et B contiendra les nœuds de bordure de ce
cluster

Marquer (v);
C := C U {v};
B := B U {v};
Pour tout e ∈ Voisins(v)
        Si e non_visité et w(e, r)≤d faire
                B := B-{v}
                Q_DFS (G, e, d, r);
        Fin
Fin
Return E, B
```

Figure II-12 : L'algorithme Q-DFS

C'est l'algorithme Q-DFS qui est utilisé dans la première phase du processus *QCP*, à savoir, la phase d'expansion. Il sera utilisé comme un module appelé dans le cœur de l'algorithme QCP que nous présenterons dans ce qui suit.

- *L'algorithme QCP*

```
NP;
Algorithme QCP (G, v, d)
Input: graphe G, le sommet de démarrage v de G et la distance d de clustering
souhaitée.
Output: le graphe G clusterisé

Si tous les nœuds son visités Faire
        return;
Sinon
        {E, B} := Q_DFS (G, v, d, NP); //E étant l'ensemble des nœud appartement
                                        à ce cluster et B étant l'ensemble des
                                        noeuds de bordure de ce cluster
        Tant que (l'ensemble bordure B n'est pas cohérent) faire
                Pour tout p ∈ B faire
                        i := nombre de NB atteint à partir de p tel que d(i, NB) ≤d;
                Fin
                Min := NB qui atteint le moins de nœuds bordures
                B := B - {Min}
        Fin // nous avons un cluster redimensionné.
        Pour tout e ∈ B faire
                QCP (G, e, d);
Fin
Fin #
```

Figure II-13: L'algorithme QCP

Les fonctions exécutées par chaque agent relèvent de la dimension fonctionnelle que nous présentons dans le paragraphe suivant.

5.3 Le modèle fonctionnel

Après avoir défini le rôle, la portée et l'emplacement de chaque entité composant la plate-forme *POrg* dans l'architecture ainsi que les techniques pour réorganiser les éléments du réseau dans la section précédente, nous définirons dans cette section l'ensemble des fonctions et actions à mettre en œuvre par *POrg* pour assurer l'ajout de nouveaux services et maintenir la QoS de bout en bout de manière transparente et ainsi assurer l'intégration verticale et horizontale.

L'une des missions pour l'intégration verticale est le contrôle de l'harmonisation à des fins de cohabitation des contrats de QoS de chaque niveau. POrg doit, donc, assurer le *contrôle* de la QoS des nouveaux flux/service entrant/traversant le réseau (sur tous les niveaux de visibilité) ce qui nous permet de dégager la première fonction principale de POrg. Afin d'être le plus réactif possible, l'architecture doit être *adaptive* aux changements dans le réseau et à l'état de ses composants, par conséquent, cette *auto-adaptation* est la deuxième fonction qui doit être assurée par le POrg. Cette auto-adaptation adaptation traduit la coopération de la mission intégration horizontale. Elle pourrait aboutir, parfois, à une *reconfiguration* des contrats de QoS (inter-domaines). La *reconfiguration* représente la troisième fonction principale de *POrg*, afin d'assure la QoS de bout en bout.

Avant de détailler ces fonctions, nous présentons, tout d'abord, l'ensemble des acteurs avec lesquels chaque agent *POrg* interagit (§5.3.1). Nous détaillons ensuite les différentes fonctions assurées par chaque agent *POrg* pour un pilotage adéquat de l'organisation de l'architecture dans le niveau réseau (§5.3.2). Notons que ces fonctions génériques sont les mêmes quelque soit le niveau considéré. La description des acteurs et fonctions nous permettra, dans le paragraphe §5.3.4 de décrire les processus de déclinaison et agrégation (facteurs de l'intégration verticale) et de donner quelques scénarii pour bien illustrer l'ensemble des processus.

Le modèle fonctionnel que nous présentons dans cette section représente, donc, les fonctions assurées par un agent *POrg*. Cette représentation est faite de manière générique, chaque agent *POrg* (L1 et L2) est une instanciation de ce modèle.

5.3.1 Acteurs

Notre objectif est de recenser tous les acteurs qui interagissent avec un agent POrg pour définir les fonctions à lui attribuer (Figure II-14). Nous avons regroupé l'ensemble de ces acteurs en trois catégories : acteurs actifs, acteurs passifs et acteurs de gestion.

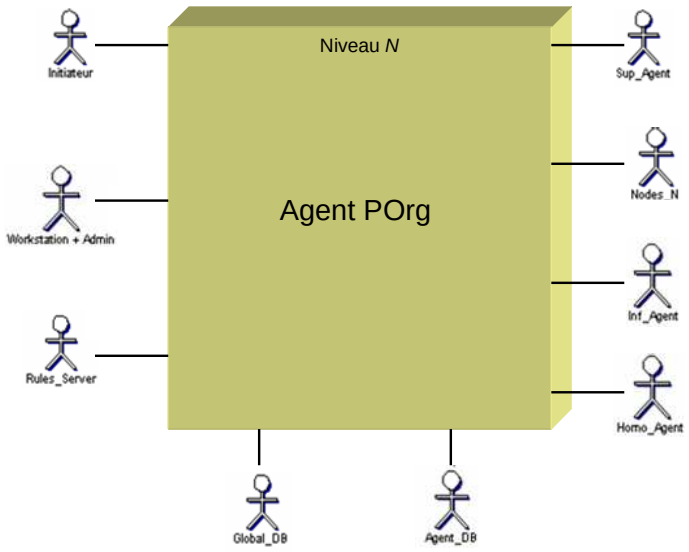


Figure II-14: Les acteurs de POrg

Acteurs Actifs

Les acteurs actifs sont des acteurs qui déclenchent les fonctions des agents *POrg*. Nous avons défini cinq (5) types d'acteurs : initiateur, nœud (routeur, serveur, équipement, etc.), agent de niveau *N*+1, agent de niveau *N*-1 et agent de niveau *N* (horizontal).

- *Initiateur*

Comme mentionné ci-dessus, le POrg est chargé de contrôler la QoS des nouveaux services (flux). Afin d'accomplir cette tâche, il a besoin de connaître la logique et les besoins QoS du service transitant par le réseau. Ces caractéristiques sont détenues par un moteur de workflow qui sollicite POrg en envoyant ces caractéristiques comme paramètre. La demande est définie à travers une requête explicite à POrg de la forme : « Service = svr, Objet= obj, Characteristic = chr, QoS = DFDC », où :

  - Service est le nouveau service à exécuter (audio, streaming, transfert, etc.),
  - Objet est la nature le l'objet manipulé par le service (audio pour le service streaming, SMS pour le service transfert, etc.),
  - Characteristic est la caractéristique de l'objet manipulé (taille, format, etc.) et

- QoS qui exprime les besoins QoS (contrat QoS) de ce service au format DFDC.

- *Nœud*

Comme mentionné ci-dessus, le POrg est chargé d'auto-adapter et de reconfigurer l'architecture suivant l'état et les changements dans le réseau. Ces changements sont détectés à travers les notifications provenant des nœuds ainsi que la lecture des tables de routages. Cette interface prend en charge la communication entre POrg et les nœuds (routeurs, serveurs, etc.). Le POrg peut aussi, à travers cette interface, commander, activer, désactiver, reconfigurer un composant dans un nœud donné.

- *Agents POrg de niveaux N+1/N-1*

Ces interfaces sont réservées aux agents L1. Elles leur permettent de communiquer entre eux pour les processus de déclinaison et d'agrégation (voir §4.3.1).

- *Agents POrg de niveau N*

Cette interface est activée pour tous les agents POrg. Elle définie l'interaction entre les agents POrg de même niveau à des fins de maintien de QoS horizontale et, éventuellement, réorganisation des domaines (voir §5.2.4).

Acteurs Passifs

Les acteurs passifs, quand à eux, permettent aux agents *POrg* de bien mener leurs fonctions. Ils jouent un rôle d'auxiliaires. Nous avons spécifié deux (2) types d'acteurs : serveurs de règles et les entités de stockage.

- *Serveur de règles*

En présence de situation où le POrg ne détient pas de règles internes traitant une nouvelle situation (nouveau flux, état inconnu du réseau, etc.) il lui faut de nouvelles règles dites, d'ingénierie. Ces règles sont préalablement configurées dans un serveur de règles.

- *Entités de stockage*

Le POrg a besoin de stocker les informations relatives à l'état du réseau (base de données locale) ainsi qu'à son fonctionnement propre afin de restaurer la configuration initiale au cas d'échec du processus de paramétrage (base de données globale). Les agents POrg utilisent deux types d'entité de stockage :

- Base de données locale : base de données propre aux agents POrg L2, elle détient les informations sur les flux de bout en bout du niveau *N* ainsi que la description et le comportement des éléments de ce niveau. Cette BDD est utilisée principalement pour le maintien et la vérification du contrat de QoS ainsi que la trace du processus de déclinaison.
- Base de données globale : il s'agit des bases de données utilisées par le système de gestion (OSS/SI). L'interaction avec ces bases de données permet aux agents POrg:
  - d'être au courant des changements dans l'architecture globale
  - de limiter leurs responsabilités et se positionner comme module supplémentaire au système de gestion, s'occupant principalement de la QoS.

Acteurs de Gestion

Ces acteurs interagissent avec le POrg à des fins de gestion du POrg lui-même. Ils sont de deux types : administrateur et poste de travail.

- *Administrateur*

L'interface administrateur représente l'autorité administrative qui peut forcer le POrg à exécuter certaines tâches non prévues par les règles appliquant ainsi des choix purement stratégiques. Il peut aussi changer l'état du POrg (le désactiver, activer).

- *Poste de travail*

Cet acteur agit principalement comme interface de gestion et d'administration du POrg.

Après présentation des acteurs qui peuvent interagir avec POrg, nous pouvons maintenant présenter les différentes fonctions du POrg.

5.3.2 Les fonctions de POrg

Cette section détaille les fonctions des agents *POrg* que nous avons regroupé par rapport à leurs objectifs. En fait, la complexité de *POrg*, qui résulte de la coopération de plusieurs fonctions induit une spécification suivant une décomposition homogène des fonctions à réaliser.

C'est à cette tâche que nous nous sommes livrés dans cette section en utilisant une modélisation en diagrammes de Cas d'Utilisation (Use Cases) d'UML [81].

En effet, UML (Unified Modeling Language) est un langage, entendu comme un ensemble organisé de composants (classes, objets messages, interactions, évènements….) régi par des règles de grammaire strictes, permettant de représenter et de spécifier un système complexe de façon formelle, favorisant la réutilisation de classes et d'objets.

Nous avons décomposé les fonctions à réaliser en deux niveaux : le premier niveau représente les fonctions globales de POrg, le deuxième niveau détaille chaque fonction du premier niveau. Le diagramme de Use Case que nous présentons dans ce qui suit (Figure II-15) résume les fonctionnalités (de niveau I) que chaque agent pourrait accomplir. Les fonctionnalités principales se résument en : *Contrôler* la QoS qui se charge de la déclinaison des besoins QoS et le maintien de cette QoS, et *Auto-adapter* qui réagit aux changements (structurels et conditionnels) dans l'état du réseau, et *Reconfigurer* qui se charge de toutes les interactions avec les nœuds du réseau à des fins de changement de paramètres/configuration.

Afin d'être le plus transparent possible, les agents POrg implémentent des fonctions d'interfaçage avec les acteurs que nous avons cité précédemment (dans §5.3.1). Ces fonctions seront présentées dans la troisième partie de ce rapport (partie implémentation)

Dans ce qui suit, nous détaillons chacune de ces fonctionnalités à travers les Use Case dit de niveau II (zoom des Use Cases du premier niveau).

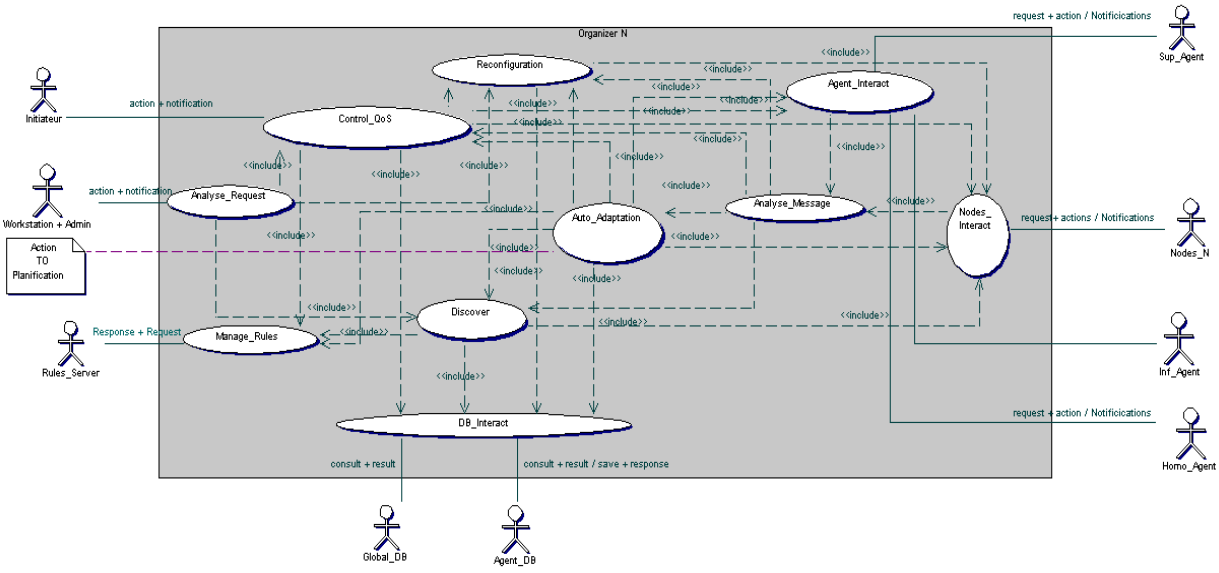


Figure II-15: Diagramme de Use Case d'un agent POrg

Le Cas d'utilisation « Control QoS »

Le contrôle de QoS concerne l'ensemble des ressources de l'architecture et englobe celui des performances et des admissions. En effet, le bout en bout s'entend, d'une part,

- de haut en bas de l'architecture du réseau, c'est-à-dire la superposition des blocs architecturaux du réseau communiquant,
- et d'autre part, de gauche à droite ou/et de droite à gauche, c'est-à-dire la juxtaposition des réseaux de communication (intra et inter domaine). C'est d'ailleurs dans le but d'une QoS de bout en bout à travers une cohabitation verticale entre les différents niveaux architecturaux et une coopération horizontale entre les éléments d'un niveau donné que les agents POrg opèrent. Ils sont donc responsables du contrôle de cette QoS en terme de cohérence des paramètres instanciés qui est assuré à travers le Use Case QoS_Layer_Control, de continuité de service par la fidélité des performances (Network_Performance_Control) et le maintien de la QoS horizontale et de la régulation des admissions (Admission_Control) qui contrôle l'admission de nouveaux flux/services au réseau. Ces trois Use Cases composent la fonction de Contrôle_QoS.

Pour résumer, nous avons divisé la fonction de contrôle de QoS en deux parties:

- Fonctions verticales : s'occupant de la détermination du choix architectural à effectuer suivant les besoins en QoS des niveaux supérieurs. Ce choix se traduit par la sélection des mécanismes qu'il faut faire intervenir pour satisfaire la QoS du service demandé (choisir d'activer la classification, shaping, etc. sur un routeur du réseau par exemple). La fonction verticale est représentée à travers le Use Case «Layer_QoS_Control ».
- Fonctions Horizontales : s'occupant du maintien du niveau de service et la consommation des ressources du réseau (intra et inter domaines). Pour cela, nous avons prévu deux Use Cases que nous désignons par : «Admission_Control» et «Network_Performance_Control». Le Use Case Admission_Control control l'accès au réseau/nœud pour les flux ne satisfaisant pas les critères requis (débit du flux demandé est grand par rapport à la bande passante réservable du réseau/nœud par exemple). L'Use Case Network_Performance_Control contrôle l'accès au routeur par rapport à ses ressources internes (CPU, mémoire, etc.). Un flux est admis si la capacité de traitement du routeur est suffisante.

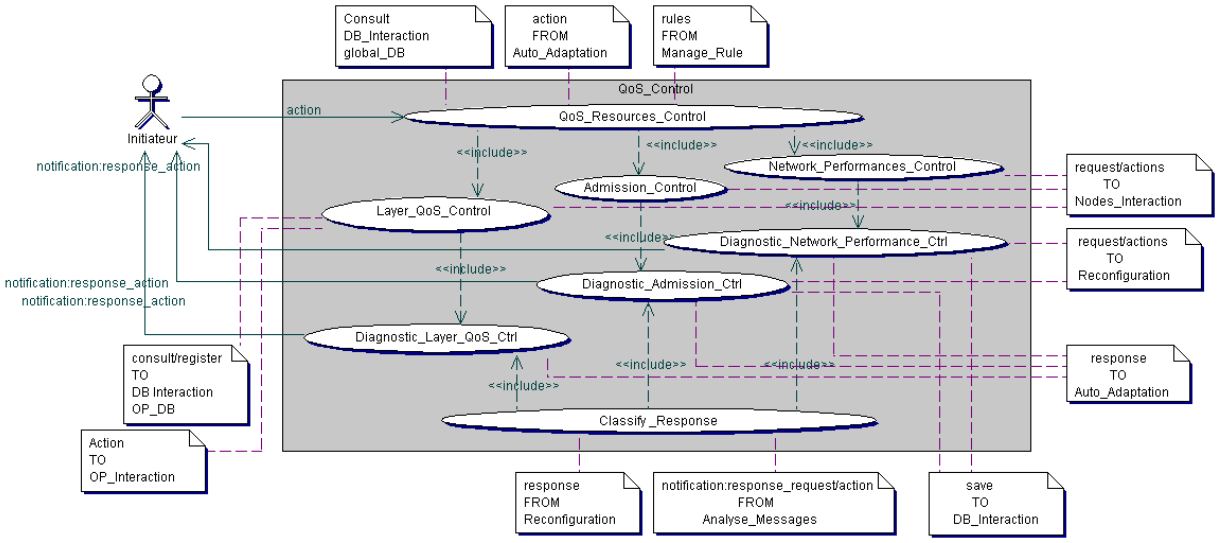


Figure II-16: Diagramme de Use Case Control_QoS

Le module de Control de QoS est invoqué de plusieurs façons, il peut être invoqué statiquement à travers l'Initiateur ou dynamiquement à travers l'Use Case d'auto-adaptation suite à des changements de l'état du réseau. Toutes ces invocations sont traitées par le module de Control_QoS_Resources qui détermine le module destinataire de la requête.

L'analyse des notifications retournées par les nœuds et les autres Uses Cases du premier niveau est assurée par les Uses Cases de diagnostique (Diagnostic Layer QoS Ctrl, Diagnotic Admission Ctrl et Diagnostic Network Perf Ctrl) qui retournent des rapports à l'Initiateur du déroulement de l'opération.

Le Cas d'Utilisation « Auto adaptation »

L'auto-adaptation est définie comme étant la capacité de réagir aux variations des contraintes ou des besoins existants. Ce besoin d'auto-adaptation est ressenti du fait de l'évolution des architectures actuelles. Rares en effet sont les architectures dont la conception s'arrête au moment du déploiement. Ainsi, suite à la composition (construction et production) de telles architectures, les modifications durant leur cycle de vie sont prises en considération par la capacité d'adaptation qui peut être autonome (auto-adaptation). Ces modifications ne sont aucunement arbitraires mais plutôt dirigées et contrôlées. De plus, les réseaux supports sont des réseaux multiservices dont la distribution (de l'utilisation) dans le temps et dans l'espace varie beaucoup en fonction des usages, des évènements et des lois qui régissent le marché de la consommation. Il faut donc le plus de dynamicité et adaptabilité possible.

Son diagramme de cas d'utilisation est représenté sur la Figure II-17. Comme cette fonction peut être initiée à la suite de changements remarqués sur son environnement, une fonction "réagir aux changements" est réalisée afin de s'adapter aux dits changements. Ensuite, le processus d'auto-adaptation pourra être créé et envoyé aux nœuds du niveau concerné afin qu'ils se mettent à jour. Ici aussi un suivi des modifications est à faire pour avoir une bonne auto-adaptation. Enfin, la base de données est mise à jour suite aux modifications d'auto-adaptation effectuées.

Nous détaillons les fonctions d'Auto-Adaptation par un diagramme du cas d'utilisation qui décrit ses fonctionnalités plus en détail (Figure II-17).

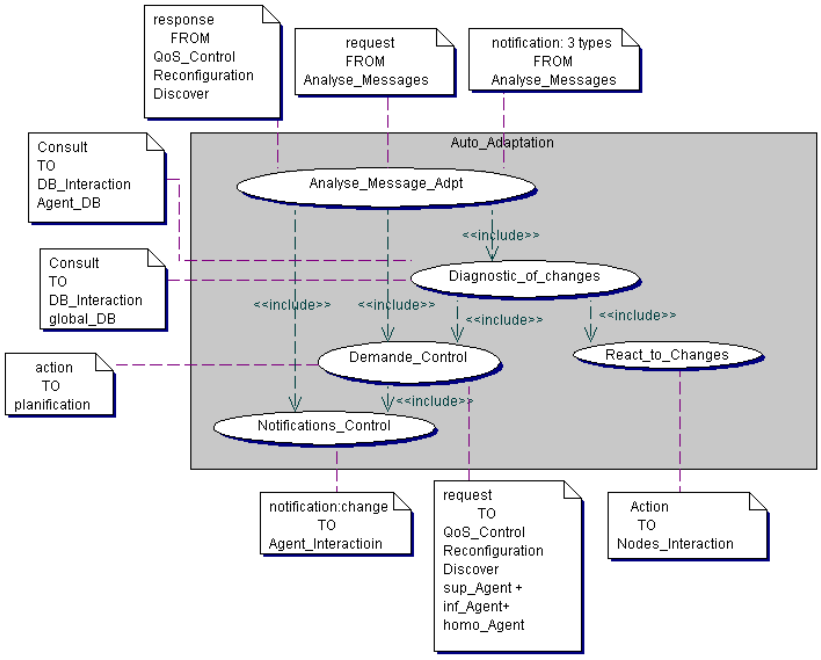


Figure II-17: Diagramme de Use Case de Auto_Adaptation

La fonction d'Analyse_Message_Adpt trie les messages en provenance des autres Use Cases. Suivant ces messages, cette fonction aguille la requête à la fonction compétente :

- s'il s'agit de notifications en provenance du Use Case Analyse_Message, le message est aiguillé vers le Use Case Diagnostic_of_Changes. Ce dernier analyse le message, organise les actions à effectuer et aiguille le message vers React_To_Changes pour effectuer les configurations nécessaires.
- dans le cas où le Use Case Auto_Adaptation est invoqué mais aucune reconfiguration n'est nécessaire, le message est aiguillé vers le Use Case Demande_Control pour l'envoyer au

niveau supérieur (pour qu'il déclenche le processus de déclinaison et agrégation que nous dériverons plus loin (p.97))

- s'il s'agit d'une fin d'opération, le message est aiguillé vers Notification_Control afin d'envoyer le message aux autres agents.

5.3.3 Processus d'évaluation de la QoS

Après avoir décrit les acteurs qui interagissent avec les agents POrg et les fonctions que doit assurer chaque agent, nous présenterons dans ce qui suit les processus de déclinaison et d'agrégation des besoins QoS que nous proposons. Il s'agit du processus mettant en action les différents acteurs et fonctions présentées précédemment.

Nous présentons ce processus à travers des étapes allant du SLA à la configuration des éléments du réseau pour le *contrôle de la QoS* et *l'auto-adaptation*, les deux (2) fonctions les plus importantes des agents POrg (comme on va le présenter dans §5.3.2).

La réservation de ressources (le provisioning) permet, en temps réel, de partager les ressources existantes selon un prévisionnel, qui ne tient pas compte en général de la QoS. Il ne tient pas compte non plus des aléas de l'exploitation. Mais si nous voulons respecter nos objectifs et les contrats passés avec l'utilisateur, nous devons suivre le comportement des flux, c'est-à-dire la QoS. Les réactions à ce suivi, doivent aboutir à une gestion dynamique de cette QoS.

Ces considérations nous ont fait adopter un processus générique de déclinaison/agrégation que nous avons dénommé « *DECIDERA* » qui définit les tâches de *POrg* en huit (8) étapes et neuf (9) actions (Figure II-18).

Le but de cette approche est d'identifier les paramètres de QoS qui influencent significativement la perception de la QoS par l'utilisateur final, selon l'architecture, l'environnement en cours d'exploitation et d'après les règles prédéfinies. Le tableau suivant présente les différentes étapes de l'évaluation de la QoS.

Etape	Action	Signification
D	*1. Caractérisation*	Identifie les paramètres concernant la QoS dans le SLA.
	2. Sélection	Sélectionne les paramètres significatifs qu'il faudra mesurer pour évaluer les critères de QoS
E	*3. Extraction*	Extrait les règles à appliquer en fonction du contexte
C	*4. pré-Configuration*	Prédéfinit les modifications de paramétrage à faire pour garantir l demande de QoS
I	*5. Identification*	Identifie le vecteur d'agrégation
D	*6. Descente*	Amorce le changement de niveau
E	*7. Evaluation*	Evalue la situation, analyse le vecteur d'agrégation
R	*8. Réaction*	Décide des choix et réagit en conséquence
A	*9. Activation*	Active toutes les actions décidées en remontant tous les niveaux.

Tableau 2: Etapes d'évaluation de la qualité de service : DECIDERA

1. Afin de spécifier complètement un service, il est nécessaire de considérer les propriétés non fonctionnelles (critères de QoS) associées à ce service. Ces propriétés, désignées sous le nom de caractéristiques (*1.caractérisation*) du service, constituent l'aspect quantifiable qui n'est autre que « la QoS ». l'objectif étant d'extraire les paramètres à calculer du SLA du service.

2. Cette première étape de Définition comporte une deuxième action de -*2.sélection*-. En effet, tous les paramètres comportementaux mesurables sont identifiés à partir des indicateurs du SLA et les informations contenues dans la DB Globale (mesures disponibles)

3. On extrait (*3.Extraction*) les règles appliquées sur ces paramètres mesurables et qui sont propres au service (SLA). Ces règles sont de type : valeurs seuil, valeur max, etc.

4. et déterminer les valeurs seuils du critère associé, si nécessaire, (*4.pré-Configuration*).

5. Pendant la pré-configuration, l'agent POrg n'effectue pas d'actions vers le réseau (niveau considéré), il identifie (*5.Identification*) le vecteur d'agrégation (Flag). Ce vecteur d'agrégation représente le résultat final de décision prise par les agents sur l'architecture globale (différents niveaux). Flag est composé des valeurs (0,1) qui indiquent si des changements (reconfigurations) ont été programmées pour un niveau N (Flag(N)=1) ou non Flag(N)=0). Ce vecteur sera transmis de niveau en niveau (entre agents L1) et indique qu'une décision de reconfiguration a été prise et qu'il faut les exécuter lors de

l'agrégation. Les paramètres préconfigurés vont être enregistrés dans la DB POrg avec la valeur du Flag.

6. Puis le passage au niveau inférieur amorce la déclinaison (*6.Descente*) de la QoS.

Répéter les actions de 2 à 6 pour les niveaux inférieurs. S'il n'y pas de modifications pour cette application, le Flag sera mis à zéro dans le vecteur d'agrégation.

7. Arrivé au dernier niveau, le processus d'agrégation peut commencer. La première action c'est l'évaluation (lecture) du vecteur d'agrégation pour le niveau considéré (*7.Evaluation*).

8. Puis il faut vérifier et analyser (DB Global) (*8.Réagir*) si des modifications ne sont pas intervenues entre temps, dans l'environnement considéré.

9. La fin de processus est l'activation/configuration (*9.Activation*).

10. A chaque niveau positionné (FLAG =1) les actions 7 à 9 sont à faire.

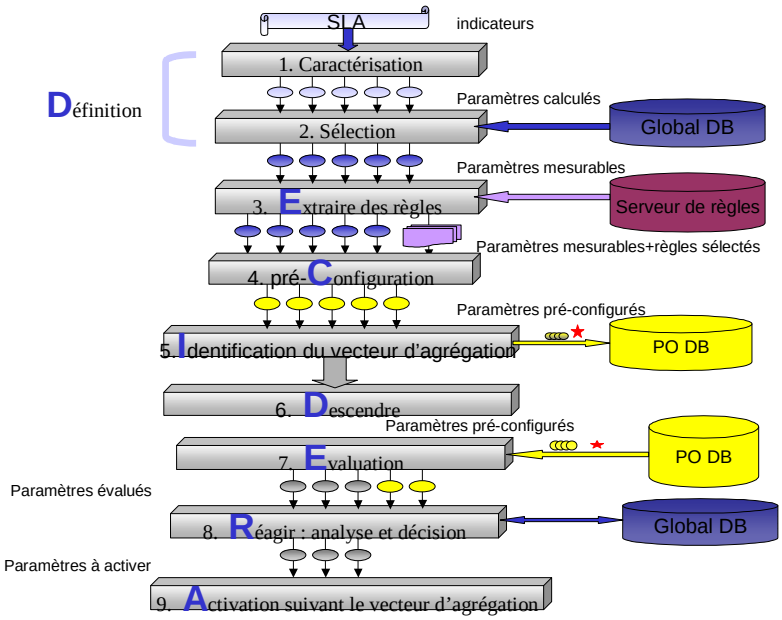


Figure II-18: Etapes génériques de vérification (contrôle) de la QoS

5.3.4 Scénarii de Déclinaison/Agrégation

Dans la section suivante, nous présenterons les scénarii de déclinaison et d'agrégation, qui traduisent les cas de figures où une reconfiguration est envisagée ou non (des changements architecturaux). Suivant le cycle de vie, nous pouvons définir deux cas : l'*initialisation*, où POrg est déclenché suite à une demande de déclinaison émanant de l'introduction d'un

nouveau service dans le réseau, et l'exploitation où POrg agit en cours d'exploitation du service et réagit aux changements dynamiquement. Nous considérons pour cela le cas où la déclinaison/agrégation concerne les agents L1 de niveaux service et réseau architecturalement superposés comme illustré dans la Figure II-19.

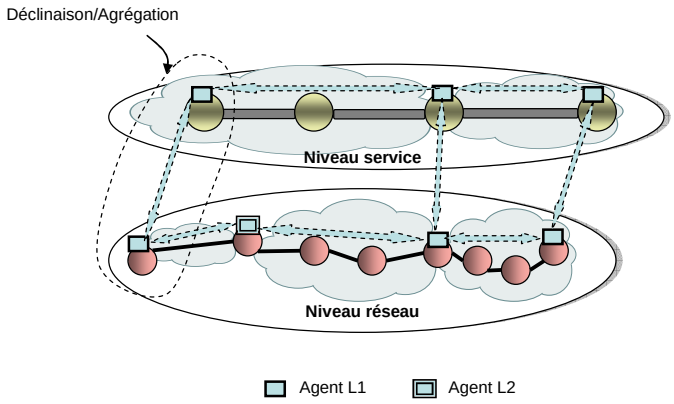


Figure II-19: Scénarii d'ingénierie d'architecture

Premier cas : A l'initialisation

Selon l'emplacement choisi pour les mécanismes de QoS, on parle de déclinaison ou d'agrégation de la QoS. En effet, la déclinaison se fait de haut en bas et l'agrégation dans l'autre sens et est à la charge du Use Case « Control QoS ». A l'initialisation, c'est-à-dire au moment de déploiement d'un nouveau service, les processus de déclinaison et agrégation permettent d'intégrer un service de manière transparente. Nous présenterons dans ce paragraphe ce cas de figure à travers des scénarii en ne mettant en exergue que les Use Cases et les acteurs qui jouent un rôle primordial :

Déclinaison

La Figure II-20 montre l'ensemble des actions prises par les agents POrg (distribué sur les couches architecturales) pour assurer l'intégration verticale.

- Une nouvelle demande (message) arrive à l'agent L1 de niveau service pour un nouveau service de la forme : « Service, Objet, Characteristic et QoS » (voir §p.90).
- Ce message est trié par la fonction QoS_Resources_Control de l'Use Case Control_QoS où la caractérisation de la demande se fait (étape « Caractérisation » du processus DECIDERA), les paramètres configurables et mesurables sont sélectionnés à partir de la base de données globale (action (2), (3) et (4) dans la Figure II-20 et

étape « Sélection » de DECIDERA) et les règles à appliquer sur les paramètres sélectionnés (seuils, contraintes de configuration, etc.) sont extraites (étape « Extraction » de DECIDERA) .

- Comme il s'agit d'une demande de déploiement d'un nouveau service, une reconfiguration des éléments de service peut être nécessaire des éléments service, pour cela, le flag est armé à 1 et enregistré dans la base de données locale (action (6) et (7)).
- Finalement, la demande de déclinaison est envoyée à l'agent L1 réseau pour accomplir le processus (étape « Descente » de DECIDERA).

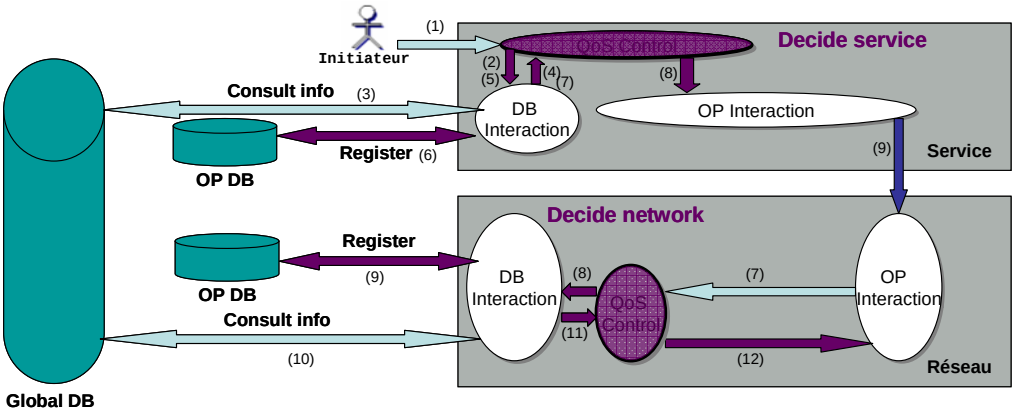


Figure II-20: Déclinaison activée à tous les niveaux

Agrégation

Une fois arrivé au plus bas niveau de l'architecture suivant le service à déployer, l'agent L1 de ce niveau initie le processus d'agrégation (Figure II-21). Il s'agit du processus de vérification de la déclinaison d'un côte et de mettre en œuvre les changements (reconfigurations) préconisés lors de la déclinaison.

- Après avoir consulter la base de données locale (étape « Evaluation ») pour voir les paramètres à configurer et la valeur du flag et consulter la base de données globale (étape « Réagir ») pour voir les changements qui sont arrivés depuis la fin du processus de déclinaison, l'agent effectue les configurations adéquates (étape « Activation ») vers les nœuds du réseau à travers la fonction React_To_Changes de l'Use Case QoS_Control.
- Le message d'agrégation est ensuite envoyé à l'agent service pour continuer ce processus.

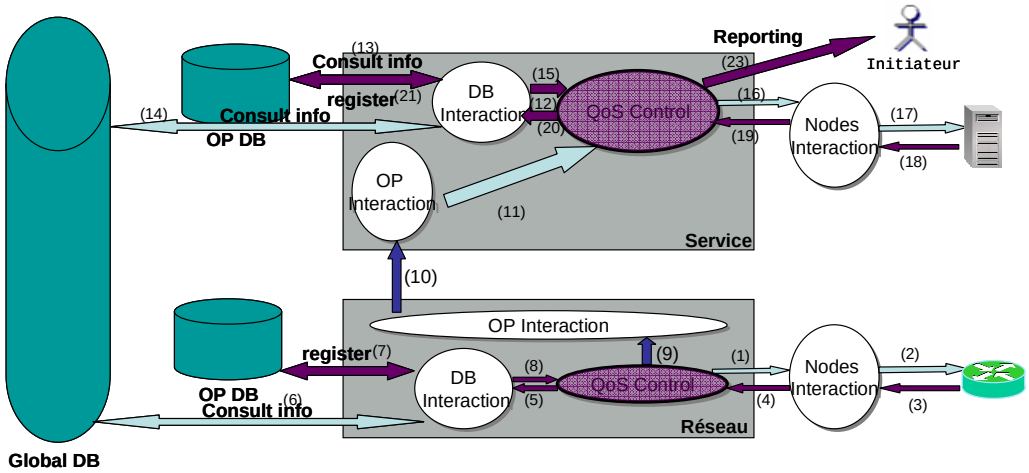


Figure II-21: Agrégation activée à tous les niveaux

En cours d'exploitation

Une fois que le service est déployé, les agents POrg se charge de maintenir la QoS de bout en bout pour le service. Lors d'un changement de l'état du réseau c'est l'Use Case « Auto-adaptation » qui est sollicité.

- Les changements sont détectés par les agents à travers la fonction Nodes_Interact à travers des notifications. Ces notifications sont transmises au module Diagnostic_of_Changes pour une analyse. Ce dernier organise les actions à effectuer et aiguille le message vers React_To_Changes de Auto_Adaptation pour prendre des décisions.
- Si des changements sont à effectuer, l'Use Case QoS_Control est invoqué à travers la fonction QoS_Resource_Control de QoS_Control. QoS_Control consulte la base de données locale pour extraire les informations relatives au traitement de la situation (seuils, etc.). La reconfiguration est ensuite effectuée à travers le module Nodes_Interact.

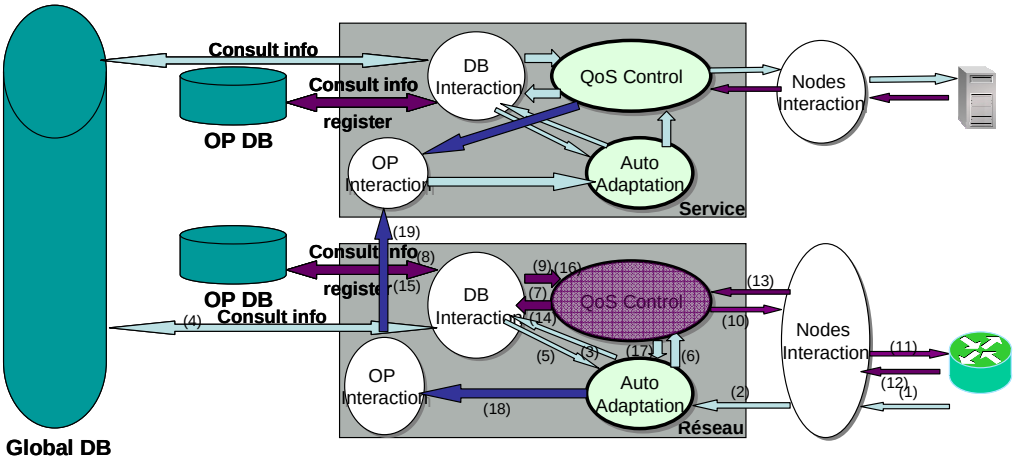


Figure II-22: Auto Adaptation en cours d'exploitation

5.4 Modèle architectural

La dimension architecturale des agents POrg de niveau *n* fait référence à la définition de la structure des modules composant l'agent et la coopération de ses modules (Figure II-23), sachant que les agents L1 et L2 sont des instanciations du même modèle.

Ce schéma met en exergue les éléments d'assemblage qui est le rôle de la dimension architecturale. Les modules cœurs Contrôle QoS, Auto-Adaptation et Reconfiguration sont ceux décrits avec leurs interactions dans le modèle fonctionnel que nous venons de décrire (§5.3)

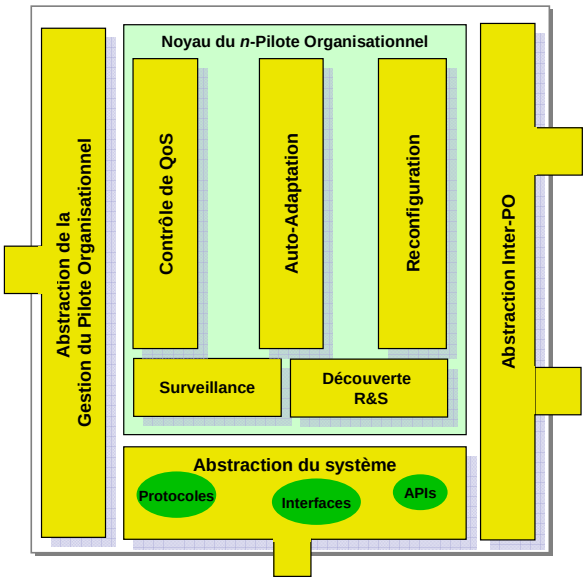


Figure II-23: Modèle architectural d'un agent POrg de niveau *n*

Par soucis de cohérence et de généricité, nous avons divisé ces modules en deux grandes parties : La première partie est celle qui permet à l'agent de communiquer avec son environnement (§5.4.1). La deuxième partie est le *Noyau* qui englobe tous les traitements internes et propres à l'agent (§5.4.2). Cette séparation entre le traitement et la communication facilite la jonction entre un agent *POrg* et n'importe quelle architecture, ainsi que l'évolutivité et l'adaptabilité de la plate-forme *POrg*. Nous décrivons dans les paragraphes suivants les différents modules architecturaux de *POrg*.

5.4.1 Modules d'interaction

Ceux sont des modules d'interfaçage qui sont principalement les fenêtres à travers lesquelles les agents POrg interagissent avec l'environnement, nous avons défini trois (3) interfaces: *ASYS* (*Abstraction du système*) qui englobe les interactions entre les agents POrg et l'architecture globale, *ABPO* (*Abstraction inter-agents*) et *GTPO* (*Gestion de l'agent*).

Abstraction du système (ASYS)

Ce module encapsule les méthodes permettant l'interaction d'un agent POrg avec l'architecture du réseau de niveau *N*. Cette interaction est traduite à travers les messages envoyés de/à l'agent (polling, reporting, configuration, réservation de ressources, etc.). Ces messages sont tout d'abord triés et différenciés (action 1). En effet, un agent POrg nécessite deux types de messages avec les nœuds du réseau :

- APIs : ces messages traduisent une demande, éventuelle, de l'agent de réservation de ressources ou toute interaction nécessitant l'implémentation d'une API (Agent.APISender) (Winsock2 QoS API par exemple)
- Messages de connexion : ces messages sont de types protocolaires comme les requêtes SNMP pour la surveillance, COPS, etc. (Agent.ProtoSender).

Ces messages sont, ensuite, encapsulés et adaptés (action 2) avant d'être envoyés à la méthode qui implémente l'API ou la machine protocolaire nécessaire.

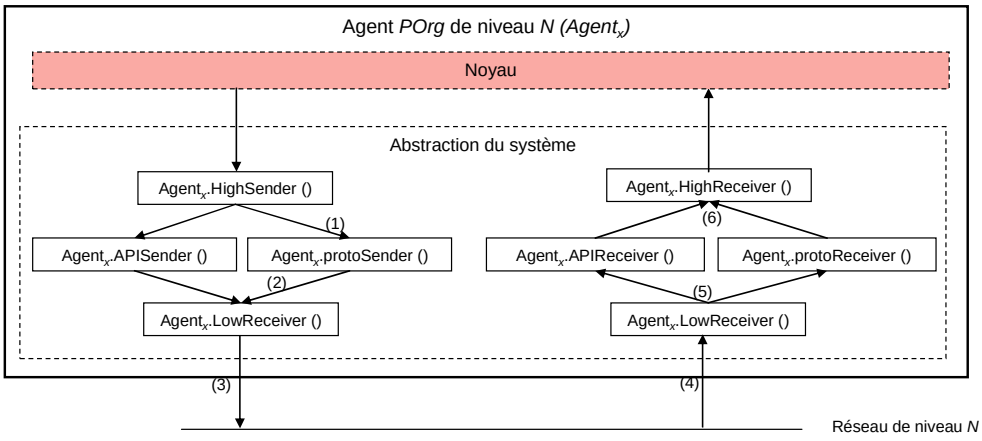


Figure II-24: Module d'abstraction du système ASYS

Les modules de réception vérifient en outre que la réponse de la part du module en amont correspond bien à la requête envoyée (actions 4, 5 et 6) en utilisant un mécanisme de numérotation.

Abstraction Inter-PO (ABPO)

Le deuxième type d'interaction est celui entre les agents des différents niveaux et ceux appartenant au même niveau. Ces interactions sont prises en charge par la couche *Abstraction Inter-PO (ABPO)*. En effet, les différents agents peuvent être implémentés en utilisant différentes technologies : système embarqué sur routeurs, systèmes multi-agents, agents mobiles, etc. Cette couche fait abstraction de la communication utilisée par ces technologies pour faire coopérer les différents « agents » de manière transparente. Pour cela, nous avons défini cette couche en deux couches : partie *frontale (front-end)* et partie *dorsale (backend)*.

La partie frontale est la partie où le type de la communication entre agents est formulé :

- A l'initialisation d'un agent, une connexion est établie avec un deuxième agent. Cette connexion est utilisée pour transporter les messages entre les agents.
- Pour chaque requête de déclinaison (flux informationnel ou fonctionnel) ou flux de données ou agrégat de flux, une session de surveillance est ouverte entre des agents afin de maintenir la QoS.

Un agent instancie autant d'ensembles (connexion, sessions) que d'agents voisins (L1 ou L2) avec qui il y a une interaction.

La partie dorsale est formée du module de communication d'ABPO. Les messages entre de/à la partie dorsale sont encapsulés et adaptés avant d'être envoyés aux modules implémentant les machines protocolaires nécessaires (protocoles utilisés entre les agents).

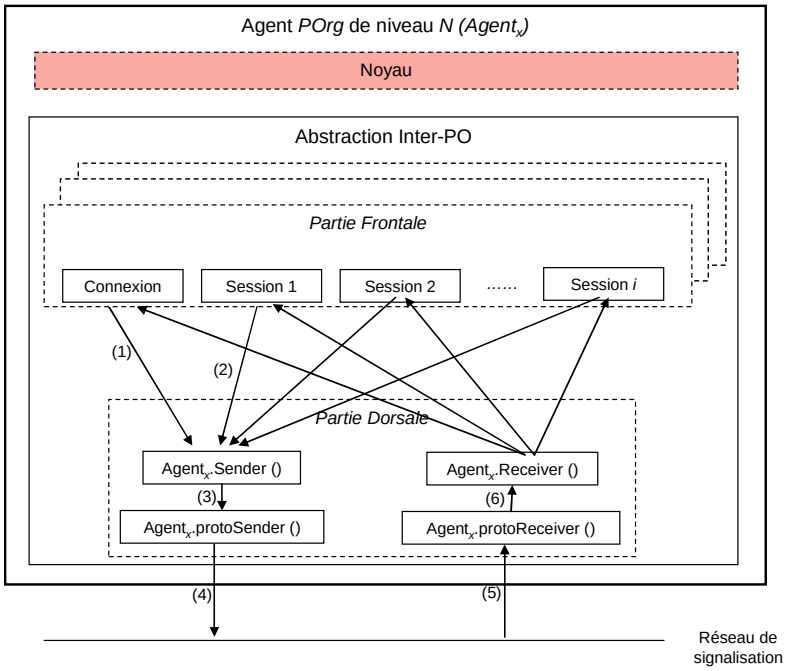


Figure II-25: Module d'abstraction Inter-PO

Gestion du POrg (GTPO)

Ce module fait abstraction des méthodes de communication utilisées pour administrer l'agent (COPS, LDAP, SNMP, etc.). Ce module est semblable au module d'abstraction du système (ASYS).

5.4.2 Noyau du Pilote Organisationnel

Les modules du noyau d'un agent POrg définissent le traitement générique qui se fait par l'agent indépendamment de la communication. En plus des modules de Contrôle_QoS, Auto_Adaptation et Reconfiguration, nous présentons dans ce paragraphe les modules de monitoring, de découverte des réseaux et service.

Surveillance

Afin de prendre en charge l'organisation de l'architecture et répondre au besoin d'évolutivité, l'agent POrg a le moyen d'analyser et de traduire le comportement du réseau en connaissances. Son utilisation permet par exemple d'établir des corrélations entre les valeurs extraites par les modules de surveillance et de définir les comportements-types du réseau.

Pour cela, le module de surveillance est spécifié afin de collecter les informations comportementales nécessaires.

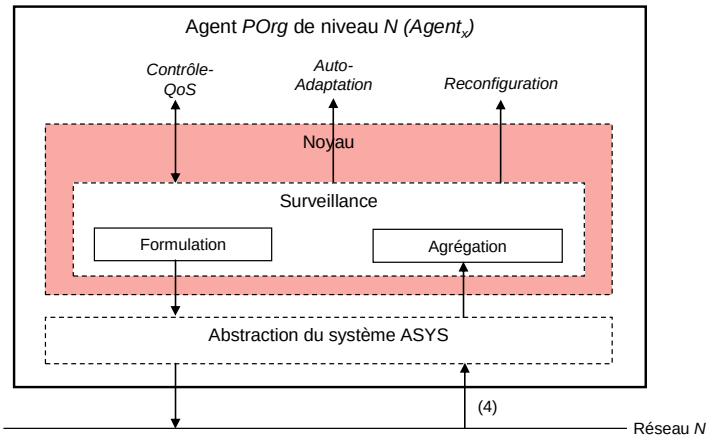


Figure II-26: Module de surveillance

La collecte des informations passe d'abord par une formulation de la demande de collecte d'informations. Cette formulation nécessite la définition de :

- *Critère/paramètre à surveiller* : Disponibilité, Fiabilité, Délai, Capacité ; nombre de frame/s, bande passante, nombre de paquet TCP, etc.
- *Seuil* : Suivant le module qui a initié la demande de surveillance, ce dernier peut demander d'être notifié si le paramètre qu'il a demandé à surveiller dépasse le seuil qu'il a défini.
- Objectif : lorsqu'un module demande une surveillance de paramètres de QoS il peut demander de surveiller une fonction de ce paramètre (*f(délai)* par exemple) où cette fonction traduit une moyenne, max, min, écart-type, etc.

Les réponses à ces demandes sont agrégées en vérifiant qu'elles correspondent bien aux requêtes envoyées.

Découverte R&S

Afin d'être le plus autonome et réactif possible, le module de découverte permet aux agents POrg d'identifier les composants du niveau *N* sans configuration manuelle. Cette découverte peut être divisée en trois parties : découverte de topologie, découverte de services et découverte de l'environnement.

- La découverte de topologie : il s'agit du module de découverte des changements de topologies du réseau. Afin d'éviter d'utiliser les outils classiques de découverte de

topologies comme Ping, Traceroute, etc. on peut opter pour l'utilisation de la lecture des tables de routages des nœuds sur lesquels les agents POrg sont embarqués.

- La découverte de services :

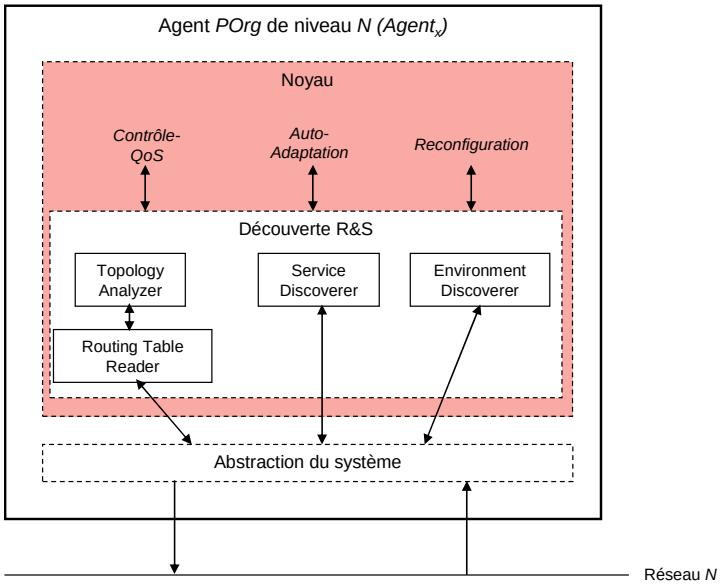


Figure II-27: Module de découverte réseau et services

Les autres modules architecturaux, à savoir, Control_QoS, Auto_Adaptation et Reconfiguration sont analogues au modèle fonctionnel de ces fonctions (§5.3.2).

5.5 Conclusion

POrg est une concrétisation de notre concept d'Ingénierie d'Architecture. Ils prennent en charge l'*intégration verticale* à travers la *cohabitation* entre les niveaux superposés autonomes mais intégrés (pour la QoS de bout en bout), et l'*intégration horizontale* à travers la *coopération* entre les nœuds et domaines juxtaposés autonomes pour la QoS horizontale.

Dans ce chapitre, nous avons spécifié POrg à travers trois dimensions :

- La dimension organisationnelle où nous avons vu que POrg est une composition d'agents ayant des rôles différents suivant leur emplacement dans le réseau. Les agents L1 attribués à un niveau donné (réseau de transport par exemple) sont des agents qui sont en interaction :
    - Avec les agents de niveaux supérieur et inférieur, pour assurer la cohabitation entre les niveaux, et dont s'occupant des processus de déclinaison et agrégation
    - Avec les agents de même niveau afin d'assurer la coopération entre les composants d'un même niveau afin de maintenir une QoS horizontale conforme, et donc d'occupant du processus de réorganisation des domaines au cas la QoS horizontale n'est pas respectée. La réorganisation des domaines est basée sur un algorithme que nous avons développé et présenté dénommé : *Path-based Clustering*. C'est un algorithme qui se base sur le principe de clustering et qui propose, contrairement aux techniques de clustering existantes, de « clusteriser » un réseau suivant la disponibilité des ressources et pas seulement la position géographique des nœuds dans le réseau.
- La dimension fonctionnelle où nous avons vu que les fonctions de POrg se résument en : *Contrôler* la QoS pour la prise en charge de la déclinaison des besoins QoS et le maintien de cette QoS, et *Auto-adapter* pour la réaction aux changements (structurels et conditionnels) dans l'état du réseau, et *Reconfigurer* qui se charge de toutes les interactions avec les nœuds du réseau à des fins de changement de paramètres/configuration. Nous avons pu à travers ces fonctions, présenter notre proposition processus de déclinaison et agrégation de la QoS (moteurs de l'intégration verticale) que nous avons dénommé : DECIDERA.

- La dimension architecturale où les modules architecturaux composant chaque agent POrg ont été décrits. Chaque agent est composé d'un noyau de traitement propre et des bloques d'interaction avec l'environnement. Cette décomposition est motivée par le besoin de jonction de *POrg* à n'importe quelle architecture, ainsi que l'évolutivité et l'adaptabilité de la plate-forme *POrg*

Dans le cadre du projet IA/PO, nous avons débuté la réalisation d'un prototype POrg. Pour cela, nous avons monté plusieurs plateformes expérimentales. La troisième et dernière partie de ce rapport sera consacrée à la présentation de ces différentes plates-formes expérimentales ainsi que l'implémentation de POrg.

Partie III SIMULATIONS ET EXPERIMENTATIONS AU SERVICE DE L'INGENIERIE D'ARCHITECTURE

Introduction

Cette troisième, et dernière, partie du rapport est consacrée au développement du Pilote Organisationnelle ainsi qu'à la présentation de la plate-forme qui nous a servi de support expérimental pour la définition des règles d'ingénierie utiles pour *POrg*.

Cette partie est divisée en deux (2) chapitres : le Chapitre 6 présente les plates-formes misent en place à travers des collaborations avec des industriels pour extraire les règles d'ingénierie utilisées par POrg. Le Chapitre 7 présente les détails d'implantation de *POrg* et les choix que nous avons effectué pour cette implémentation.

Chapitre 6

Extraction des règles d'ingénierie

Ce chapitre sera consacré à la présentation des plates-formes expérimentales qui nous servent de source des règles d'ingénierie utilisées par *POrg* afin s'assurer ses fonctions. Ces plates-formes ont été montées dans le cadre de projets en collaboration avec des industriels (SFR R&D dans le cardre du projet IA/PO, et Alcatel R&I dans le cadre du projet ISR – Intelligent Switch Router –) que nous allons citer par la suite.

Comme nous l'avons laissé entendre dans les chapitres précédents, *POrg* s'assure (i) de la cohérence des choix de superpositions des blocs architecturaux d'un côté, et (*intégration verticale*) (ii) et du respect du contrat de QoS confié au niveau considéré (*intégration horizontale*). Ces deux modèles constituent les composantes de base de l'ingénierie d'architecture.

Pour cela, nous présentons dans ce chapitre deux plates-formes les plus importantes auxquelles j'ai particulièrement participé. Elles peuvent paraître hétéroclites, mais en fait, elles sont conçues et elles évoluent en fonction des contrats qui nous lient à nos partenaires. Le but étant d'avoir le support adéquat pour avoir les réponses aux questions urgentes qui se sont posées à nous.

La première plate-forme est la plate-forme expérimentale pour l'étude des contrats de QoS dans un contexte réel DiffServ (RIM de Cegetel) (§6.1). Le comportement du réseau (résultats des expérimentations) ne pourra être communiqué dans ce rapport pour des raisons de confidentialité.

La deuxième plate-forme porte sur l'étude de l'identification des acteurs de la chaîne de bout en bout impactant la QoS (§6.2). Pour ce faire, on étudie le comportement des applications

face aux changements de chacun de ces acteurs (impact des choix architecturaux sur le QoS perçue de bout en bout).

La troisième plate-forme est la plate-forme BGP (§6.3) qui a pour vocation d'étudier le comportement (problèmes et anomalies liées à la QoS horizontale) du protocole de routage BGP afin d'identifier les symptômes et indices permettant de localiser les problèmes dans le réseau. Ces symptômes et indices vont être utilisés par POrg pour son fonctionnement (réorganisation des AS).

6.1 La plate-forme QoS (intégration verticale) : Les contrats (contexte DiffServ)

Nous présenterons, dans ce chapitre, la plate-forme expérimentale que nous avons montée en collaboration avec SFR R&D. Il s'agit d'une plate-forme expérimentale pour l'étude des contrats de QoS dans un contexte réel DiffServ (Réseau IP Multiservice -RIM- de Cegetel). Le but étant d'arriver à un paramétrage optimal de DiffServ suivant les flux applicatifs traversant le réseau. Ces informations de paramétrage seront traduites en règles d'ingénierie utilisées par les agents POrg afin d'assurer les processus de déclinaison et d'agrégation (intégration verticale).

Nous présentons dans §6.1.1 les outils utilisés, et dans §6.1.2 les détails de la plate-forme matérielle. Dans la section §6.1.3, nous décrivons quelques scénarii de tests du comportement du réseau par rapport aux différents flux traversés.

6.1.1 Les outils utilisés

Générateurs de trafic

- **Générateur de LAAS**

Nous utilisons comme générateur de trafic un outil fourni par le LAAS-CNRS de Toulouse. C'est un générateur qui fonctionne sur une plate-forme LINUX. Il permet de générer des flux UDP à distribution constante ou exponentielle, des flux TCP, des flux audio et vidéo.

Le logiciel doit être installé sur chaque PC générateur / absorbeur. Avant de lancer un trafic sur le réseau par le générateur il faut lancer l'outil installé sur l'absorbeur afin qu'il attende un trafic sur un port UDP (ou TCP) spécifié par l'utilisateur. Ensuite, on peut lancer le trafic à partir du PC générateur à l'aide d'une commande qui contient en paramètre les propriétés du flux à envoyer.

Ce générateur offre entre autres la possibilité de paramétrer le débit des paquets générés, les adresses source et destination du trafic à envoyer, le numéro de port à utiliser, le type du flux (voix, vidéo, images, etc.) et sa distribution. (Constante, exponentielle).

- **AX4000**

L'AX4000 est un produit matériel de la compagnie « Spirent ». Il est utilisé pour la génération de trafic IP. Le trafic est généré et absorbé par le même équipement (interface de réception). L'AX4000 intègre plusieurs fonctionnalités pour la génération de trafic non

seulement d'un LAN à un autre mais aussi sur des interfaces de divers types (Frame-Relay, ATM, etc.).

Analyseurs de trafic

- **IpEngine de Ipanema**

Nous utilisons pour mesurer le trafic de bout en bout dans le RIM l'outil QoSmart qui est un outil de mesure de bout en bout fourni par la société Ipanema Technologies©. Il comprend deux sondes appelées ipEngine qui se branche sur les deux LANs, une antenne GPS pour chaque sonde et une console d'administration. Chaque sonde fait une capture de tous les paquets qui circulent sur le LAN (voir détails en annexe). Les antennes GPS servent à synchroniser les sondes avec le temps GPS dans le but de calculer le délai de transit des paquets IP. La console d'administration contient un outil qui collecte et traite les données des sondes ainsi qu'une version 'runtime' de l'outil InfoVista pour afficher des rapports sur les mesures effectuées. A un intervalle fixé par l'utilisateur, les sondes envoient à la console d'administration (IpBoss) les corrélation-records relevées sur les LANs. IpBoss les traite et les stocke dans une MIB propriétaire appelée « IPANEMA-TECHNOLOGIES.mib ». A partir de cette MIB l'outil *InfoVista* collecte les données spécifiques aux différentes statistiques calculées par la console d'administration et les affiche sous forme de rapports. Ces rapports sont exportables sous format HTML, sous format image (GIF) ou bien sous format texte, offrant ainsi la possibilité de ré-exploiter ces données.

Les quatre paramètres mesurés pour chaque flux sont :

  - Le débit (Throughputs)
  - Le délai (Latency)
  - La gigue (Jitter)
  - Les pertes (Packet Loss)

- **Mib Browser**

Il s'agit d'analyseur utilisant les MIB des équipements (routeurs) pour extraire les indicateurs pertinents de qualité de service, la charge CPU des routeurs et les informations sur l'état des différents liens et interfaces comme leur taux de charge, leur disponibilité, etc. Pour plus de précision sur l'analyse de la QoS, nous utilisons un outil de gestion des MIBs qui est "Mib

Browser Professional", c'est un gestionnaire de MIBs de la compagnie MG-Soft©. Il supporte les versions v1, v2 et v3 du protocole SNMP.

6.1.2 La plate-forme matérielle

La plate-forme QoS est constituée de deux (2) LANs reliés par un backbone IP/MPLS comme indiqué dans la Figure III-1.

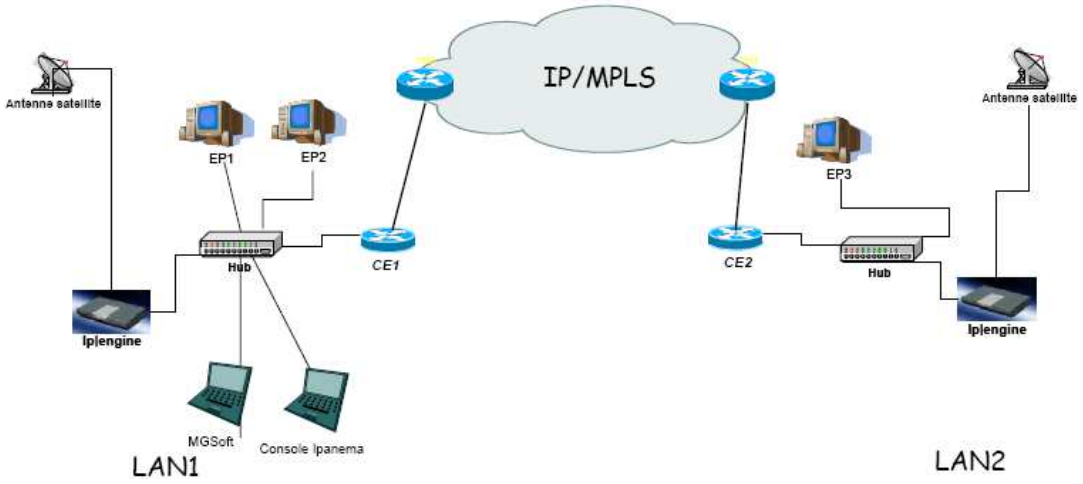


Figure III-1: Plate-forme de test de QoS

Les deux routeurs CEx sont connectés soit aux deux LANs (lorsqu'on utilise le générateur de LAAS) soit à deux ports du SmartBits (lorsqu'on utilise le SmartBits).

Sur le LAN1 on trouve:

- Un routeur (noté CE1)
- La console d'administration Ipanema (administration des sondes Ipanema, IpBoss)
- Une sonde IpEngine1 et une antenne GPS
- Un Hub 10/100 à 24 ports reliant tous les équipements du LAN1
- La machine Mib browser avec les logiciels MGSof
- Deux endpoints pour le générateur du LAAS est installé dans EP1 et EP2.

Sur le LAN2 on trouve :

- Un routeur (noté CE2)
- Une sonde IpEngine2 et une antenne GPS

- Un Hub 10/100 à 24 ports reliant tous les équipements du LAN2
- Une machine avec le système d'exploitation LINUX qui tourne dessus servant à l'absorption de trafic généré par le générateur du LAAS (EP3).

6.1.3 Extraction des règles d'ingénierie

Configuration des classes de services

Nous avons utilisé trois classes de service principales : C1, C2 et C3, les détails de configuration de ces classes ne peuvent être communiqués pour des raisons de confidentialité :

		Débit	DSCP	Queuing	Scheduling	Shaping
Classe C1		D1	XX	Tail Drop	PQ	TS
Classe C2	**C2**	D2	XX	WRED	CBWFQ	TS
	C2'	D3	XX	WRED	CBWFQ	TS
Classe C3		D4	XX	WRED	CBWFQ	TS

- *Trafic temps réel (C1*) : Des flux de types : G711, G726 et G729 sont générés par le générateur de LAAS. Les tailles des paquets sont respectivement : 150 octets, 118 octets et 70 octets. Les ports UDP sont compris entre 16384 et 32767.
- *Trafic Données critiques (C2)* : Des flux UDP de nature constante (CBR) sont générés. Les tailles des paquets seront de 1024 octets. Les débits sont variables (en augmentation de 500kbps jusqu'à 2,2 Mbps). Les ports UDP sont compris entre 400 et 499.
- *Trafic Best effort (C3)* : Des flux UDP de nature constante (CBR) sont générés. Les tailles des paquets seront 1024 octets. Les débits sont variables (en augmentation de kbps jusqu'à 1,1Mbps). Les ports UDP seront compris entre 300 et 399.

Déroulement des tests

La génération de trafic s'effectue avec le générateur du LAAS. Le générateur est placé sur le LAN1 et l'absorbeur de trafic sur le LAN2.

Les sondes Ipanema en notre possession ne supportent pas un débit supérieur à 2Mbps. Par conséquent, le déroulement de la mesure sera différent suivant le débit des flux envoyés : débit inférieur à la bande passante maximale de la ligne d'accès (mode non congestionné) et débit supérieur à la BP de la ligne d'accès (mode congestionné). Nous avons déroulé les tests suivant trois (3) phases: signature des flux, mode non congestionné et mode congestionné.

- *Signature des flux*

Les tests de signatures des flux consistent à effectuer des mesures individuellement dans chaque classe pour obtenir, grâce aux différents outils, des indicateurs étalons des valeurs de débit, délai, gigue et pertes.

Les sondes Ipanema sont placées respectivement sur chacun des deux LANs distants (LAN 1 et LAN 2). Le trafic est mesuré entre la sonde IPe1 (source) placée sur le LAN1 et la sonde IPe2 (destination) placée sur le LAN2. La console IpBoss permettant la visualisation des différents paramètres de configuration et de synchronisation est placée sur le LAN1.

Les tests effectués vont être réalisés à partir de flux issus des trois classes définies dans le pack multimédia individuellement (C1 toute seule, C2 toute seule et C3 toute seule).

- *Mode non congestionné*

Les sondes Ipanema sont placées respectivement sur chacun des deux LANs (LAN 1 et LAN 2). Le trafic est mesuré entre la sonde IPe1 (source) placée sur le LAN1 et la sonde IPe2 (destination) placée sur le LAN2. La console IpBoss permettant la visualisation des différents paramètres de configuration et de synchronisation est placée sur le LAN1.

Les tests effectués vont être réalisés à partir de flux issus des trois classes définies dans le pack multimédia de CEGETEL :

Flux unitaires de classe C1, de classe C2 ou de classe C3,

Flux agrégés de classes C1+C2, C2+C3 et C1+C2+C3.

- *Mode congestionné*

Le gestionnaire de MIB (Mib Browser) est lancé sur la machine Mib Browser sur le LAN 1. L'agent SNMP est activé sur le routeur d'accès CE 1. Mib browser permet de « pooler » les

MIBs du routeur CE1 et d'extraire les indicateurs de QoS, charge CPU et consommation de la mémoire.

On extrait aussi des indicateurs de QoS en accèdent aux routeurs CE1 et PE egress par CLI et en introduisant la commande : « *show policy-map interface num-int-serie, show frame-relay pvc 'num-pvc'* ».

Les flux générés sont majoritairement des flux UDP à distribution constante ; des tests ont également été réalisés avec des flux exponentiels, audio et vidéo issus du générateur du LAAS.

Paramètres à mesurer

Les paramètres DFDC de QoS à mesurer dans le cadre des tests de QoS sont les suivants :

- **Pour les mesures de bout en bout**
    - le débit (throughput)
    - le délai (latency)
    - la gigue (jitter)
    - le taux de perte de paquets (packet loss).
- **Pour les mesures en local (local aux routeurs)**
    - Les mesures utilisant SNMP

Pour ces mesures, nous interrogeons trois types de MIBs : MIB de QoS, MIBd surveillance du CPU et MIB de surveillance de la mémoire :

- **MIB de QoS**

  Ce sera « CISCO-CLASS-BASED-QOS-MIB », on a besoin des valeurs des objets suivants :

    - cbQosCMPrePolicyPkt : Nombre de paquets reçus en entrée avant l'application de n'importe quel mécanisme de QoS.
    - cbQosCMDropPkt : Nombre de paquets éliminés après l'application des mécanismes de QoS.
    - cbQosCMDropByte : Nombre d'octets éliminés après l'application des mécanismes de QoS.

Le rapport (cbQosCMDropPkt*100)/ CbQosCMPrePolicyPkt donne le taux de perte dans le routeur d'accès CE1 après l'application des mécanismes de QoS sur les paquets entrants. On pourra ainsi comparer les pertes observées par les sondes Ipanema à ceux extraient par MIB Browser.

- CbQosCMNoBufDropPkt : Nombre de paquets éliminés à cause d'un manque de buffers libres (mémoire libre)

- **MIB de surveillance du CPU**

Ce sera « OLD-CISCO-CPU-MIB », on a besoin des valeurs des objets suivants :

- BusyPer Pourcentage d'utilisation du CPU pendant les 5 dernières secondes.
- AvgBusy1 : Pourcentage moyen d'utilisation du CPU pendant la dernière minute.
- AvgBusy5 : Pourcentage moyen d'utilisation du CPU pendant les 5 dernières minutes.

- **MIB de surveillance de la mémoire**

Ce sera « CISCO-MEMORY-POOL-MIB», on a besoin des valeurs des objets suivants:

- FreeMem : quantité de mémoire libre (en octets)
- BufferSmTotal et BufferSmFree : le nombre total et libre de buffer de taille « petite »
- BufferMdTotal et BufferMdFree : le nombre total et libre de buffer de taille « moyenne »
- BufferBgTotal et BufferBgFree: le nombre total et libre de buffer de taille « grande »
- BufferLgTotal et BufferLgFree: le nombre total et libre de buffer de taille « large »
- BufferHgTotal et BufferHgFree: le nombre total et libre de buffer de taille « très grande »

- Les mesures en accès CLI :

On devra récupérer nombre de paquets perdus pour chaque classe de service à l'aide des commandes « ***show policy-map interface*** *numero-int-serie,* ***show*** *frame-relay* ***pvc*** *num-pvc* »

Nous présentons dans ce qui suit la deuxième plate-forme de QoS.

6.2 La plate-forme QoS (intégration verticale): Les choix

Nous présenterons, dans ce chapitre, la plate-forme expérimentale que nous avons montée à l'ENST. Cette plate-forme a pour vocation d'étudier le comportement des applications face aux changements qui se produisent dans le réseau.

L'étude du comportement des applications réelles nous permettra de dégager les règles, dites, d'ingénierie qui seront utilisées par la POrg comme un input pour l'aide à la décision.

Nous présentons dans §6.2.1 les outils utilisés, et nous présentons dans §6.2.2 les détails de la plate-forme matérielle. La section §6.2.3 sera consacrée à l'étude du comportement de l'application *Streaming* à travers des tests de QoS de bout en bout. Nous verrons à travers ces tests l'impact des choix architecturaux (intégration verticale) sur la QoS perçue de bout en bout.

6.2.1 Les outils utilisés

Pour l'étude du comportement des applications face aux changements du réseau, nous avons utilisé des générateurs de trafic, des analyseurs de trafic et des gestionnaires actifs du lien d'accès, communément appelés *Groomers*.

Applications réelles

Il s'agit des applications vidéo *Netmeeting, Windows Media Player* et *Windows Media Server.* Pour ce dernier, étant donné que nous avions windows 2000 Server comme poste serveur, nous avons la possibilité d'utiliser le serveur Media de Microsoft. Avec Windows Media Encoder, on peut enregistrer un fichier vidéo sous le format wmv. Le serveur de streaming est lancé avec la vidéo à diffuser et attend les connexions du client. Ainsi ce dernier utilisant Windows Media Player peut demander sa vidéo à travers le port 8080 sous le protocole RTSP. Le client peut également utiliser le protocole de streaming de Microsoft MMS (Multi Media Server) qui fonctionne soit sur TCP soit sur UDP.

Analyseurs de trafic

Nous avons utilisé Ehereal pour l'analyse du trafic.

Groomer

Nous utilisons dans notre plate-forme un outil de gestion active du lien d'accès Le M100 et le M200 sont des équipements actifs, dédiés à la gestion totale des bandes passantes sur les liaisons d'accès. Ils prennent en charge la régulation des flux définie à partir du centre de gestion, le SGM version 2.1. Ce dernier communique avec l'ensemble des équipements actifs dans le réseau (M*x*00) de façon à permettre leur configuration (règles, adresses,…), à relever les informations statistiques et à activer les nouveaux services.

6.2.2 La plate-forme matérielle

La plate-forme QoS est constituée de deux (2) LANs reliés par un backbone IP/MPLS comme indiqué dans la Figure III-2.

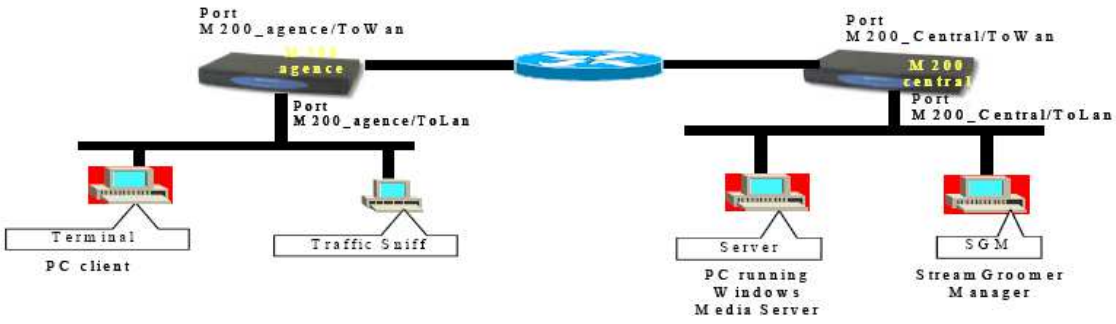


Figure III-2: La plate-forme QoS

Sur le LAN1 on trouve:

- Streamroomer M200
- Un Hub 10/100 à 24 ports reliant tous les équipements du LAN1
- Terminal client de streaming : Windows Media Player

Sur le LAN2 on trouve :

- streamGroomer M200
- Un Hub 10/100 à 24 ports reliant tous les équipements du LAN2
- Serveur de streaming : Windows Media Server
- Gestionnaire des streamgroomers (StreamGroomer Manager, SGM).

6.2.3 Extraction des règles d'ingénierie

Dans cette section, nous allons présenter les expérimentations que nous faisons sur la plate-forme QoS. Le but est d'étudier l'impact des choix architecturaux sur le QoS perçue de bout en bout et d'extraire les règles d'ingénierie utilisées par *POrg*.

Scénario n°1 : Impact du choix du protocole de transport

- *Déroulement du scénario*

Le client WMP[6] se connecte au serveur WMS[7] pour une diffusion vidéo à la demande utilisant RTSP. La vidéo est codée en WMV à *1131 kbps* pour une durée de 300 secondes. Les streamgoomers sont utilisés pour limiter la bande passante disponible dans le réseau (simuler les différents cas qui peuvent se présenter dans la réalité).

RTSP utilise soit TCP ou UDP comme protocole de transport. Dans ce scénario nous étudions l'impact du choix de TCP ou UDP sur la qualité de diffusion de ma vidéo chez le client. Nous augmentons la bande passante disponible dans le réseau de *1 Mbps* à *2 Mbps* et nous observons les pertes des paquets et la qualité de diffusion de la vidéo exprimée en nombre d'images ignorées (Figure III-3).

- *Résultats*

La Figure III-3(b) montre qu'en présence de congestion (1 Mbps), UDP présente une perte de 10%. TCP est plus robuste aux pertes (0%) mais cause une ignorance d'images (Figure III-3(a)).

[6] Windows Media Player

[7] Windows Media Server

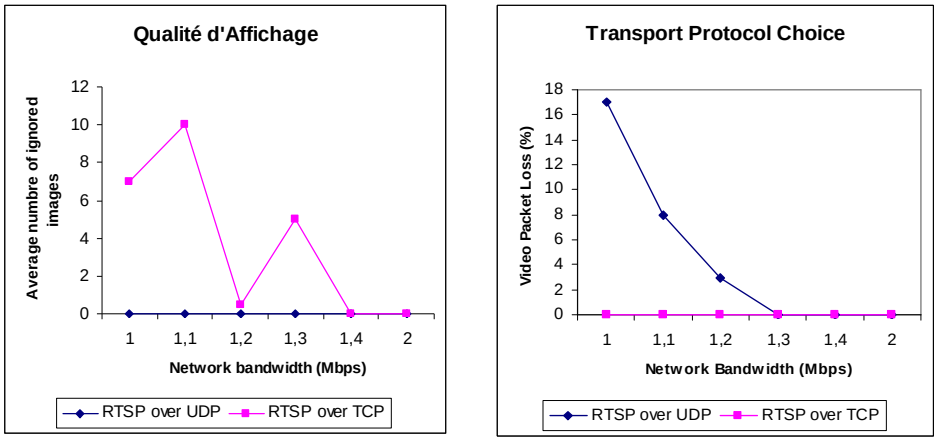


Figure III-3: Impact du choix du transport

Globalement, pour un seul flux de streaming, l'utilisation de TCP ou UDP est indifférente même en congestion.

Scénario n°2 : Impact de la capacité des équipements

- *Déroulement du scénario*

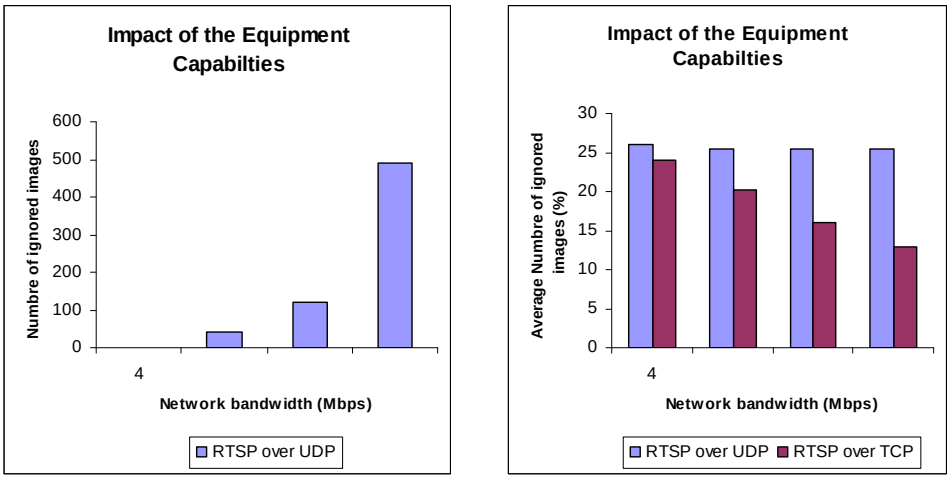
Quatre clients WMP, l'un après l'autre sur la même machine, se connectent au serveur WMS pour une diffusion vidéo à la demande utilisant RTSP. La vidéo est codée en WMV à *1131 kbps* pour une durée de 300 secondes. La bande passante est configurée à 10Mbps, largement suffisante pour les quatre flux.

- *Résultats*

La Figure III-4 montre la qualité de diffusion de la vidéo exprimée en nombre moyen d'images ignores et nombre moyen d'images diffusées (frame per second, fps) utilisant TCP et UDP.

Le taux de perte des deux protocoles est de 0%. Dans la Figure III-4(a), nous remarquons que l'utilisation de TCP cause un nombre important d'images ignorées (500) par rapport à UDP (0 image).

La Figure III-4(b) montre l'utilisation de TCP dans un environnement chargé cause une dégradation de la QoS perçue (moins de 15 fps) par rapport à UDP (28 fps). Une qualité excellente est obtenue avec 28 fps.



(a) (b)

Figure III-4: Impact des équipements

Conclusion

Ces tests ont montré qu'en présence de charge sur l'équipement (calcule du % d'utilisation du CPU, mémoire, etc.), pour l'application streaming basculer à UDP au lieu de TCP comme protocole de transport.

Ces tests doivent être étoffés (expérimentations en cours). On peut étudier la taille TPDU à utiliser pour chaque application, et ainsi personnaliser la taille suivant les applications, contrairement à ce qui se fait aujourd'hui où la taille TDPU est unique pour toutes les applications (taille TPDU configurée sur le système d'exploitation car la pile TCP/IP est implémentée sur l'OS) ;

Après avoir présenté la plate-forme QoS, nous présentons dans ce qui suit la troisième plate-forme qui nous a service de source d'étude des règles d'ingénierie.

6.3 La plate-forme BGP : Intégration horizontale

Nous présenterons, dans cette section, la plate-forme expérimentale que nous avons montée en collaboration avec Alcatel R&I dans le cadre du projet : « BGP Routing Monitoring » qui vise à extraire les informations pertinentes permettant d'identifier les problèmes dans le réseau par les éléments du réseau eux-mêmes. Ces informations seront, par la suite, utilisées par les agents POrg (modules embarqués sur les routeurs) afin d'assurer une auto-configuration (réorganisation) des routeurs suivant l'état du réseau.

Ce projet fait partie du projet Alcatel « Intelligent Switch Router (ISR) [92]» ayant pour objectif d'embarquer la plus grande partie de d'intelligence du réseau dans les couches les plus basses des éléments du réseau. Cette plate-forme a pour vocation d'étudier le comportement des applications face aux changements qui se produisent sur le réseau.

L'étude du comportement des applications réelles nous permettra de dégager les règles, dites, d'ingénierie qui seront utilisées par la POrg comme un input pour l'aide à la décision.

Nous présentons dans §6.3.1 les outils utilisés, et nous présentons dans §6.3.2 les détails de la plate-forme matérielle. La section §6.3.3 sera consacrée à l'étude du comportement de CRTP (implémentation BGP de Alcatel).

6.3.1 Les outils utilisés

Générateurs de trafic : AX4000

L'AX4000 est un produit matériel de la compagnie « Spirent ». Il est utilisé pour la génération de trafic IP. Le trafic est généré et absorbé par le même équipement (interface de réception). L'AX4000 intègre plusieurs fonctionnalités pour la génération de trafic non seulement d'un LAN à un autre mais aussi sur des interfaces de divers types (Frame-Relay, ATM, etc.).

Analyseurs de trafic : Ethereal

BGP Alcatel (CRTP)

Nous étudions le comportement de BGP à travers l'implémentation BGP de Alcatel (CRTP).

6.3.2 La plate-forme expérimentale

La plate-forme BGP est constituée de deux (2) stations Linux RedHat connecté à l'AX4000 à travers des interfaces FastEthernet. Chaque station peut héberger plusieurs nœuds BGP. Les nœuds BGP activés dans les deux stations forment le réseau BGP à tester. L'AX4000 est administré à travers une troisième station.

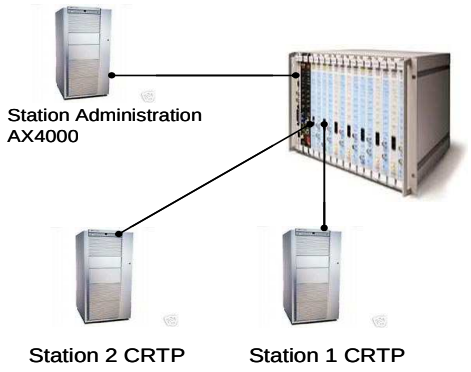


Figure III-5: La plate-forme de test BGP

6.3.3 Extraction des règles d'ingénierie

Dans cette section, nous allons présenter les expérimentations que nous avons faits sur la plate-forme BGP afin d'extraire les informations pertinentes permettant d'identifier les problèmes dans le réseau par les éléments du réseau eux-mêmes. Ces informations seront, par la suite, utilisées par les agents POrg (modules embarqués sur les routeurs) afin d'assure une auto-configuration (réorganisation) des routeurs suivant l'état du réseau. Nous avons divisé les expérimentations pour identifier les problèmes du routage BGP en deux parties : tests de conformité qui identifient les problèmes liés à la machine à état BGP, et test de fonctionnalité qui identifie les problèmes de convergence, de stabilité et de scalabilité et leurs symptômes.

Pour des raisons de confidentialité, les résultats relatifs au comportement de CRTP ne peuvent être publiés.

Tests de conformité

Il s'agit d'étudier la machine à état BGP (CRTP) afin d'identifier les points qui peuvent impacter le comportement BGP dans le réseau. Il s'agit d'étudier un seul nœud BGP Alcatel comme illustré dans la Figure III-6

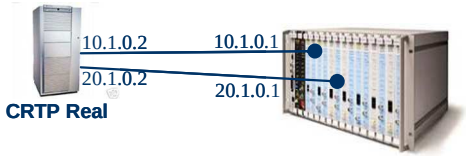


Figure III-6: Tests de conformité BGP

.

Pour cela, nous proposons d'analyser la procédure d'initialisation d'une session BGP (Figure III-7).

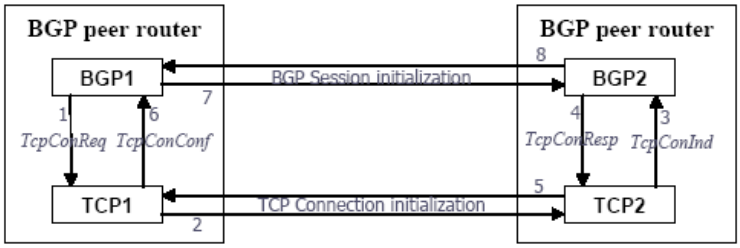


Figure III-7: Ouverture d'une session BGP

Le service BGP (BGP1) commence par demander au protocole de transport sous-jacent (TCP1) d'ouvrir une connexion au peer distant (BGP2) en utilisant TcpConReq. Le peer (BGP2) reçoit un TcpConInd du protocole TCP sous-jacent (TCP2). Il répond alors par un TcpConResp. BGP1 reçoit alors un TcpConConf indiquant le succès de l'ouverture de la connexion.

Ceci montre que nous avons deux automates : automate BGP et automate TCP. De ce fait, nous avons identifié deux cas, suivant qui commence la session BGP : *Initiateur* et *Répondeur*. Ces deux états n'apparaissent pas dans la spécification de BGP4 (RFC 1771 [46]).

Nous avons proposé suite à cela, une réorganisation de la machine à état BGP afin de séparer les deux rôles et proposé les messages à échanger (Figure III-8).

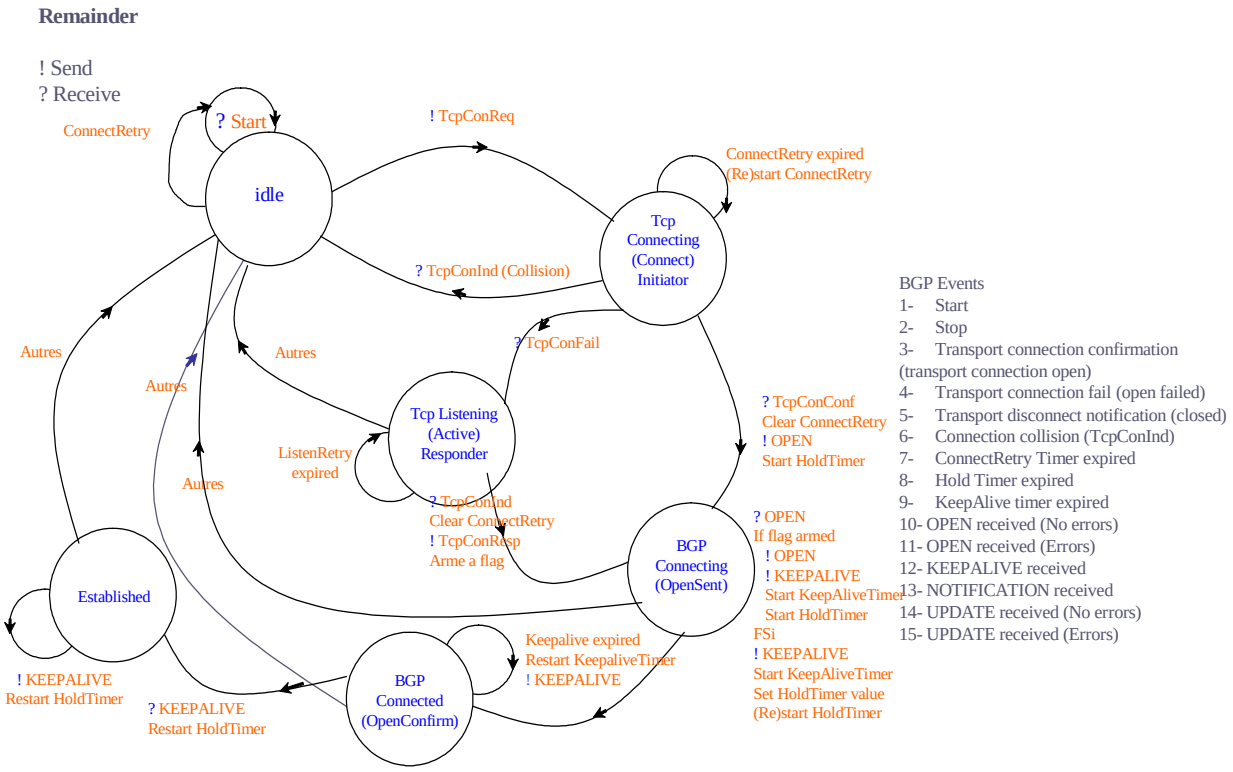


Figure III-8: Machine à état BGP modifiée

Les différents états et les actions à exécuter dans chaque état sont présentés par une table de décision complète.

Events / States	**Start**	**TCP connection success (TcpConConf)**	**TCP connection fails (TcpConFail)**	***ConnectRetry* expiration**	**BGP OPEN received (No errors)**	**BGP OPEN received (Error)**	**TCP Connection Indicator (TcpConInd)**	**Hold Time expiration**	**TCP disconnect notification**
Idle o Refuse all incoming connections) o No resources allocated to the peer	o Change state to "***TCP Connecting***"	Ignored	Ignored	Restart avec un décalage exp.	Ignored	Ignored	Ignored	Ignored	Ignored
TCP Connecting Waiting for the TCP Connection	o Initialize resources o Initialize TCP Connection *(TcpConReq)* o Start ConnectRetry	o Clear *ConnectRetry* timer o Send **OPEN** o Start *Hold Time* o Change state to "***BGP Connecting***"	o *Restart ConnectRetry* timer o Change state to "***TCP Listening"***	o *Restart ConnectRetry* timer o Still in "***TCP Connecting"*** state	Change state to "***Idle"***	Change state to "***Idle"***	Change state to "***Idle"***	Change state to "***Idle"***	Change state to "***Idle"***
TCP Listening	Ignored	o Send **NOTIFICATION(FSM Error)** Change state to "***Idle"***	Change state to "***Idle"***	Restart n fails Still in "***TCP Connecting"*** state	Change state to "***Idle"***	Change state to "***Idle"***	Armer flag Change state to "***BGP Connecting"***	Change state to "***Idle"***	Change state to "***Idle"***
BGP Connecting Waiting for OPEN message from the peer	Ignored	o Send **NOTIFICATION(FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Traitement HoldTimer o Send **KEEPALIVE** o Start *KeepAlive* timer o Test flag o Change state to "***BGP Confirmation"***	o Send **NOTIFICATION** o Change state to "***Idle"***	o Send **NOTIFICATION** o Change state to "***Idle"***	o Send **NOTIFICATION** o Change state to "***Idle"***	o Change state to "***Idle"***
BGP Confirmation Waiting for KEEPALIVE	Ignored	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION** o Change state to "***Idle"***	Change state to "***Idle"***
BGP Established o Start KeepAlive timer o Send **KEEPALIVE**	Ignored	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION** Change state to "***Idle"***	Change state to "***Idle"***

… / ….	**Stop**	**KEEPALIVE received**	**NOTIFICATION received**	***KeepAlive* expiration**	**UPDATE received (No errors)**	**UPDATE received (Errors)**
….	Ignored	Ignored	Ignored	Ignored	Ignored	Ignored
….	Change state to "***Idle"***	Change state to "***Idle"***	Change state to "***Idle"***	Change state to "***Idle"***	Change state to "***Idle"***	Change state to "***Idle"***
….	Change state to "***Idle***	o Send **NOTIFICATION(FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION(FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION(FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION(FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION(FSM Error)** Change state to "***Idle"***
….	o Send **NOTIFICATION** o Change state to "***Idle"***	o Send **NOTIFICATION(FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***
….	o Send **NOTIFICATION** o Change state to "***Idle"***	o Restart Hold Time o Change state to " **BGP Established"**	Change state to "***Idle"***	o Send **KEEPALIVE** o Restart *KeepAlive* o Still in "***BGP Confirmation***" state	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***	o Send **NOTIFICATION (FSM Error)** Change state to "***Idle"***
….	o Send **NOTIFICATION** Change state to "***Idle"***	Armer le HoldTimer Still in "***BGP Established***" state	Change state to "***Idle"***	Envoyer KeeAlive Armer le HoldTimer Still in "***BGP Established***" state	Armer le HoldTimer Still in "***BGP Established***" state	o Send **NOTIFICATION** Change state to "***Idle"***

Tests de fonctionnalité

Il s'agit d'étudier le comportement de BGP dans le réseau comme illustré dans la Figure III-9. L'étude de comportement se résume en trois types d'études : étude des problèmes qui surviennent suite au routage BGP, identifier leurs causes, identifier leurs symptômes et finalement proposer des solutions.

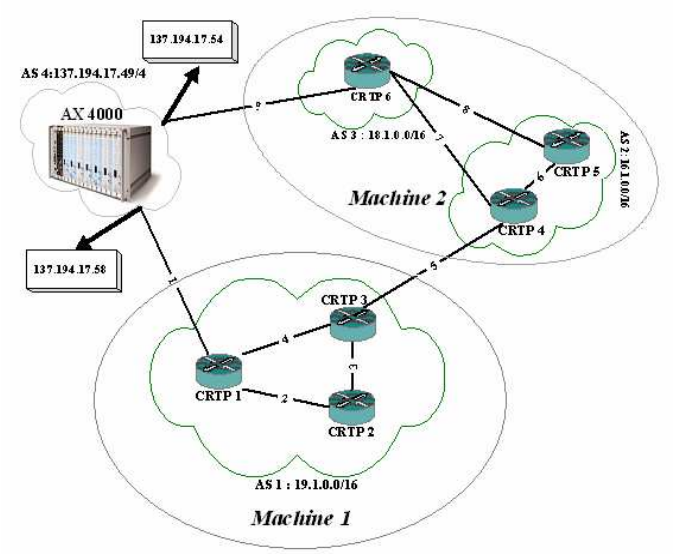


Figure III-9: Etude de fonctionnalité de BGP

Nous avons classifié deux types de problèmes : instabilité et convergence.

- *Problème de stabilité*

Nous définissons l'instabilité du routage comme étant l'état dont les changements trop rapides ne permettent pas d'avoir la durée nécessaire au traitement des informations d'accessibilité et de topologie.

Les symptômes d'une instabilité que nous avons identifiés sont les suivants (utilisées par *POrg*) :

- Routes implicitement annulées et remplacées par les mêmes routes (dans le même message).
- Envoi répétitif d'annulation d'une route déjà inaccessible.

Les causes d'une instabilité du routage BGP sont :

- Choix d'implémentation des constructeurs : ceci est dû au non maintien des informations sur les anciennes annonces de routes (messages UPDATE)
- Mauvais paramétrage des timers : le problème des timers est une des causes majeures à l'instabilité de BGP, ce problème est causé généralement par deux phénomènes :
  - Auto-synchronisation : c'est-à-dire que tous les routeurs BGP dans le réseau, même si initialement leurs timers ne sont pas synchronisés, au bout d'un certain moment ils se synchronisent causant l'envoi des messages BGP de tous les routeurs en même temps, causant à son tour une surcharge des routeurs.
  - Timers CSU : les liens T1 et T3 utilise des modems haut débit (Channel Service Unit). Une mauvaise configuration du CSU cause une oscillation de la ligne causant ainsi l'instabilité des tables de routage BGP (annonces et suppressions successives des mêmes routes).

Plusieurs solutions peuvent être envisagées (par les agents POrg) afin de limiter l'impact de l'instabilité dans le réseau :

- Donner la plus haute priorité aux messages BGP (prévenir les problèmes de surcharge des routeurs)
- Amortir les routes: ne pas accepter des Update si le nombre d'Update excède un nombre fixé par heure par exemple
- Retenir une partie des messages Update

- *Problème* de convergence

Nous définissons la convergence le fait *d'atteindre une vue consistance du réseau après une panne de changement de topologie (politique).*

Les causes d'une lenteur de convergence du routage BGP sont :

- Choix d'implémentation des constructeurs : qui sont de deux natures :
  - Application du timer *MinRouteAdverInterval*, qui est un timers contrôlant l'envoi des annonces de nouvelles routes ou les mises à jour des routes, aux messages d'annulation qui indiquent la non accessibilité d'une route.

- La détection des boucles ne se fait qu'après la réception des annonces (messages UPDATE). Causant l'envoi de ces messages aux peers qui sont déjà inclusent dans ces messages (boucle).

Plusieurs solutions peuvent être envisagées afin de limiter l'impact de l'instabilité dans le réseau :

- Détection des boucles à l'émission et la réception des annonces
- Ne pas appliquer le timer MinRouteAdvertInterval sur les messages d'annulation

La plate-forme Pilote Organisationnel

Ce chapitre présente le début d'implémentation de *POrg*. Dans un premier temps, nous présentons l'implémentation de l'algorithme Path-Based Clustering à travers un simulateur que nous avons développé dans le cadre du projet IA/PO (§7.1). Les résultats des plates-formes que nous venons de décrire dans le chapitre précédent, seront intégrées au cœur du prototype IA/PO. Dans la section (§7.2), nous rajoutons les Uses Cases d'implémentation, à savoir, dans un environnement expérimental.

7.1 Implémentation du simulateur Path-Based Clustering

Après avoir décrit les particularités de notre mécanisme de clustering, nous nous proposons durant ce chapitre de décrire son implémentation. Le but de l'implémentation est de valider le bon fonctionnement de notre algorithme ainsi que d'étudier son efficacité.

Nous commencerons ce chapitre par décrire l'environnement de développement (§7.1.1) et nous présenterons à la fin les résultats recueillis par notre implémentation. Nous montrerons quelques interfaces de visualisation pour le mécanisme de clustering (§7.1.2) et un travail d'analyse est élaboré à la fin pour décrire les résultats de l'algorithme (§7.1.3).

7.1.1 Environnement de développement

Pour implémenter notre mécanisme de clustering, nous avons choisi de développer un simulateur de clustering dans le langage Java ce qui va nous permettre d'étudier la faisabilité de notre méthode.

Le choix du langage Java était dans le but de rejoindre les travaux du groupe qui a choisi une implémentation java pour projet IA/PO. Cette harmonisation du langage pour les intervenants dans le projet facilitera éventuellement l'interfaçage du présent travail avec d'autres modules IA/PO qui seront développés ultérieurement.

L'environnement de développement qui accueille l'implémentation du simulateur est NetBeans IDE 4.1 de l'éditeur Sun Microsystems Inc.

7.1.2 Description du simulateur

Notre simulateur prend en charge une matrice d'adjacence d'une topologie réseau qui renferme des nœuds et des liens ainsi que des coûts sur les liens. Le simulateur procède de la façon suivante :

- Création de la topologie réseau en fonction de la matrice d'adjacence
- Lecture des paramètres de départ (Nœud périphérique, distance effective)
- Calcule des clusters
- Affichage du résultat de clustering ainsi que du temps d'exécution du mécanisme

Dans la première version du simulateur, nous procédions par une entrée manuelle de la matrice d'adjacence ainsi qu'un affichage textuel des résultats du clustering. Dans le but d'apporter une visualisation en guise de démonstration de notre simulateur de clustering, nous avons prévu une interface graphique présentant trois topologies réseau sur lesquelles nous pouvons visualiser le résultat du clustering. La Figure III-10 décrit cette interface avec la topologie choisie dans le chapitre précédant pour l'explication de Path-based clustering

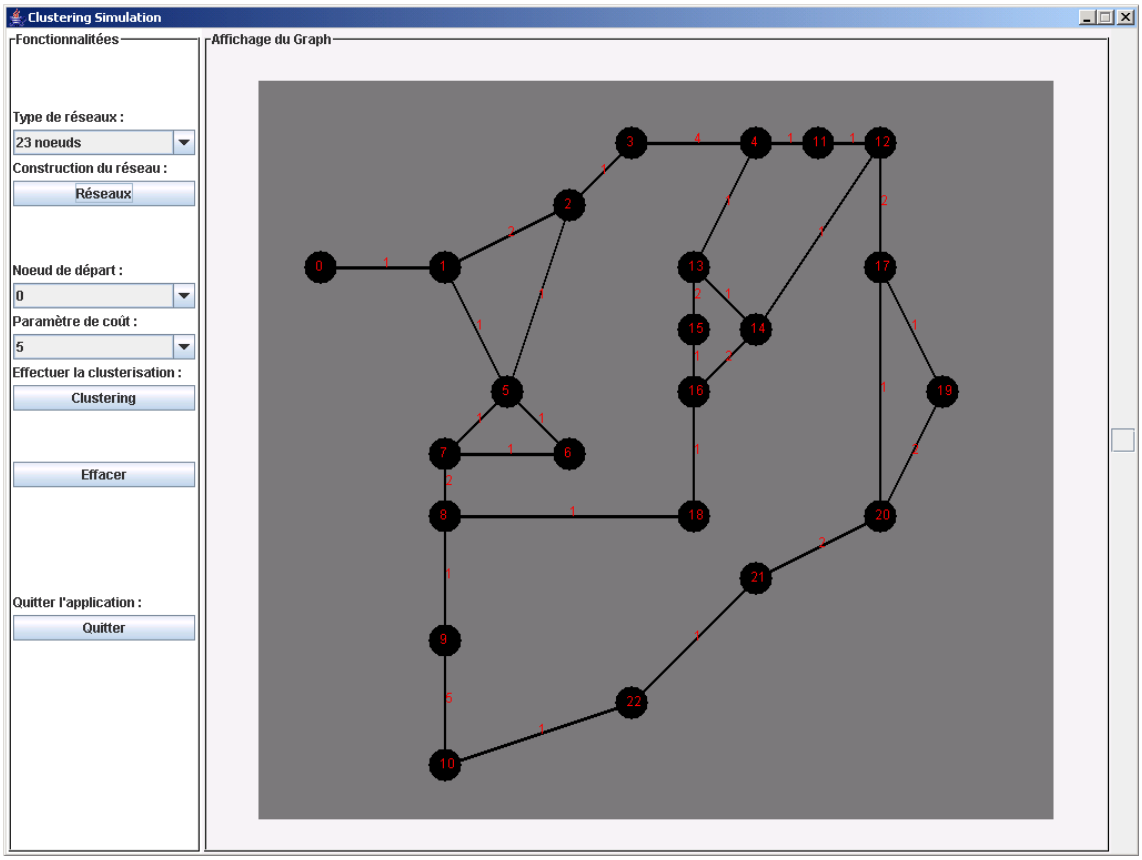


Figure III-10: Le simulateur de PBC

Cette interface prend en charge le choix de l'utilisateur du nœud de départ ainsi que la distance effective. L'action sur le bouton « Clustering » laisse la main au programme afin qu'il puisse procéder aux calculs nécessaires pour la construction des clusters sur la topologie. Le résultat est affiché par la suite avec la coloration des clusters ainsi que la coloration des points ES (Entrée/Sortie) pour délimiter les bordures des clusters établis. La Figure III-11 montre l'affichage final du clustering que nous avons lancé à partir du nœud 0 avec une distance effective de *9* :

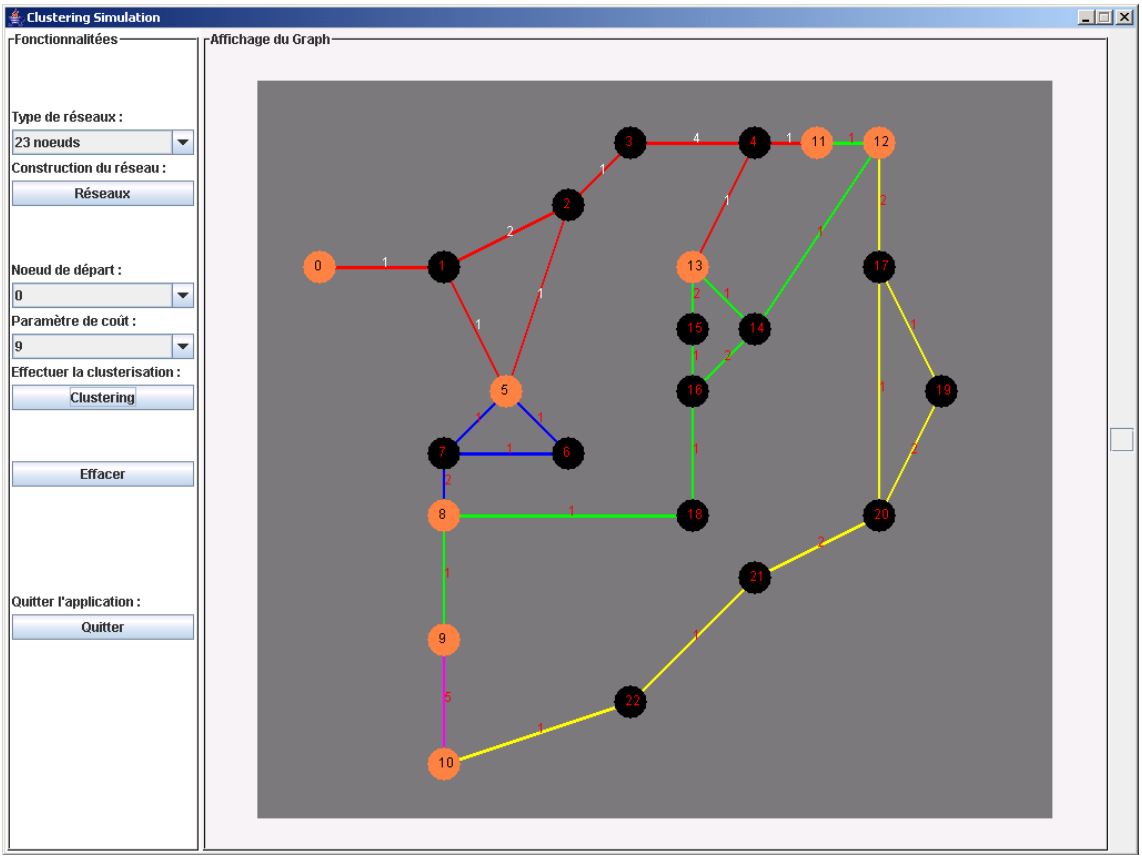


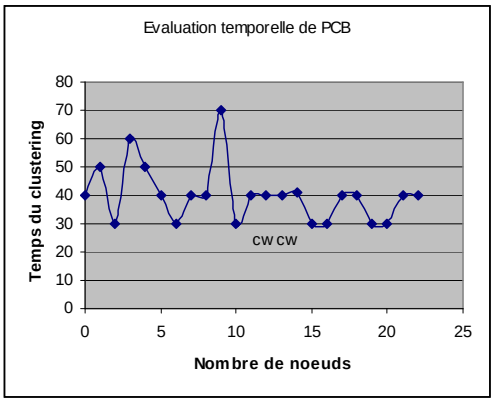
Figure III-11: Affichage du résultat de PBC

7.1.3 Performances de PBC

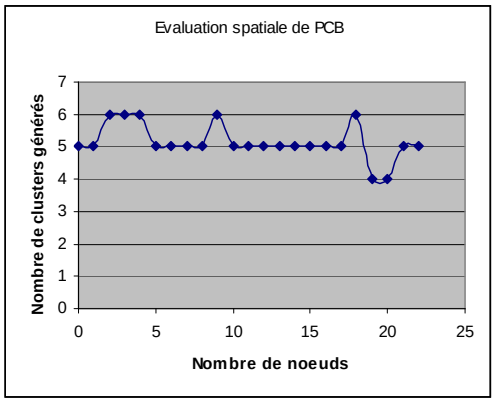
Afin d'avoir une idée sur la fiabilité de l'algorithme nous avons relevé le temps de clustering (Tc) sur la topologie que nous avons présentée précédemment toute en faisant varier le point de départ par lequel nous entamons le clustering. Le délai de clustering quand est fixé à *9*. Les résultats recueillis sont résumés la Figure III-12.

Nous constatons à partir des résultats présentés qu'en moyenne nous sommes en présence d'un temps de clustering moyen de 41,01 ms (rappelons que le calcule à été fait sur une machine « Pentium III 700 MHz 256Mo »).

Quand au nombre moyen de cluster construit pour un délai de clustering de 9 il est de *5,13* clusters.



(a) Temps de clustering de PCB



(a) Nombre de clusters générés par PCB

Figure III-12: Performances de PBC

Ces calculs nous permettent de valider le bon fonctionnement de notre algorithme de clustering ainsi que sa rapidité d'exécution malgré la combinatoire importante du problème.

Nous avons présenté dans cette section l'implémentation de notre mécanisme de clustering «Path-based clustering ». Cette implémentation nous confirme le bon fonctionnement de notre mécanisme et sa faisabilité. Par ailleurs, il est important de souligner que d'autres critères d'évaluation pourront être fait afin d'enrichir d'avantage l'étude des performances du Path-based Clustering.

7.2 Cas d'Utilisations

Les résultats des plates-formes que nous venons de décrire dans le chapitre précedent, seront intégrées au cœur du prototype IA/PO. Dans cette section, nous rajoutons les Uses Cases d'implémentation, à savoir, dans un environnement expérimental.

Afin que POrg interagissent avec l'environnement, nous avons spécifié des modules d'interaction. Ils permettent l'implémentation des agents POrg indépendamment de la

technologie de transport utilisée. Deux cas d'utilisation ont été implémentés : Analyse_Message (§7.2.1), DB_Interact (§7.2.2) et PO_Interact (§7.2.3).

7.2.1 Le Cas d'Utilisation « Analyse_Messages »

Ce cas d'utilisation permet de formater les messages de/à l'agent POrg des nœuds (notifications) et les autres agents *POrg* (requêtes) (Figure III-13). Puis par raffinement (Figure III-14) on aura le traitement complet des messages.

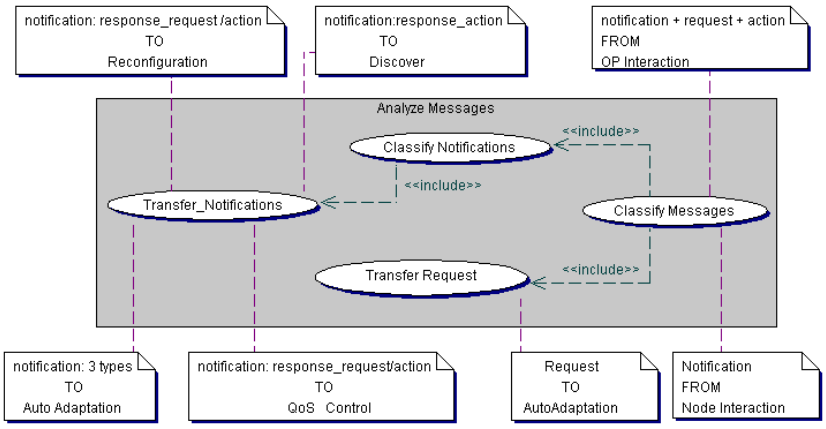


Figure III-13: Diagramme de Use Case de Analyse_Message

Pour analyser ces messages, on fait d'abord une première analyse pour savoir s'agit d'une requête (provenant d'un agent POrg) ou d'une notification (provenant d'un agent POrg ou d'un nœud). Cette analyse est faite à travers le Use Case « Classify_Message ». Ensuite, s'il s'agit d'une notification, on analyse ce message suivant le schéma décrit dans Figure III-14. Nous transférons ces messages, suivant cette analyse, pour qu'ils soient utilisés par les autres Use Case de premier niveau.

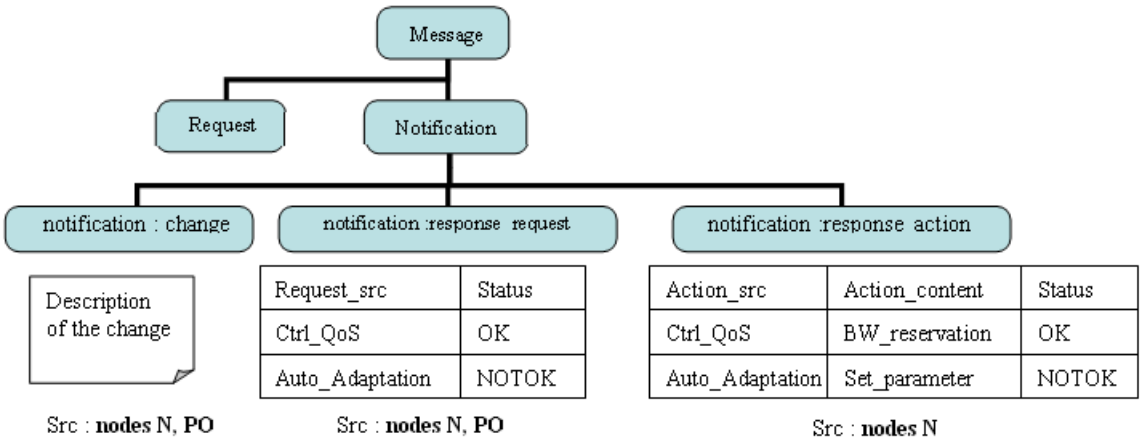


Figure III-14: Structure des messages échangés entres les agents POrg

7.2.2 Le Cas d'Utilisation « DB Interact »

Afin que les agents POrg n'aient pas d'interactions bloquantes avec les bases de données locale et globale, nous avons spécifié cet Use Case qui sépare entre les requêtes entrantes et les requêtes sortantes.

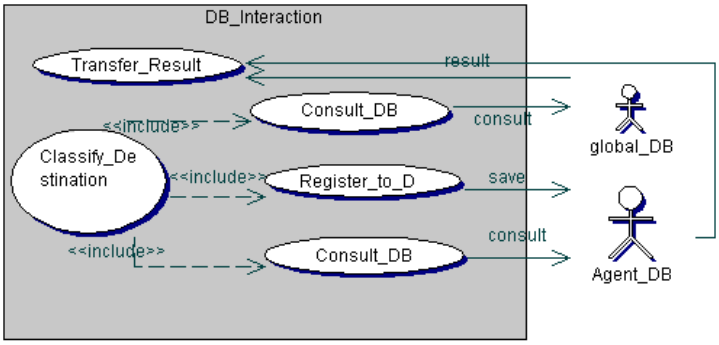


Figure III-15: Diagramme de Use Case de DB_Interact

7.2.3 Le Cas d'Utilisation « PO Interact »

Cet Use Case abstrait les interactions entre les agents POrg. Nous sélectionnons le PO destinataire par le Use Case « Classify Destination » et transférons les messages au PO destinataire (« Transfer »).

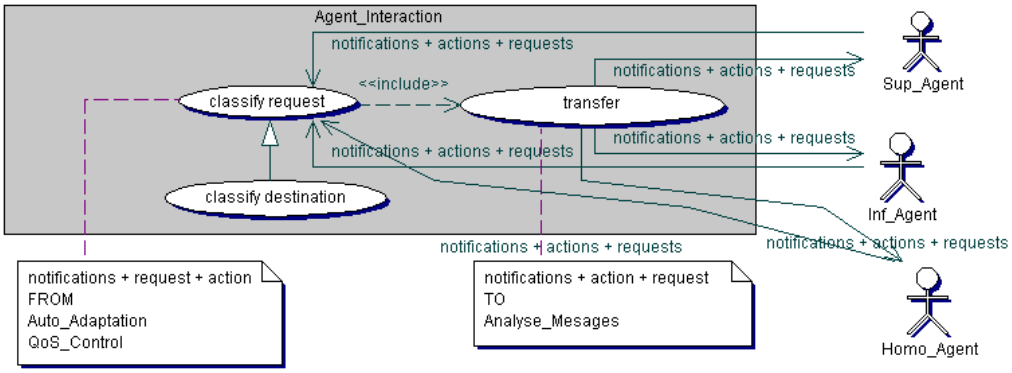


Figure III-16: Diagramme de Use Case de PO_Interact

CONCLUSION GENERALE ET PERSPECTIVES

Conclusion

Grâce à la convergence des services de télécommunication, l'usage des services ne se réduit plus aux communications téléphoniques, mais à une vision plus large du futur numérique. Le monde des télécommunications devient plus dépendant des désirata des usagers des services en termes de perception de la QoS et de personnalisation, mais la capacité des architectures actuelles à répondre à ces besoins demeure une question ouverte. L'un des objectifs principaux est de rendre les architectures de télécommunication *flexibles* en terme d'ajouts de nouveaux services à moindre coût et *réactives* aux changements de l'environnement afin de maintenir le niveau de service contracté par les usagers, qui n'est autre que la QoS de bout en bout. De ce fait, la conception d'architectures intégrées, c'est-à-dire, des architectures où les changements préconisés dans un niveau donné (réseau, service, etc.) peuvent être évalués et quantifiés par rapport au comportement global perçu par l'usager final, est devenue une nécessité absolue.

L'objectif de ce travail a été de proposer des clés à cette problématique en mettant en œuvre un concept d'intégration des architectures de télécommunications en incluant tous les acteurs superposés de la chaîne de bout en bout : équipements matériels, le réseau de transport, les service et les usagers de ces services afin de supporter leurs évolutions trans-générations.

L'étude des propositions existantes, dans la première partie, nous a permis de formuler les besoins et dégager un cahier de charge de ce qu'on appelle une architecture intégrée. Cette formulation est d'autant plus importante que la tâche à laquelle sont confrontées les architectures actuelles, qui est de plus en plus complexe.

Pour la réaliser, nous avons proposé, dans la deuxième partie, un concept baptisé : Ingénierie d'Architecture. Il s'agit de l'assemblage dynamique des composants d'une architecture à des fins de QoS de bout en bout. Son objectif principal est d'assurer l'adéquation entre les besoins des utilisateurs et l'exploitation du réseau de l'opérateur pour une QoS de bout en bout, tout en tenant compte des contraintes de l'existant afin d'avoir une architecture *flexible*.

A des fins d'ingénierie d'architecture, nous avons modélisé l'architecture globale d'un fournisseur de service en quatre (4) niveaux : équipements, réseaux, services et usagers. Ces quatre (4) niveaux identifient au mieux les différents acteurs de la chaîne de QoS de bout en bout. Cette QoS est impactée par (i) les choix de superpositions des blocs architecturaux d'un côté, et (*intégration verticale*) (ii) et du respect du contrat de QoS confié au niveau considéré (*intégration horizontale*). Ces deux modèles constituent les composantes de base de l'ingénierie d'architecture.

Pour l'intégration verticale, nos avons proposé un modèle de *cohabitation* qui s'assure de la consistance de la structure d'ensemble. Notre approche repose sur deux principes : principe de la *déclinaison* de la QoS et principe de *l'agrégation* de la QoS.

- La déclinaison de la QoS fait référence au processus d'analyse de l'architecture globale de haut en bas (top-down) en visant deux objectifs : (*i*) trouver une solution pour une QoS de bout en bout, (*ii*) maintenir la cohérence de la structure d'ensemble.
- L'agrégation fait référence au processus d'analyse de l'architecture globale de bas en haut (bottom-up), c'est à dire les couches supérieures doivent recevoir des informations de rétroaction des couches inférieures afin que les niveaux supérieurs s'assurent du bon déroulement de la déclinaison et éventuellement relancer le processus de déclinaison suivant ces informations de rétroaction.

Un processus générique, que nous avons dénommé DECIDERA, regroupant les processus de déclinaison et agrégation a été proposé. Ce processus met en exergue tous les acteurs et fonctions rentrant dans les processus de déclinaison et agrégation, à travers des étapes allant du SLA à la configuration des éléments du réseau.

Pour l'intégration horizontale, nous avons proposé un modèle de coopération pour superviser et évaluer en temps réel la QoS offerte à travers les différents « sous-réseaux » (domaine) constituant le niveau horizontal (équipement, réseau, service ou usagers). Chaque domaine est autonome, c'est-à-dire, responsable de la supervision de sa propre QoS (SLA entre domaines). Si une dégradation de la QoS est constatée sur un (ou plusieurs) domaine(s), et

donc non-conformité au contrat de QoS, nous avons proposé de résoudre le problème en trois (3) étapes:

1. re-négocier le contrat de QoS (SLA) entre les domaines de telle sorte de rattraper la dégradation constatée dans le domaine *i* dans le domaine suivant (*i*+1).
2. changer l'itinéraire des flux traversant le domaine sur lequel on constate une dégradation pour les faire passer par un autre domaine qui aurait les ressources nécessaires aux besoins du flux, ceci est assuré à travers l'interaction avec le protocole de routage en vigueur (routage réseau, routage applicatif, etc.).
3. ré-organiser les domaines de telle sorte que des ressources soient disponibles de bout en bout horizontal.

Si ces trois étapes n'ont pas abouti à une solution, un feedback négatif au niveau supérieur indiquant la non possibilité d'accomplir le contrat vertical pour qu'il prenne des décisions (modifier les besoins, réorganiser le niveau *N*+1, etc.) à travers le principe d'agrégation.

Afin de réorganiser les domaines de telle sorte que des ressources soient disponibles de bout en bout horizontal, nous avons proposé un algorithme de clustering basé sur la disponibilité des ressources dans les domaines. Nous l'avons appelé *Path-Based Clustering*.

Afin de mettre ces modèles en œuvre, nous avons conçu Pilote Organisationnel (POrg), une plate-forme d'ingénierie d'architecture, suivant trois dimensions : dimension organisationnelle qui définit le rôle, la portée et l'emplacement de chaque entité composant la plate-forme POrg dans l'architecture, la dimension fonctionnelle qui définit l'ensemble des fonctions et actions à mettre en œuvre par POrg pour assurer l'ajout de nouveaux services et maintenir la QoS de bout en bout de manière transparente et la dimension architecturale qui définit la structure et la coordination des modules composant *POrg*.

Un début de réalisation d'un prototype POrg, notamment, le processus DECIDERA et Use Cases Contrôle_QoS et Auto-Adaptation, nous conforte dans la faisabilité de l'intégration des architectures de télécommunications. Un démonstrateur doit être présenté à l'équipe R&D de SFR, partenaire de ce projet IA/PO.

Nos travaux sur l'extraction des règles d'ingénierie de la QoS ont été adoptés par des organismes officiels tel que l'AFUTT (Association Française des Utilisateurs des Télécommunications) [91]. Nous avons ainsi fait une révision des normes ETSI concernant la perception des usagers et leur point de vue de la QoS.

Perspectives

Il nous semble particulièrement important par la suite d'étudier deux perspectives de notre travail, perspectives à court terme (immédiates), et perspectives à moyen et long terme:

Les perspectives à court terme se résument à :

- la continuation de l'implémentation du prototype POrg en se focalisant sur les processus de déclinaison et agrégation
- l'étude approfondie du comportement des applications et le réseau dans le contexte opérateur afin de dégager des règles d'ingénierie avec plus de recul et de maturité.
- l'étude d'extension de l'algorithme Path-based Clustering (QCP) afin de prendre en compte le clustering multi-critères (de QoS) avec étude de performances plus approfondie.

Les perspectives à moyen et long terme consistent à :

- compléter la spécification de POrg pour finaliser avec la dimension de communication générique qui définit les échanges entre les modules des agents POrg indépendamment du réseau de transport, ainsi que la dimension informationnelle qui pour avoir une image structurelle et comportementale des entités composant *POrg*.
- la mise en service de POrg dans un contexte réel (opérateur) avec définition de son modèle métier (business model).

PUBLICATIONS

- Tarek Nadour, Noémie Simoni, Antoine Boutignon, “Where are we Going with IP Multimedia Subsystem (IMS)?” *4th International Information and Telecommunication Technologies Symposium (I2TS'2005), Florianópolis, Brazil 2005.*
- Tarek Nadour, Noémie Simoni, Ghislain Du-Chéné, "Pilotage QoS Dépendant de l'Organisation des Architectures: Prototype", *GRES'05, Luchon, France.*
- Fayçal Bennani, Tarek Nadour, Noémie Simoni, " QoS de Bout en Bout: vue Usagers vs. vue Fournisseur", *GRES'05, Luchon, France.*
- Tarek Nadour, Noémie Simoni, Ghislain Du Chéné, "Towards Dynamic Vertical Self-Organization", *IEEE International Conference on Networks (ICON'04), November 2004, Singapore.*
- Tarek Nadour, Noémie Simoni, Antoine Boutignon, "Architecture Engineering: Mastering Architecture Evolution and Traffic Engineering Rules", *IEEE International Conference on Communication (ICC'04), Juin 2004, Paris.*

REFERENCES

[1] SASAN ROSTAMBEIK, NOEMIE SIMONI, "Virtual and Personalized Access to Services using a Personalization Middleware", *IEEE Computer Communications journal on Emerging Middleware for Next Generation Networks, 2006.*

[2] ROGER L. FREEMAN, "Fundamentals of Telecommunications- 2nd edition." *IEEE Press, 2005*.

[3] C. RIGAULT, "Principes de commutation numérique, du téléphone au multimédia", *ISBN 2-86601-690-4, Hermes, 1998.*

[4] "Telecommunication Transmission Engineering, 2nd edition", *American Telephone and Telegraph Co. New York, 1977.*

[5] S. R. ALI, "Implementation of Firmware on SPC Switching Systems", *IEEE Journal on Selected Areas in Communications, Vol. 6, no. 8, Octobre 1988.*

[6] "The Basics of Telecommunications, 5th edition", *International Engineering Consortium, ISBN: 0-931695-23-7., 2005.*

[7] "Specifications of Signaling System No. 7 (Q.700 Series)", *CCITT Yellow Books, 7ème Assemblée Plénière, Genève, 1980.*

[8] "Signaling System No. 7-Signaling Network Structure », *Recommandation UIT-T. Q.705, 1993.*

[9] "Specifications of Signalling System n°7: Signalling Data Link", *Recommendation UIT-T, Q.702, 1993.*

[10] "Switching and Signalling: Specifications of Signalling System n°7- Message Transfer Part: Signalling Link", *Recommendation UIT-T, Q.703, 1996.*

[11] "Switching and Signalling: Specifications of Signalling System n°7- Message Transfer Part. Signalling Network Functions and Messages", *Recommendation UIT-T, Q.704, 1996.*

[12] "Specifications of Signalling System n°7: Signalling Connection Control Part", *Recommendation UIT-T, Q.711-Q.716, 1996.*

[13] "Specifications of Signalling System n°7: ISDN User Part", *Recommendation UIT-T, Q.761-Q.767, 1997.*

[14] "Specifications of Signalling System n°7: Transaction Capabilities Application Part", *Recommendation UIT-T, Q.771-Q.775, 1997.*

[15] "Specifications of Signalling System n°7: Functional Description of the Signalling System n°7, Telephone User Part (TUP)", *Recommendation UIT-T, Q.721-Q.725, 1993.*

[16] BANCEL-CHARENSOL, « La déréglementation des télécommunications dans les pays industrialisés », *Economica. Paris. 1996.*

[17] "Glossary of terms used in the definition of intelligent networks", *Recommendation UIT-T, Q.1290, 1998.*

[18] P. ALMQUIST, "Type of Service in the Internet Protocol Suite", *Internet RFC 1349, Juillet 1992.*

[19] UYLESS BLACK, "Frame Relay Networks: Specifications & Implementations", *MacGraw Hill Series on Computer Communications, ISBN 0-07-005558-0, 1994.*

[20] DARPA, "Internet Protocol: Internet Program Protocol Specification", *Internet RFC 791, Spetembre 1981.*

[21] G. PUJOLLE, "Les Réseaux ", *Eyrolles, ISBN 2-212-11437-0, 2005.*

[22] ATM FORUM/94-1015, "On Models of Internetworking", *1994*

[23] J. LUCIANI, "Classical IP to NHRP Transition", *Internet RFC 2336, Juillet 1998.*

[24] F. BENNANI, "IP et la QoS: vers une maîtrise dynamique de bout en bout", *thèse de doctorat, ENST Paris, 2002.*

[25] IETF, "Multiprotocol Label Switching (mpls) Working Group", *http://www.ietf.org/html.charters/mpls-charter.html*

[26] D. AWDUCHE, J. MALCOM, J. AGOGBUA, M. O'DELL, J. MCMANUS, "Requirements for Traffic Engineering over MPLS", *IETF RFC, September 1999.*

[27] E. ROSEN, A. VISWANATHAN, R. CALLON, "Multiprotocol Label Switching Architecture", *Internet RFC 3031, January 2001.*

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

[29] S. SHENKER, C. PATRIDGE, R. GUERIN, « Specification of Guaranteed Quality of Service », *Internet RFC 2212, Septembre 1997.*

[30] J. WROCLAWSKI, “Specification of the Controlled-Load Network Element Service”, *Internet RFC 2211, Septembre 1997.*

[31] R. BRADEN, D. HOFFMAN, “RAPI – An RSVP Application Programming Interface”, *Version 5, Internet Draft, work in progress, August 1998.*

[32] “The MicroSoft QoS Components”, *http://www.microsoft.com/technet/prodtechnol/ windows2000serv/maintain/featusability /qoscomp.mspx, 2005.*

[33] Differentiated Services Working Group: http://www.ietf.org/html.charters/diffserv-charter.html

[34] S. BLACK, D. BLACK, M. CARLSON, E. DAVIES, Z. WANG, W. WEISS, “An Architecture for Differentiated Services*”, IETF RFC2475, December 1998.*

[35] CISCO SYSTEMS, “LAN QoS*”, Internet Protocol Journal, Vol. no. 4, Issue no. 1, March 2001.*

[36] F. AGHAREBPARAST AND V.C.M. LEUNG, "QoS support in the UMTS/GPRS backbone network using DiffServ", *in Proc. IEEE Globecom'02, Taipei, ROC, Nov. 2002.*

[37] V. JACOBSON, K. NICHOLS, K. PODURI, “An Expedited Forwarding PHB”, *Internet RFC 2598, Juin 1999.*

[38] J. HEINANE, F. BAKER, W. WEISS, J. WROCLAWSKI, “Assured Forwarding PHB Group”, *Internet RFC 2597, Juin 1999.*

[39] J. MOY, “OSPF Version 2”, *IETF RFC 2328, April 1998.*

[40] IS-IS FOR IP INTERNETS (ISIS) WORKING Group:http://www.ietf.org/html.charters/isis-charter.html

[41] E. W. DIJKSTRA, “A note on two problems in connexion with graphs”, *In Numerische Mathematik. 1, 1959,, pp. 269–271.*

[42] G. APOSTOLOPOULOS, D. WILLIAMS, S. KAMAT, R. GUERIN, A. ORDA, T. PRZYGIENDA, “QoS Routing Mechanisms and OSPF Extensions”, *IETF RFC 2676, Août 1999.*

[43] K. ISHIGURO, T. TAKADA, ‘Traffic Engineering Extensions to OSPF Version 3”, http://www.ietf.org/internet-drafts/draft-ietf-ospf-ospfv3-traffic-01.txt, *IETF Work in Progress, August 2003.*

[44] H. SMIT, T. LI, “IS-IS Extensions for Traffic Engineering”, http://www.ietf.org/internet-drafts/draft-ietf-isis-traffic-05.txt, *IETF Work in Progress, August 2003.*

[45] R. COLTUN, "The OSPF Opaque LSA Option", IETF RFC 2370. Juillet 1998.

[46] Y. REKHTER, T. LI, "A Border Gateway Protocol 4 (BGP-4)", *IETF RFC, March 1995.*

[47] B. QUOITIN, C. PELSSER, L. SWINNEN, O. BONAVENTURE, S. UHLIG, "Interdomain Traffic Engineering With BGP", *IEEE Communication Magazine, May 2003.*

[48] O. BONAVENTURE, "Using BGP to distribute flexible QoS information", *Internet Draft draft-bonaventure-bgp-qos-00.txt, Work in progress, February 2001.*

[49] "Traffic Engineering for Quality of Service in the Internet, at Large Scale, TEQUILA", *http://www.ist-tequila.org/*

[50] B. ABARBANEL, S. VENKATACHALAM, "BGP-4 Support for Traffic Engineering", *Internet drafts draft-abarbanel-idr-bgp4- te-00.txt. Work in progress, September 2000.*

[51] A. FEI AND M. GERLA, "Extending BGMP for Shared-tree Interdomain QoS Multicast". *In Proceedings of IWQoS, 2001.*

[52] R. BRADEN ED., L. ZHANG, S. BERSON, S. HERZOG, S. JAMIN, " Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", *Internet RFC 2205, Septembre 1997.*

[53] P. PAN, H. SCHULZRINNE, "YESSIR: a simple reservation mechanism for the Internet", *ACM SIGCOMM Computer Communication Review, Vol. 29, Issue 2, Avril 1999.*

[54] D. AHLARD, J. BERGKVIST, I. CSELENYI, T. ENGBORG, "Boomerang - A Simple Resource Reservation Framework for IP", *draft-ahlard-boomerang-framework-00.txt.*

[55] D. AWDUCHE, L. BERGER, T. LI, V. STRINIVASAN, G. SWALLOW, "RSVP-TE: Extensions to RSVP for LSP Tunnels", *IETF RFC, December 2001.*

[56] J. P. VASSEUR, A. CHARNY, F. LE FAUCHEUR, J. ACHIRICA, J. J. LEROUX, "MPLS Traffic Engineering Fast reroute: bypass tunnel path computation for bandwidth protection", draft-vasseur-mpls-backup-computation-02.txt, *IETF Work In Progress, February 2003*

[57] S. B. MOON, J. KUROSE, D. TOWSLEY, "Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms", *ACM Multimedia Systems, 1999.*

[58] S. FOSSE-PARISIS BOLOT, D. TOWSLEY, "Adaptive FEC-Based Error Control for Interactive Audio in the Internet", *IEEE Infocom'99, New York, USA, March 1999.*

[59] F. J. MACWILLIAMS, N. SLOANE, "Theory of Error-Correcting Codes", *North-Holland, 1977.*

[60] J. BLOMER, M. KALFANE, R. KARP, M. KARPINSKI, M. LUBY, D. ZUCKERMAN, "An XOR-Based erasure-resilient coding scheme", *Rapport n°TR-95-048, International Computer Science Institute, Berkeley California, Août 1995.*

[61] J. ROSENBERG, H. SCHULZRINNE, "An RTP Payload Format for Generic Forward Error Correction", *Internet RFC 2733, December1999.*

[62] C. PERKINS, "An RTP Payload format for Reed Solomon Codes", *IETF RFC 2198, Septembre 1997.*

[63] J. SANDVOSS, J. WINCHLER, H. WITTIG, "Network Layer Scalling: Congestion Control in Multimedia Communication with Heterogenous Networks", *Rapport n°43.9401 IBM European Networking Center, Heidelberg, 1994.*

[64] H. SCHULZRINNE, S. CASNER, R. FREDERICK, V. JACOBSON, "RTP: A Transport Protocol for Real-Time Applications", IETF RFC 3550, Juillet 2003.

[65] A. CARZANIGA, A. L. WOLF, "Content-Based Networking", *White Paper, Department of Computer Science, University of Colorado. http://www.cs.colorado.edu/~carzanig/papers/content-based_networking.pdf.*

[66] A. CARZANIGA, D. S. ROSENBLUM, A. L. WOLF, "A Routing Scheme for Content-Based Networking", *IEEE INFOCOM 2004.*

[67] B. SEGALL AND D. ARNOLD, "Elvin has left the building: A publish/subscribe notification service with Quenching", *in Proc. of AUUG97, Brisbane, Queensland, Australia, Sept 1997.*

[68] G. BANAVAR, T. D. CHANDRA, B. MUKHERJEE, J. NAGARAJARAO, R. E. STROM, AND D. C. STURMAN, "An efficient multicast protocol for content-based publish-subscribe systems", *in The 19th IEEE International Conference on Distributed Computing Systems (ICDCS '99), Austin, TX USA, May 1999.*

[69] A. CARZANIGA, D. ROSENBLUM, A. WOLF, "Design and Evaluation of a Wide-Area Event Notication Service", *ACM Trans. on Computer Systems, Août 2001.*

[70] G. CUGOLA, E. D. NITTO, AND A. FUGETTA, "The jedi event-based infrastructure and its application to the development of the opss wfms" *IEEE Transactions on Software Engineering, Septembre 2001.*

[71] R. CHAND, P.A. FELBER, “A Scalable Protocol for Content-Based Routing in Overlay Networks”, *IEEE International Symposium on Network Computing and Applications, USA, 2003.*

[72] S. MAFFEIS, “iBus: The Java Intranet Software Bus”, *Technical report, SoftWired AG, Zurich, Switzerland, Feb. 1997.*

[73] G. DERMLER ET AL., "A Negotiation and Resource Reservation Protocol (NRP) for Configurable Multimedia Applications", *in Proc. IEEE International Conference on Multimedia Computing and Systems (ICMCS '96), 1996.*

[74] C. N. CHUAN, “Providing End-to-End QoS for IP-based Latency sensitive Applications”, *Dessertation Proposal, 2001.*

[75] Q. ZHANG, W. ZHU, Y. Q. ZHANG, “A Cross-Layer QoS Supporting Framework for Multimedia Delivery over Wireless Internet”, *International Packet Video Workshop (PV’02), Pittsburgh, PA, 2002.*

[76] J. CHEN, T. LV, H. ZHENG, “Joint Cross-layer Design for Wireless QoS Content Delivery”, *IEEE International Conference on Communication (ICC’04), Juin 2004, Paris.*

[77] “Terms and Definitions Related to Quality of Service and Network Performance Including Dependability”, *ITU-T Recommandation E.800, Août 1994.*

[78] ANDERBERG, “Cluster Analysis for Applications”, *Academic Press, Inc., New York, NY, 1973*

[79] A. K. JAIN, M. N. MURTY, PJ. FLYNN, « Data Clustering: A Review, ACM Computing Surveys», *Vol. 31-3, September 1999.*

[80] J. H. REIF, “Depth-First Search is Inherently Sequential”, *Information Processing Letters, 20(5):229--234, June 1985*

[81] OBJECT MANAGEMENT GROUP, “Unified Modeling Language”, *www.uml.org, 2005.*

[82] J. C. BOLOT, T. TURLETTI, “A Rate Control Mechanism for Packet Video in the Internet”, *IEEE INFOCOMM’94, Toronto, Canada, Juin 1994*

[83] I. BUSSE, B. DEFFNER, H. SCHULZRINNE, “Dynamic QoS Control of Multimedia Applications Based on RTP“, *Rapport de Recherche, MGD-Fokus, Hardenbergplatz 2, D-10623 Berlin, Allemagne, Mai 1995.*

[84] F.BENNNANI, N.SIMONI, “Dynamic Management for End-to-end IP QoS: From requirements to Offers”. *IEEE International Symposium on Computers & Communication (ISCC). France. July 2000.*

[85] R. C. DUBES, JAIN, “Clustering methodology in exploratory data analysis”. *In Advances in Computers, M. C. Yovits, Ed. Academic Press, Inc., New York, NY, 113–125.*

[86] ZAHN, “Graph-theoretical methods for detecting and describing gestalt clusters”, *IEEE Trans. Comput. C-20, 68–86, Avril, 1971.*

[87] M. S. DASKIN, “Network and Discrete Location Model, Algorithms, and Applications”, *John Wiley & Sons, Inc., 1995*

[88] S. KHULLER, Y. J. SUSSMANN, “The Capacitated K-Center Problem”, *SIAM Journal on Discrete Mathematics, Vol. 13-3, 403-418, Society for Industrial and Applied Mathematics, 2000*

[89] S. Y. LU, FU, “A sentence-to-sentence clustering procedure for pattern analysis”, *IEEE Trans. Syst. Man Cybern. 8, 381–389, 1978*

[90] L. ZADEH, “Fuzzy sets”, Inf. Control 8, 338–353, 19

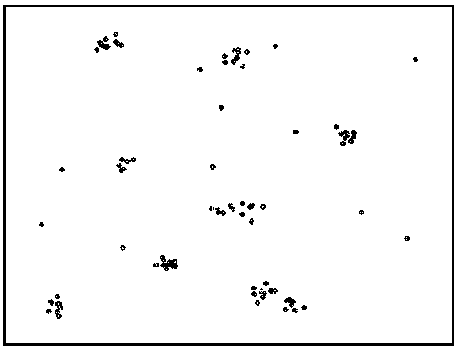
[91] AFUTT, www.afutt.org

[92] ALCATEL, “Intelligent Switch Router Project”, *http://www-rocq.inria.fr/fr/technologie/iliatech/club_iliatech/slides/Randriamasy.pdf.*

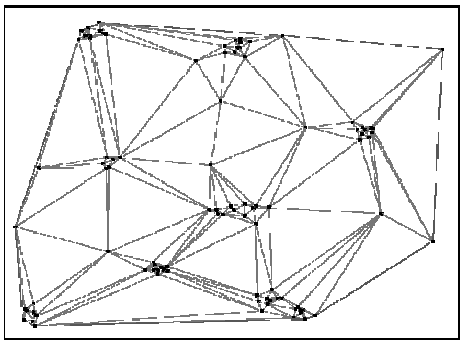
Partie IV ANNEXES

Annexe A Classification des techniques de clustering

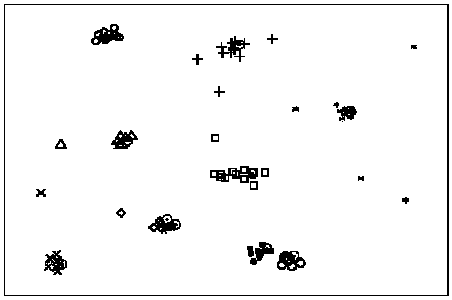
Le clustering peut être décrit comme étant le processus d'organisation d'objets en des groupes partagent une notion d'équivalence propre [79]. Ce qui nous amène à penser que le clustering est finalement cet effort d'abstraction qui fera d'un ensemble d'entités similaires une structure singulière. L'intérêt du clustering réside dans le fait de pouvoir élever la visibilité de départ qu'on portait à des entités éparpillées à une visibilité de groupe. En effet, le fait de réunir et de grouper les entités en des clusters permet de réduire la complexité d'un traitement impliquant la gestion de plusieurs éléments. Un tel avantage est fourni par la construction même d'un schéma de clustering prenant en compte *N* entités au départ et fournissant en sortie *K* cluster à manipuler, *K* étant nettement inférieur à *N*. La Figure A-1 représente un résultat final du clustering



(a) Représentation spatiale des éléments de départ



(b) Considération des métriques euclidiennes pour le clustering



(b) Le résultat du clustering

Figure A- 1: Clustering

Il est important de souligner la différence entre clustering (classification non supervisée) et l'analyse basée sur les dissimilitudes (classification supervisée). Dans la classification supervisée nous somme face à un ensemble d'entités prés classifiés par des labels, le problème est d'affecter un label à une nouvelle entité non classifié. Par contre dans le clustering qui est classification non supervisé le problème est de grouper une collection donné d'entités non labellisées dans des clusters significatifs. Dans un sens la labellisation est associée entre autre aux clusters eux même.

Le clustering et présent dans de nombreux domaines à savoir l'analyse de données, le groupage, la prise de décision (desition-marking) et les situations d'apprentissage automatique incluant le datamining, la segmentation d'image, etc.

Afin de classifier les méthodes de clustering plusieurs dimensions peuvent être envisagées. Parmi ces dimensions nous citons :

- Le schéma de clustering généré (une structure hiérarchique de clusters ou une partition unique).
- La construction des clusters (par agglomération ou par division).
- L'appartenance d'un élément à un cluster (exclusive ou flou).
- La prise en compte des éléments à clustériser (incrémental ou non incrémental)

Ces dimensions peuvent nous aider dans notre travail de spécification d'un modèle de clustering qui répond à notre problématique. Il est par ailleurs important de souligner l'existence d'autres dimensions qui soulignent d'autre aspect du clustering. Nous avons retenu uniquement dans notre recherche les éléments caractéristiques d'un grand nombre de méthodes de clustering. Dans ce qui suit, Nous allons décrire les particularités de ces points en soulignant à chaque fois les différences qui existent entre chaque technique. L'ensemble des ces dimensions est résumé dans la **Error! Reference source not found.**.

A.1 Le schéma de clustering : Hiérarchique *vs.* Partitionnement

La distinction entre ces deux approches réside dans le fait que les techniques hiérarchiques produisent une série imbriquée de clusters par contre le partitionnement renvoi une seule configuration non imbriqué. Une classification à été fait dans cette dimension par Jain et Dubes [85]. Nous détaillons cette classification dans les paragraphes qui suivent :

A.1.1 Le clustering hiérarchique

La technique de clustering hiérarchique est illustrée sur la Figure A-2. Cette figure présente sept entités nommées A, B, C, D, E, F et G reparties sur 3 clusters.

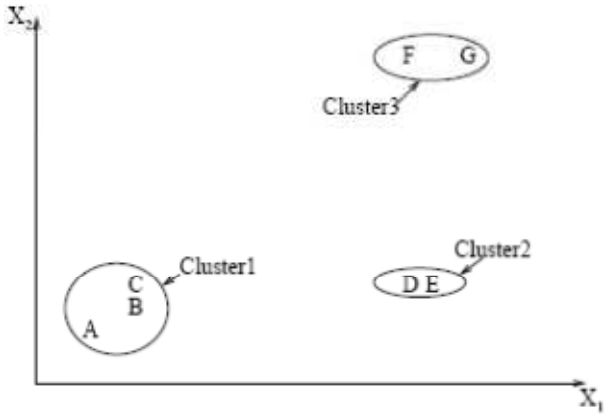


Figure A- 2:Clustering hiérarchique

Un algorithme hiérarchique de clustering va représenter la possibilité d'emboîtement de ces éléments en des structures plus grandes en fonction d'un degré de similarité. Le digramme représenté à la Figure A-3 décrit ce processus d'imbrication pour l'exemple précédant.

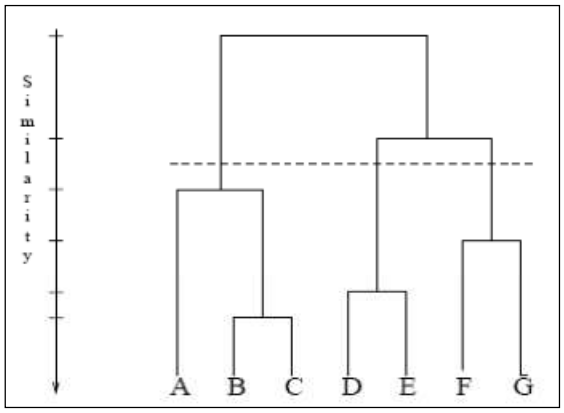


Figure A- 3: Processus d'imbrication hiérarchique

Le but principal des techniques hiérarchiques de clustering est de retrouver une organisation hiérarchique des entités de départ. Deux variantes majeures se retrouve par ailleurs dans ces techniques de clustering hiérarchique à savoir :

- Clustering sur un seul lien (Single link clustering)

Les opérations du single link clustering sont :

  - Affecter chaque élément de départ dans un cluster.

- Assembler les clusters les plus semblables par pairs selon une métrique qui doit se présenter sur au moins un lien entre les éléments (Single-link) des clusters à fusionner.
- Si tous les éléments font partie d'un même cluster global arrêter sinon répéter le traitement.

Le résultat de l'algorithme est une chaîne imbriquée de clusters qui peut être partitionné à tout moment par la précision d'un degré de dissimilitude.

- Clustering sur tous les liens (Complete-link Clustering)

Les opérations de cette technique sont les suivantes :

- Affecter chaque élément de départ dans un cluster.
- Assembler les clusters les plus semblables par pairs selon une métrique qui doit se présenter sur tous les liens entre les éléments (Complete-link) des clusters à fusionner.
- Si tous les éléments font partie d'un même cluster global arrêter sinon répéter le traitement.

A.1.2 Le clustering par partition (Partitional clustering)

Les algorithmes de clustering à partition génèrent une partition unique des éléments de départ contrairement à une structure hiérarchique obtenu par les algorithmes hiérarchiques. Les techniques basées sur le partitionnement sont avantageuses dans le cas ou l'ensemble de départ est conséquent et/ou la représentation hiérarchique est coûteuse en termes de calcul. Par ailleurs le problème rencontré lors d'une technique de clustering basée sur le partitionnement est le nombre de clusters désiré en sortie qui reste non spécifié.

Les techniques de partitionnement produisent généralement des clusters en se basant sur un critère à optimiser localement (au sein d'un même cluster) ou globalement (sur l'ensemble des éléments de départ à clusteriser). Une recherche combinatoire du partitionnement optimal est une démarche très complexe, c'est pour cette raison que les algorithmes établit dans cette voie proposent plutôt un déroulement selon plusieurs configurations initiales ce qui permet après de choisir la meilleur configuration possible en sortie selon ces choix initiaux. Parmi les techniques de clustering basées sur le partitionnement nous citons :

Les algorithmes d'erreur quadratique "Squared error algorithmes"

Les critères les plus utilisés dans les algorithmes de partitionnements sont des critères d'erreur quadratique. L'erreur quadratique d'un clustering C et d'un ensemble d'entités E (contenant K clusters) est :

$$e^2(\mathbf{C}, \mathbf{E}) = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \|\mathbf{x}_i^{(j)} - \mathbf{c}_j\|^2,$$

Où X_i est le $i^{ème}$ élément appartenant au $j^{ème}$ cluster est le c_j est le centroide du $j^{ème}$ cluster.

L'algorithme *k-means* est un des plus utilisé avec cette technique [78]. Il commence par un partitionnement aléatoire des éléments de départs est réaffecte les éléments itérativement aux clusters les plus appropriés en fonction d'un degré de similarité entre chaque élément et le centre représentatif du cluster jusqu'à l'aboutissement à un critère de convergence (diminution importante de l'erreur quadratique).

L'algorithme k-means est populaire grâce sa facilité d'implémentation ainsi que sa complexité réduite (de l'ordre de $O(n)$, n étant le nombre des éléments de départ). Par contre le point sensible de cet algorithme est son choix de la partition initiale à optimiser du fait qu'il peut converger vers un minimum local de la fonction d'erreur quadratique si le choix de cette partition n'est pas approprié.

Le clustering des graphes

- Partitionnement selon un arbre de recouvrement minimal :

Cette technique se fait à l'aide de la construction d'un arbre de recouvrement minimal [86] (MST : Minimal Spanning tree), suivie d'une suppression des branches les plus longues de l'arbre afin de générer des clusters. La Figure A-4 montre un partitionnement de graphe basé sur la construction d'un MST. En brisent le lien qui porte le coût max (dans l'exemple c'est le lien entre C et D avec un coût 6) nous obtenons deux clusters qui sont respectivement :

C1 = {A, B, C}

C2 = {D, E, F, G, H, I}

Le deuxième cluster peut à son tour être divisé en deux, en brisent le lien EF qui porte un coût de 4,5.

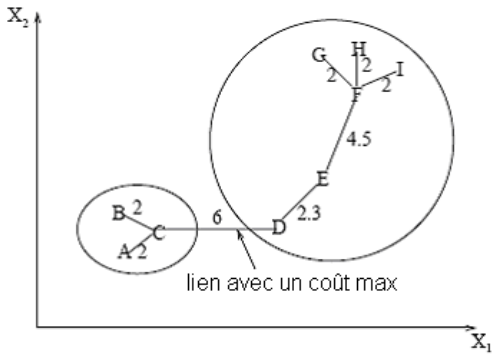


Figure A- 4: Clustering par partitionnement

- Clustering des graphes suivant les modèles de localisation

La théorie de la localisation est une thématique très présente dans la recherche opérationnelle. Elle s'intéresse à la localisation d'un ensemble de sommets suivant des formulations mathématiques. En intégrant la notion de clustering, les sommets à localiser seront les centres respectifs des clusters calculés. Le scénario de base est de localiser un nombre de sommets (exemple : station de service, ambulance, air de repos) dans un graphe complexe.

Nous allons à présent décrire les deux modèles de localisation les plus répondu à savoir le Set Covering [87] et le K-center [88]. Ces modèles sont exposés sous forme de programmes linéaires à optimiser :

- *Le Set Covering*

  C'est un modèle très simple pour repérer les sommets, entre autre les centroides d'un schéma de clustering. L'objectif du set covering est de retrouver un ensemble de sommets de coût minimal qui couvre tous les éléments de départ. La notion de couverture est en fait une proximité bornée par une distance maximale séparent le sommet du cluster de l'élément le plus éloigné. On peut formuler cet aspect suivant le programme linéaire suivant :

$$\text{Minimiser} \quad \sum_{i \in F} x_i$$

$$\text{Sous les contraintes} \quad \forall j : \sum_{i \in F} a_{ij} x_i \geq 1$$

$$\forall i : \quad x_i \in \{0,1\}$$

$$\forall i, j : \quad a_{ij} \in \{0,1\}$$

X_i est une variable décisionnel ayant la valeur 1 si et seulement si le sommet *i* dans l'ensemble *F* des sommets est retenu. La fonction objective minimise le coût total de sélection des sommets. a_{ij} est une matrice de connectivité, un élément de la matrice (ex : l'élément a_{ij}) vaut 1 si et seulement si le sommet *i* couvre le nœud *j*. La distance entre *i* et *j* étant symbolisé par d_{ij}, on se donne une distance de couverture d_c, le sommet *i* couvre l'élément *j* (a_{ij} = 1) si et seulement si $d_{ij} \leq d_c$, sinon a_{ij} = 0. La distance d_{ij} peut être calculé à partir d'une fonction métrique de distance (distance euclidienne, plus court chemin …), par contre la distance d_c est spécifiée comme un paramètre d'entrée.

La première contrainte garantie que le modèle affectera tous les éléments de départ à des sommets, les deux autre indiquent le caractère booléen de X_i et a_{ij}. En assimilant les sommets du set covering au centres des clusters, nous pouvons dire qu'une telle modélisation permettra de calculer à partir d'un graphe et d'une borne métrique, un nombre minimale de cluster (vu qu'on minimise le nombre de centres) avec leurs centres respectifs (Figure A-5). La relation définit dans le cluster est que chaque élément qui en fait partie est à proximité du centre à une distance bornée par la métrique d_c.

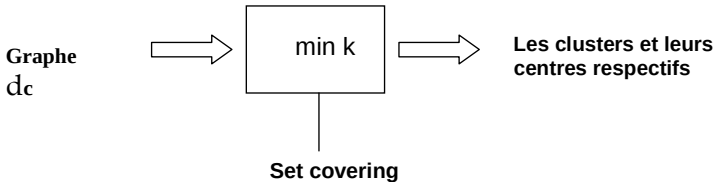


Figure A- 5: L'aspect fonctionnel du set covering

- *Le k-center*

Dans le set covering une distance d_c doit être spécifiée en entré. Résoudre le problème du set covering nous permet d'avoir le nombre de clusters et la position de leurs centres selon une distance de couverture. L'approche K-center vise plutôt à minimiser la distance de couverture d_c avec un nombre maximale k de clusters. Ce modèle minimise la distance maximale entre le chaque élément du cluster et son centre respectif. Nous présentons ci-dessous la formulation linéaire du K-center :

$$\text{Minimiser} \quad d_c$$

$$\text{Sous les contraintes} \quad \forall j \in C: \quad d_c \geq \sum_{i \in F} d_{ij} x_{ij}$$

$$\forall i \in F, j \in C: \quad y_i - x_{ij} \geq 0$$

$$\sum_{i \in F} y_i \leq k$$

$$\forall i \in F: \quad y_i \in \{0,1\}$$

$$\forall i \in F, j \in C: \quad x_{ij} \in \{0,1\}$$

La fonction objective minimise la distance de couverture d_c, la distance maximale entre un élément et son sommet le plus proche. F est un ensemble de sommets, C est l'ensemble de tous les éléments de départ, d_{ij} est la distance entre un élément j et un sommet i. y_i et X_{ij} sont des variables décisionnelles. y_i vaut 1 si et seulement si le sommet i est retenu et X_{ij} vaut 1 si et seulement si l'élément j est couvert par le sommet i. La première contrainte garantie que d_c doit être supérieur à la distance maximale entre un élément du cluster et son centre respectif. La seconde contrainte permet d'éviter qu'un élément j soit affecté à un élément non retenu pour être un centre. La troisième contrainte stipule qu'au maximum on aura k centres (donc k clusters) dans le graphe.

Le modèle k-center peut être vu comme étant la forme dual du set covering. Il prend en entrée le nombre de clusters et fournit en sortie la structure des clusters en assignant les éléments aux centres qui les couvrent au mieux (minimisation de la distance de couverture) alors que le set covering prend en entrée la distance de couverture maximale toléré et essaye de minimiser le nombre de clusters finaux. La Figure A-6 illustre les entrées/sorties du modèle k-center.

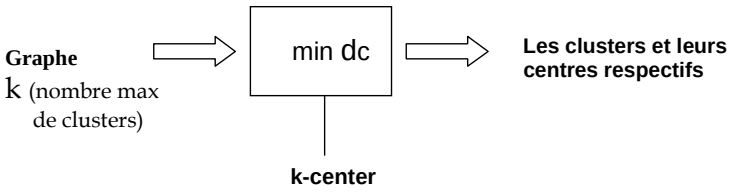


Figure A- 6: L'aspect fonctionnel du k-center

A.2 La construction des clusters : Agglomération *vs.* Division

Cet aspect est un relié à la structure algorithmique de la méthode de clustering. Une approche agglomérative commence par mettre chaque entité dans un singleton et successivement elle va fusionner ces ensembles atomiques jusqu'à la satisfaction d'un critère d'arrêt. Une méthode basées sur des divisions par contre commence par considérer que tous les éléments de base font partie d'un seul cluster, une coupe et effectuée à chaque itération afin de délimiter des clusters plus petit jusqu'à atteindre un critère d'arrêt au delà duquel il n'y aura plus de division.

Les deux figures suivantes décrivent une construction hiérarchique de clusters par les deux approches. La Figure A-7 montre une construction hiérarchique par agglomération, tant disque la Figure A-8 décrit une construction hiérarchique par division.

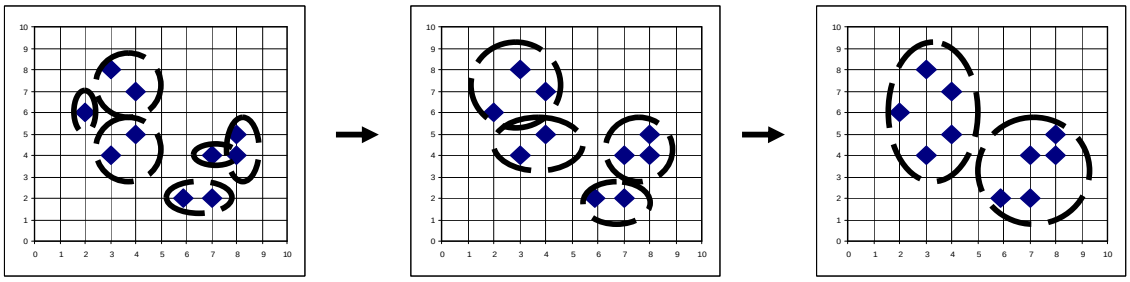


Figure A- 7: Clustering par agglomération

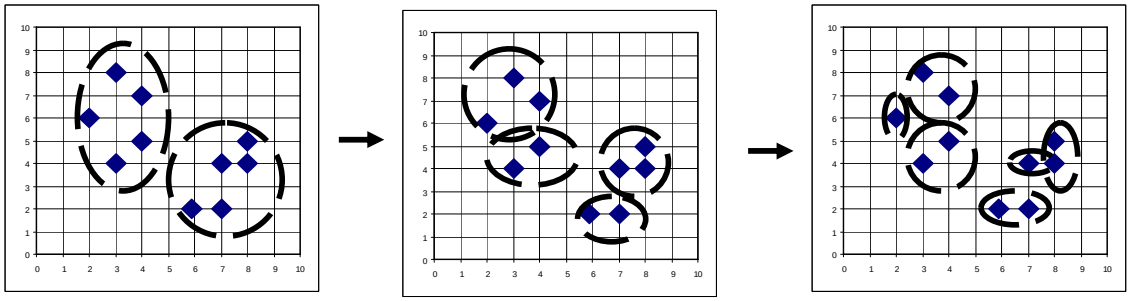


Figure A- 8: Clustering par division

Une variante pour les techniques de clustering par agglomération est l'ajout d'éléments en se basant sur la proximité du voisinage. Cette technique se base la distance minimale au prochain voisin comme critère de sélection de l'élément à rajouter. Une procédure itérative a été propose dans [89] consistant à affecter chaque élément au cluster auquel appartiendrait son voisin le plus proche. Le processus continue jusqu'à ce que tous les éléments de départ sot affectés à des clusters.

A.2.1 L'appartenance d'un élément à un cluster : Exclusive *vs.* Floue

Certaines techniques de clustering peuvent considérer la notion d'appartenance d'un élément comme étant exclusive (le Hard clustering), c'est-à-dire que chaque entité appartient a un seul cluster et ce durant la construction des clusters et à la fin du mécanisme de clustering. Contrairement au Hard clustering, il existe des techniques qui utilisent une autre notion d'appartenance en affectant pour chaque élément un degré d'appartenance à plusieurs clusters au même temps. Ces techniques portent le nom de Fuzzy clustering (Clustering Flou) [90]. La structure des clusters obtenus par ces techniques n'est donc pas rigide comme celle des méthodes exclusives.

A.2.2 Prise en compte des éléments à clusteriser : incrémental *vs.* Non incrémental

Cet aspect se distingue quand l'ensemble de départ à clustériser est important et qu'on est on présence de contraintes concernant le temps d'exécution de l'algorithme de clustering ainsi que les ressources en mémoire disponibles. Lors de l'apparition des techniques de clustering, il n'y a pas eu des cas de figure présentant un grand volume de données à clustériser. Mais avec l'avènement d'application comme le Datamining, le besoin de traiter cet aspect est devenu crucial pour la faisabilité des algorithmes de clustering. Pour palier à ce cas de figure, des techniques particulières ont vu le jour, notamment comme la réduction des entités de départ durant l'exécution d'une façon sélective ou la réduction du volume d'informations concernant les entités de départ à clusteriser. Les approches incrémentales utilisent ces astuces quand au non incrémentales, ils n'en font pas appel.

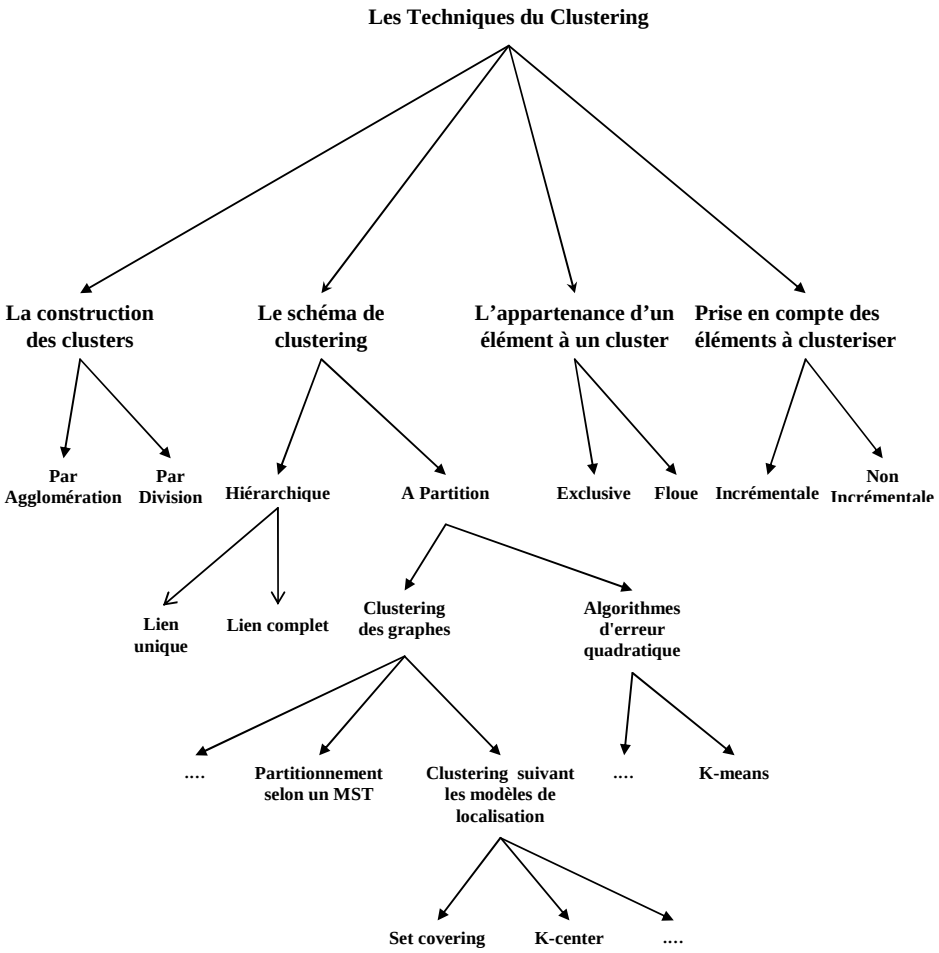


Figure A- 9: La classification des techniques de clustering les plus répondus

Annexe B Les protocoles de communication : vers le modèle de communication de POrg

Dans la continuité de la spécification des différentes dimensions de Pilote Organisationnel, nous sommes arrivés à l'étape de définition de la dimension de communication. Elle définit les règles qui régissent les échanges entre les différentes entités de notre PO. L'étude des messages échangés entres les différentes entités raffine la définition des rôles et la structure de ces entités.

La première question que nous nous posons est : sur quel plan architectural (données, contrôle ou gestion) travaille le PO (relation entre les entités du PO), la réponse à cette question identifie le type d'échange entre les entités du PO.

La troisième question que nous nous posons est : quelle est la nature de l'interaction du PO avec les différents composants de l'architecture ?

Cette partie de l'étude essaie de répondre à ces deux questions en partant de l'existant, c'est-à-dire, en partant des différentes solutions protocolaires proposées actuellement pour faire interagir plusieurs entités coopérantes.

Les flux générés lors d'une communication sont d'abord identifiés par la localisation des entités architecturales émettrices et réceptrice. Ces composants appartiennent à l'un des trois plans : gestion, contrôle ou usager. Avant d'analyser les différents protocoles de communication, il nous faut d'abord définir un filtre à travers lequel nous pouvons analyser ces protocoles. Nous nous proposons, donc, de donner d'abord des définitions claires et précises de ce que sont les flux de gestion, de contrôle et de données (usager) dans la section §2. Nous proposons dans la section §3 une classification des différents protocoles suivant les filtres exposés dans §2.

B.1 Les filtres

Afin d'assurer un service nécessaire à une application donnée, nous devons analyser ce service à travers les trois plans suivants : plan de gestion (§2.1), plan de signalisation (§2.2) et plan de usager (§2.3). Dans ce qui suit, nous proposons des définitions des flux (échanges entre entités communicantes) de chacun des plans selon une spécification à travers cinq (5) filtres principaux, quatre filtres dimensionnels et un filtre de raffinement :

- Fonctionnel : cette dimension représente l'ensemble des traitements mis en œuvre pour lesquels ces flux (protocolaires) sont générés. En d'autre terme, le but pour lequel ces flux sont définis.
- Informationnel : cette dimension renvoie l'image, et ainsi la nature, de l'ensemble des données échangées entre les différents éléments intervenant dans le flux (protocolaire).
- Architectural : cette dimension définit la structure des entités rentrant dans la composition du flux protocolaire.
- et Organisationnel : alors que la dimension architecturale définit la structure des entités rentrant dans la composition du flux protocolaire, la dimension organisationnel spécifie le rôle de chacun de ces éléments.
- Le dernier filtre étant le filtre temporel définit quant à lui, le moment où ce flux intervient courant le cycle de vie du service.
- Temporel (à quel moment ils interviennent),

B.1.1 Flux de Gestion

Les protocoles de gestion sont caractérisés comme suit :

Fonction

Afin d'exercer pleinement leur rôle, les flux de gestion servent à un certain nombre de fonctions. Dans les solutions d'aujourd'hui, ces fonctions génèrent des échanges d'information entre le système gérant et les systèmes gérés.

Le système gérant, peut être soit une ou plusieurs stations de gestion, soit un réseau où ces fonctions sont alors réparties.

Le système géré est soit des composants de réseau : routeurs, serveurs, chemins, bande passante, mémoires tampon, etc., soit des protocoles.

Ces fonctions sont regroupées en cinq (5) aires fonctionnelles (FCAPS) [ref] : gestion des fautes (faults), gestion de la configuration (configuration), gestion de la comptabilité (accounting), gestion des performances (performance), gestion de la sécurité (security) :

Gestion des fautes : elle regroupe l'ensemble des fonctionnalités qui permettent la détection, l'isolation et la correction des anomalies dans le système. Lorsqu'une erreur est détectée, une analyse des informations de l'état du système doit permettre de la localiser et de diagnostiquer sa cause (localisation de proche en proche [réf] ou partant de l'analyser des couches basses et remonter jusqu'à la couche application [réf]). Une analyse curative doit s'en suivre, permettant la reprise du fonctionnement du système.

- *Gestion de la configuration* : on entend par la gestion de la configuration les procédures permettant de contrôler une entité gérée et d'identifier, collecter des données sur ses entités gérées. L'objectif étant de s'assurer au fonctionnement continu du système. Des outils fonctionnels s'imposent alors pour permettre de :
    1. Démarrer, initialise, arrêter le système
    2. Modifier les paramètres du système
    3. Coordonner l'ajout, modification, suppression de nouveau éléments dans le système (équipement, composants logiciels, etc.)
- *Gestion de la comptabilité* : elle a pour but de collecter les informations d'utilisation des ressources (comptage) dans le but non seulement d'élaborer les factures liées à leur utilisation, mais aussi de connaître la répartition de cette utilisation. Elle permet en par la suite :
    - La tarification : qui traduit le coût d'un service suivant la politique tarifaire du fournisseur de ce service.
    - La facturation qui consiste à établir des factures pour les usagers selon leurs spécificités (profile).
- *Gestion des performances* : la gestion des performances a pour but essentiel l'évaluation permanente du comportement du réseau. C'est-à-dire qu'elle doit s'accompagner de fonctionnalités qui permettent de mesurer et de comparer le niveau de performance du

système (évaluations mathématiques, empirique, etc.). Ces fonctionnalités doivent donc permettre :

- Le relevé des données statistiques (taux d'erreurs, délais ; etc.).
- Tenue et la gestion d'un journal de bord d'historiques d'événements.

- *Gestion de la sécurité* : la gestion de la sécurité est l'ensemble des fonctionnalités relatives au support des politiques de sécurité dans un système. Ces fonctionnalités sont essentiellement :
  - La distribution des informations relatives à la sécurité (comme les clés)
  - La compte rendu des événements relatifs à la sécurité (intrusions, tentatives d'accès, etc.)

Information

Les informations transportées par les flux de gestion sont principalement des informations descriptives (image de l'ensemble des composants à priori tous hétérogènes) de représentation structurale (état) et comportementale (performances) du réseau pour avoir une vue plus ou moins homogène (modèle informationnel).

Organisation

L'activité de gestion est réalisée entre des systèmes ouverts par coopération entre un ou plusieurs éléments de gestions, certains agissent en gestionnaires centralisés, d'autres étant gérés à travers des agents. Les rôles de gestionnaire et d'agent sont attribués au début de la communication mais peuvent changer dans le temps :

- Le gestionnaire (manager) est celui qui a la responsabilité d'une ou plusieurs activités de gestion. Il peut envoyer des demandes d'opérations aux agents, il reçoit des réponses concernant les opérations qu'il a demandées ou des notifications. Le gestionnaire est donc l'élément qui pilote un ensemble d'agents dans le cadre de l'activité de gestion.
- L'agent est l'élément qui reçoit des demandes d'opérations et tente de les exécuter ou qui envoie des notifications et des données collectées (polling –question/réponse- et reporting –réponses-). L'agent manipule les informations locales à la ressource gérée.

La liaison entre l'agent et le manager peut être soit directe (1 hop) ou indirecte (multi-hop en traversant plusieurs nœuds ou proxy). L'envoi des informations peut être à la demande du manager où initié par les agents eux même (ex : agent RTCP).

Afin de réduire la complexité de la gestion dans un contexte hétérogène des agents situés au niveau des ressources à surveiller se chargent de collecter les données. Le gestionnaire envoie des pollings pour récupérer ces données. Les agents peuvent être chargés d'envoyer des notifications ou des rapports d'activité (feedback de réception de données par exemple) régulièrement sans attendre de demande du gestionnaire.

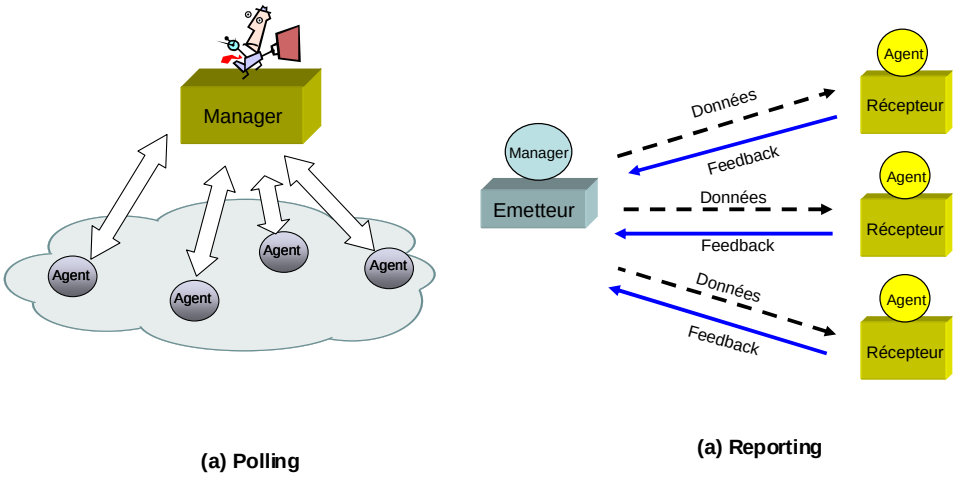


Figure B- 1 : Organisation des éléments de gestion

Dans cette organisation, quasiment tous les traitements sont effectués par le gestionnaire qui doit souvent interagir avec un grand nombre d'agents.

Afin de réduire la charge sur les gestionnaires, la tâche de gestion peut être répartie entre plusieurs gestionnaires de sous-réseau (sous-domaine). Ces gestionnaires sont à leur tour gérés par des gestionnaires niveau supérieur, et ainsi de suite. Des problèmes peuvent alors être résolus au niveau intermédiaire, les informations peuvent être agrégées ou filtrées.

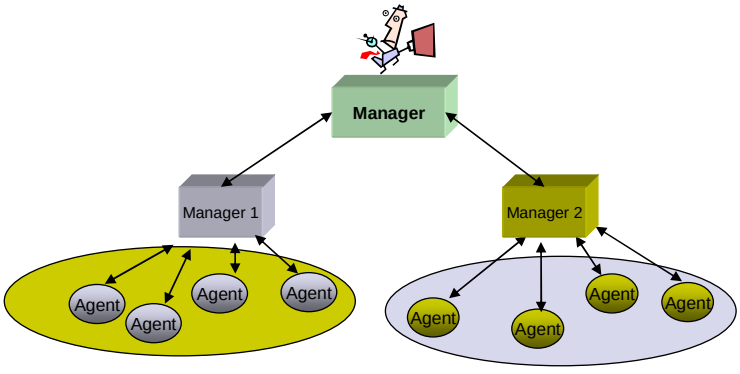


Figure B- 2: Organisation hiérarchisée des éléments de gestion

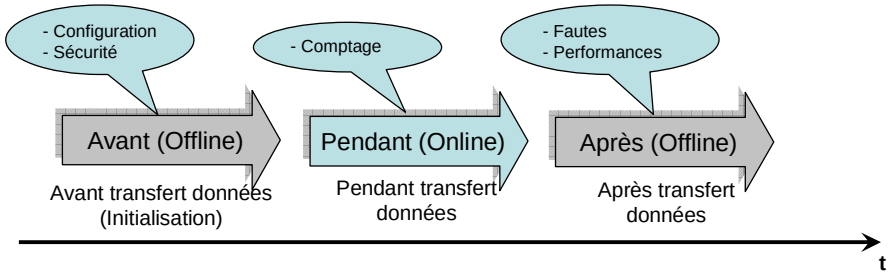
Architecture

La structure des entités échangeant des flux pour des fins de gestion

Rentrant dans Client / Serveur

Temps

Les différentes actions de la gestion peuvent intervenir à différents moments par rapport au cycle de vie du service demandé (flux usager): offline (avant ou après le transfert des données) ou online (pendant le transfert).



- *Offline*

Quand on parle de flux de gestion pure, on parle soit de configuration (réservation statique avant l'exploitation des liens), soit d'actions (analyse et réaction) qui interviennent après le transfert des données. Ces actions auront pour but d'améliorer le rendement global du réseau. Les quatre (4) aires fonctionnelles de la gestion, à savoir, configuration, fautes, performances et sécurité sont généralement des fonctions off-line. C'est-à-dire qu'ils interviennent généralement après ou avant le transfert des données.

- *Online*

La dernière aire fonctionnelle de la gestion, à savoir, la comptabilité (accounting) (comptage des ressources consommées, tickets de consommation) est une fonction qui intervient toujours on-line, c'est-à-dire, en cours de transfert des données.

B.1.2 Flux de signalisation

Les flux de contrôle qu'on peut appeler aussi flux de signalisation regroupent tous les flux utilisés pour établir (provisionner) une liaison entre deux interlocuteurs (entités).

Fonction

L'établissement d'une liaison (connexion, session, etc.) entre deux interlocuteurs se traduit en réalité par une réservation de ressources nécessaires à cette liaison. Les ressources peuvent être de diverses natures : bande passante, entrée dans la table de routage, numéros de ports, temps CPU, mémoire tampon, temps, etc. La réservation de ces ressources dans une configuration donnée est la fonction principale d'un flux de contrôle.

Information

Les informations échangées entre les entités de signalisation (transportées par les flux de signalisation) permettent d'exprimer une demande d'un service donné et de négocier la façon avec laquelle on souhaite avoir ce service. En d'autre terme, les informations transportées par les flux de signalisation sont des information prédictive permettant le provisionnement (provisioning) d'un service donné.

Architecture

Les flux de signalisation peuvent être classifiés en deux catégories : « Path-Coupled » et « Path-Decoupled » :

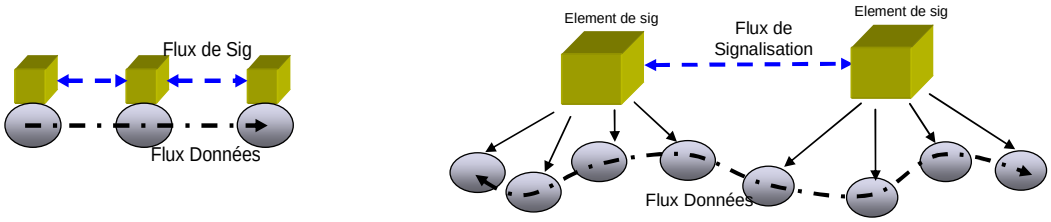
- *Path coupled*

Dans ce mode, le flux de signalisation est acheminé à travers les mêmes nœuds et liens (physiques) que le flux de données pour lequel on réserve des ressources. Dans ce mode, les éléments intervenant dans la chaîne de signalisation sont distribués tout au long du chemin de bout en bout. La liaison entre les deux plans, signalisation et données, se fait à l'intérieur de chaque nœud traversé par ces deux flux (liaison interprocessus).

- *Path-Decoupled*

Dans ce mode, les flux de signalisation sont acheminés à travers des nœuds et des liens (physiques) qui ne participent pas à l'acheminement des données. Ce réseau de signalisation est en relation avec les réseaux d'acheminement de données pour les fonctions de configuration avec un protocole de gestion.

La figure illustre la différence entre les modes path-coupled et path-decoupled.



(a) Signalisation Path-Coupled

(a) Signalisation Path-Decoupled

Figure B- 3: Catégories de signalisations

Organisation

Les éléments intervenant dans le flux de signalisation sont distribués au long du chemin entre les interlocuteurs. Suivant le niveau sur lequel intervient ce flux (réseau, application, etc.) le nombre d'éléments intervenant peut augmenter ou diminuer. Ça peut aller de l'ensemble des éléments de bout en bout (RSVP par exemple) jusqu'à ne contenir que les acteurs d'extrémité (SIP par exemple).

Temps

Les flux de signalisation interviennent toujours avant la transmission des données, en temps réel, au moment du besoin, donc en phase de provioning qui caractérise les flux de signalisation.

B.1.3 Flux Usager

Un flux usager désigne la charge utile généré par l'usager lors de l'utilisation d'un service donné (les paquets voix quand il utilise le service téléphone par exemple).

Fonction

Les flux usagers transportent les données utilisateur ainsi que les informations permettant le traitement de ces données (taille messages, fragmentation, compression, etc.).

Information

Informations liées au traitement des données (protocole et les données elles mêmes (celles de l'user))

Organisation

Les éléments intervenant dans le flux usagers ont pour rôle principale de transmettre les données utilisateurs d'une source à une (des) destination (s). Les éléments intervenant dans cet acheminement dépendent de la nature du service demandé par ces utilisateurs ou ainsi l'influence des acteurs d'extrémité. Suivant chacune de ses configurations, l'organisation des éléments intervenant dans le flux usager change. Nous avons dégagé les organisations suivantes :

- Personne à personne : dans cette configuration, les composants de manipulation des flux usagers peuvent être mises aux points finaux afin d'être le plus prés à l'utilisateur final et ainsi moins de changement dans les autres composants. Des composants comme le correcteur d'erreur (notamment pour la voix et la vidéo) peut être omis tenant compte les personnes aux extrémités sont aptes à corriger les erreurs eux même.
- Personne à serveur (et vis versa) : contrairement à la configuration précédente, comme l'acteur d'extrémité est un serveur (automate), dans ce cas la des composants comme le correcteur d'erreur sont indispensables.
- Personnes à personnes ou Personnes à serveur (et vis versa) : cette configuration nécessite des points de dispatching (forking) de données entre les différents utilisateurs.

Temps

Durant le transfert (users' data) puisque le flux usager, c'est le flux du transfert pour lequel les deux autres flux interviennent pour assurer le partage des ressources.

B.2 Classification des protocoles de communications

Après avoir défini les filtres à travers lequel nous pouvons analyser les différents protocoles de communication. Nous proposons une classification de ces protocoles suivant leurs nature et les fonctions, à savoir, protocoles de gestion, de contrôle et de données. Nous analysons dans cette section chacun des protocoles de communications suivant les quatre (4) filtre présentés ci-dessus : fonctionnel, temporel, organisationnel et informationnel. Cette étude nous permettra de définir les règles qui régissent les échanges entre les différents composants du Pilote Organisationnel.

B.2.1 Protocoles de gestion

Suivant les définitions que nous avons données ci-dessus, nous avons classifié les protocoles : COPS (§3.1.1), Diameter (§3.1.2), SNMP (§3.1.3), LDAP (§3.1.4) et RTCP (§3.1.5) comme protocoles de gestion.

COPS-P

Le protocole de COPS (COPS-P : Common Open Policy Service Protocol) est le protocole utilisé par les composants COPS pour échanger les informations de politiques de décisions. Quant à COPS lui même, il définit les mécanismes et les moyen de prise de décisions d'admission en se basant sur les politiques.

Le protocole de COPS (COPS-P) a été proposé par l'IETF pour renforcer les méthodes d'accès au réseau et étendre les choix décisionnels qui étaient jusque là basés sur les ressources disponibles dans le réseau, aux politiques tels que le type de trafic, identité des utilisateurs et applications, etc.

- Fonction

COPS est un protocole qui permet l'échange des informations décisionnelles entre un serveur de politiques (décisions, choix, etc.) et ses clients. Cet échange peut être à des fins de configuration, de sécurité ou de comptabilité (accounting). Ce protocole offre les moyens basiques d'établir et de maintenir un dialogue entre le client et le serveur et d'identifier les différentes requêtes.

- Information

Les informations décisionnelles échangées entre le client et le serveur sont de la forme de rapports des clients vers le serveur ou des commandes du serveur vers les clients. Nous décrivons dans cette section de manière brève le format des messages ainsi que les types d'objets échangés (Figure 4).

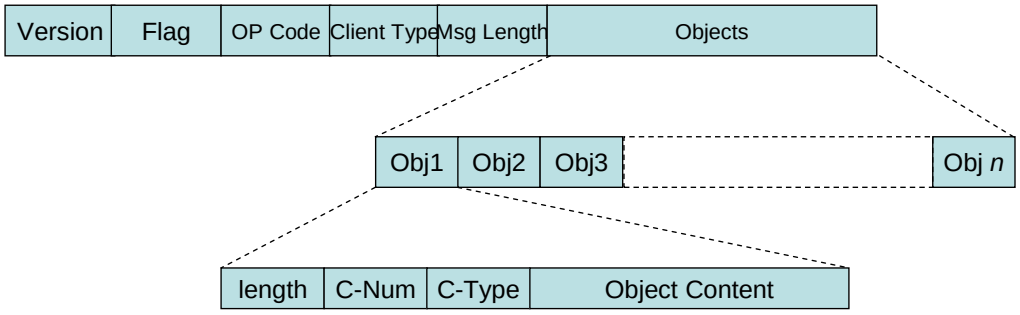


Figure B- 4: Format d'un message COPS

- OP Code : code de l'opération à effectuer (Request, Decision, Report State, Delete Request State, Synchronize State Req, Keep-Alive, etc.)
- Client Type : identifie le client de politique, l'interprétation des objets encapsulés est liée à ce champs (RSVP, DiffServ, IP Highway, HP OpenView Policy Xpert, GGSN – Go interface -, etc.)
- C-Num : identifie la classe d'information contenue dans l'objet.

- Organisation

COPS est un protocole de niveau applicatif basé sur le modèle centralisé. La Figure 3 illustre l'ensemble d'entités entrant dans la composition de COPS. Nous décrivons dans ce qui suit les différents composants de base de COPS :

- PEP (Policy Enforcement Point) : C'est une entité logique qui applique les décisions reçues du serveur (PDP). Le PEP représente l'agent dans le modèle Agent/Manager.
- PDP (Policy Decision Point) : entité logique qui prend les décisions pour un élément donné (PEP). Le PDP représente le manager dans le modèle Agent/Manager. Dans un environnement inter-domaine, on peut avoir plusieurs PDP, chacun gérant un domaine.
- LPDP (Local PDP) : Le PEP peut aussi prendre des décisions en local à travers son « Local PDP". Cependant, le PDP reste l'autorité de décision dans tous les cas. Ceci signifie que les informations de décisions locales doivent être transmises au PDP.
- Base de données de politique: les politiques de décision sont stockées dans une base de données.

- Policy Management Tool : qui représente l'interface entre l'administrateur et COPS, il peut gérer à travers cette interface la base de données de politique (ajout, suppression, mise à jour des politiques) et les PDP.

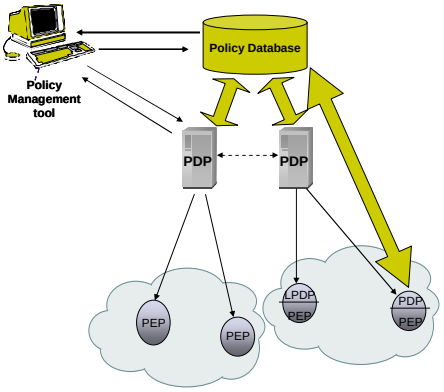


Figure B- 5: Composants COPS

L'organisation proposée par COPS est donc une organisation centralisée où les PDPs envoient les informations/commandes suite à une demande des PEP distribués sur des éléments du réseau.

- Temps

Afin de distinguer différents types de clients, le type du client est identifié dans chaque message. Différents types de client peuvent avoir besoin de différents types de décisions à différents moments. Le contexte temporel de chaque requête correspond au type de l'événement qui l'a déclenché. COPS prévoit deux modèles pour traiter les différents types d'événements à différents moments : Modèle d'externalisation (Outsourcing Model) et modèle d'approvisionnement (Provisioning Model).

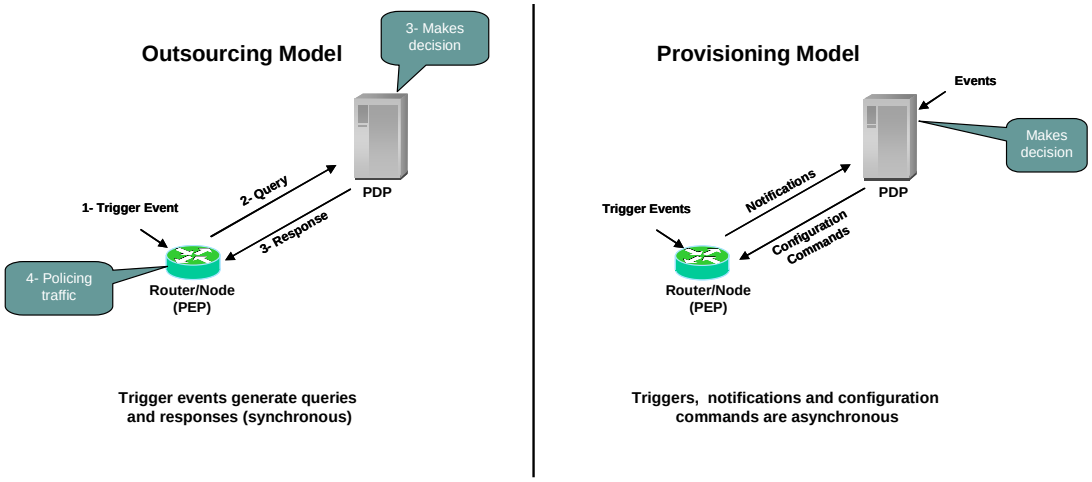


Figure B- 6: Modèles COPS

- *Outsourcing Model (mode offline)*

Dans ce modèle, le PEP envoie une requête au PDP suite à un événement qui nécessite une nouvelle décision de politique avant l'acceptation du flux (nouveau flux, changement de SLA, etc.) (Figure 2). Le PDP prend alors une décision et retourne une réponse au PEP. Toutes ces actions sont exécutées avant la transmission des données.

Le PEP est responsable de la suppression ou la mise à jour de l'état de la requête par le renvoi d'une autre requête pour le même événement.

- *Provisioning Model (mode online)*

Dans ce modèle, le PEP envoie des notifications sur les événements qu'il reçoit. Suivant les notifications et les événements qu'il reçoit, le PDP prend une décision et envoie un message de configuration au PEP.

Quand le PDP ne « souhaite » plus que le PEP utilise une configuration donnée, il envoie un message de décision indiquant la configuration correspondante et la commande de suppression de la configuration. Toutes ces actions sont exécutées en cours de transmission des données (online).

- Fonction

Principalement, Diameter offre deux types de services aux applications : authentification, autorisation d'un côté et comptabilité de l'autre côté.

  - Authentification/Autorisation : ces deux fonctions sont liées. Une requête d'authentification peut transporter elle-même une demande d'autorisation. Ces fonctions sont gérées par session, c'est-à-dire, pour accéder à un service donné (accès réseau, base de donnée, service applicatif, etc.) une session d'autorisation est établie pour permettre l'accès à ce service pour une période donnée.
  - Comptabilité (Accounting) : cette fonction permet la génération de jetons et la comptabilisation de l'accès aux services, et donc, aux ressources disponibles.

- Information

Toutes les informations transportées par ce protocole sont sous la forme d'AVPs (Attribute Value Pairs). Ils contiennent les informations d'authentification, autorisation, de comptabilité, de routage (des messages) et de configuration des éléments du réseau. Certaines des valeurs AVP sont utilisées par le protocole lui-même, tandis que d'autres sont utilisées par les applications utilisant DIAMETER.

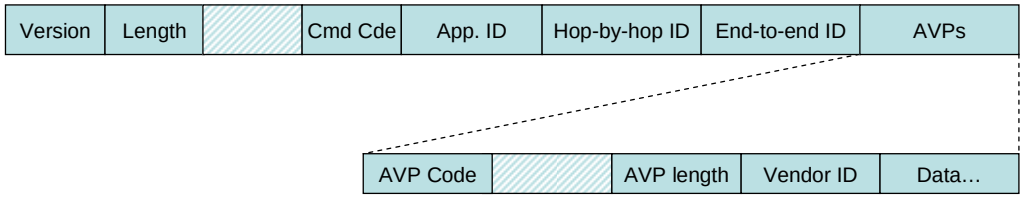


Figure B- 7: Format d'un message DIAMETER

Un message DIAMETER est composé entre autres :

- Cmd Cde représente qui est le code de la commande à exécuter lors de la réception du message (configuration). Les valeurs de ce champ sont prédéfinies par l'IANA (Internet Assigned Numbers Authority).
- App. ID définit le code de l'application qui utilise Diameter comme protocole. Les valeurs de ce champ sont prédéfinies par l'IANA
- Hop-by-hop permet de faire la correspondance entre une requête et sa réponse. Pour les requêtes, ce champ est replacé à chaque passage par un agent. A la réponse, la valeur initiale est remise à chaque passage par un agent.
- End-to-end permet d'identifier les messages dupliqués. La valeur de ce champ, qui est généralement l'identité de l'initiateur de la requête, est inchangée en cours d'acheminement de la source à la destination.
- AVP Code : Ce champ identifie un AVP de manière unique. Ce code traduit tout type d'information pouvant s'échanger entre les composants DIAMETER (commandes, jetons de comptabilité, informations sur les source/destination, erreurs, authentification, etc.).

- Organisation

Dans le protocole Diameter, n'importe quel composant peut initier une session. C'est un protocole basé sur l'échange d'attributs appelés AVP (Attribute Value Parameter) entre les différents composants de la chaîne Diameter. Il repose sur trois (3) éléments principaux : Clients, Serveurs et Agents :

- Client : C'est l'entité qui initie les requêtes AAA. Généralement, le client est un serveur d'accès au réseau (NAS).
- Serveur : C'est l'entité qui répond aux requêtes clients et qui traite l'authentification, l'autorisation et la comptabilité pour l'accès à un domaine donné.
- Agents : Les agents peuvent être déployés à des fins de partage de charge, concentration des requêtes dans un point central ou faire de traitement additionnels sur les messages échangés. Plusieurs agents sont définis pas Diameter suivant l'emplacement dans le domaine (realm) :
    - Agent Relais : Ces agents ont pour rôle d'acheminer les messages entre les composants Diameter. Ils peuvent modifier les messages par l'insertion et la suppression des informations d'acheminement, mais pas les autres informations. Ils sont généralement utilisés comme des concentrateurs de messages pour réduire la charge induite pas la configuration des clients et des serveurs.
    - Agent de Redirection : Ces agents représente une source centralisée pour la résolution des adresses des composants Diameter externe au domaine (il s'agit généralement de trouver l'adresse du serveur).
    - Agent Proxy
    - Agent de Traduction : Ces agents offrent un service de traduction entre Diameter et un autre protocole AAA (Radius par exemple).

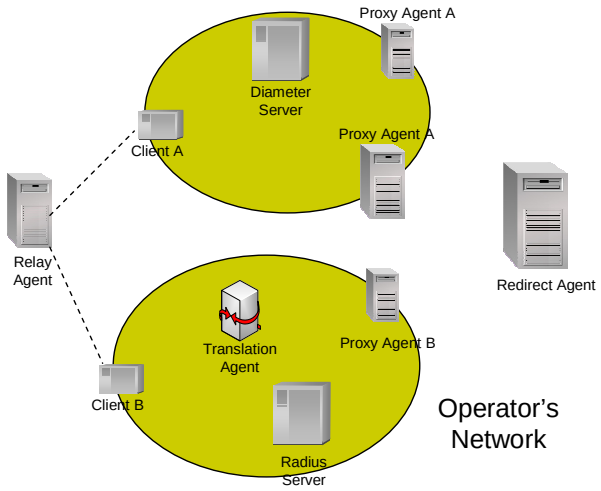


Figure B- 8: Composants DIAMETER

L'organisation proposée par DIAMETER est donc une organisation principalement hiérarchique (§2.1.3), ou des clients envoient des requêtes AAA au serveur. Toutefois, cette requête peut être acheminée à travers plusieurs nœuds DIAMETER pour arriver à destination (serveur).

SNMP

SNMP (Simple Network Management Protocol) est né dans la communauté de la recherche et la défense américaine. Recommandé depuis 1988 comme RFC, SNMP s'est imposé standard « de fait » pour les réseaux TCP/IP.

- Fonction

SNMP est un protocole qui permet l'échange des informations de gestion entre un élément du réseau qui est l'élément géré et une station d'administration. SNMP offre les moyens d'établir et maintenir un dialogue entre la station d'administration et les éléments du réseau pour des fins de configuration (à travers la requête Set), de polling (à travers les requêtes Get et GetNext) et de reporting (à travers la Trap).

- Informations

SNMP est un protocole utilisé pour échanger les différents messages entres les agents et les managers, à savoir, les managers envoient des messages Get, GetNext, Set et Trap (SNMPv1), les agent répondent en envoyant des rapports d'exécution de ces commandes.

Le manager envoie des Get/GetNext pour lire la représentation structurale et comportementale du réseau à travers des paramètres que les agents mettent à jour en temps réel. Les Sets sont envoyés par les managers pour configurer les équipements.

Les agents envoient des Traps quand un événement prédéfini par le manager arrive.

La Figure 6 illustre le format des messages échangés entre le manager et les agents. les informations de gestions sont transportés dans le champs SNMP PDU et plus exactement dans les champs Variable-bindings. Chaque varible-binding contient un identifiant de l'objet (variable) et sa valeur

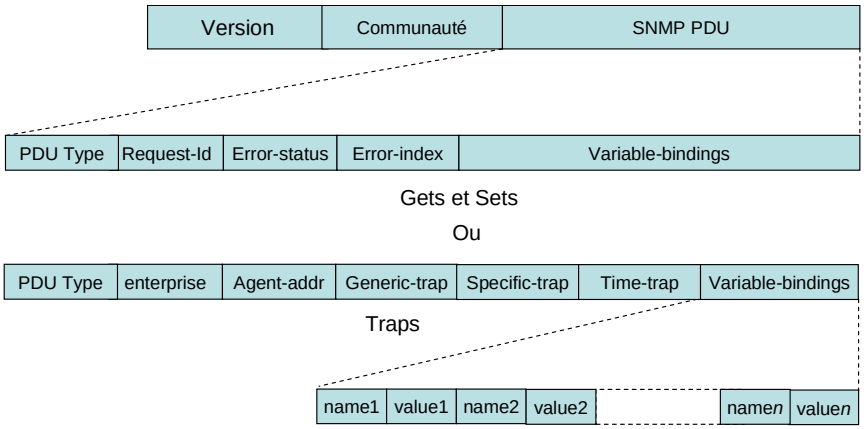


Figure B- 9: Formats des messages SNMP

- Organisation

SNMP repose sur deux éléments principaux: un superviseur et des agents. Le superviseur est la console qui permet à l'administrateur réseau d'exécuter des requêtes de management. Les agents sont des entités qui se trouvent au niveau de chaque interface connectant l'équipement managé au réseau et permettant de récupérer des informations sur différents objets. Ces objets gérables peuvent être des informations matérielles, des paramètres de configuration, des statistiques de performance et autres objets qui sont directement liés au comportement en cours de l'équipement en question. Ces objets sont classés dans une MIB ("Management Information Base"). De manière synthétique, les composants du protocole SNMP sont les suivants :

  - Les équipements gérés (managed devices) sont des éléments du réseau (ponts, hubs, routeurs ou serveurs), contenant des "objets de gestion" (managed

objects) pouvant être des informations sur le matériel, des éléments de configuration ou des informations statistiques ;

- Les agents distribués que les éléments du réseau, c'est-à-dire une application de gestion de réseau résidant dans un périphérique et chargé de transmettre les données locales de gestion du périphérique au format SNMP ;
- Base de donnée (MIB) par élément de réseau (et donc par agent) qui représente le modèle informationnel du réseau et stocke les données collectées par les agents.
- Les systèmes de management de réseau (Network Management System NMS), c'est-à-dire une console au travers de laquelle les administrateurs peuvent réaliser des tâches d'administration.

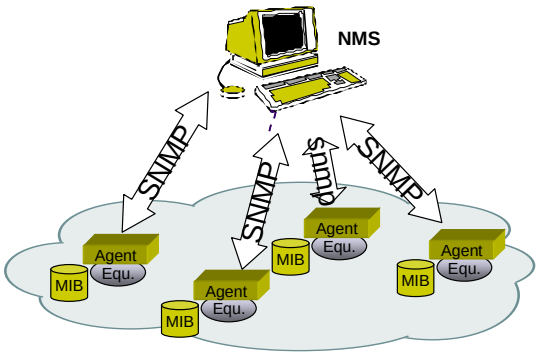


Figure B- 10: Organisation des éléments SNMP

MGCP

MGCP a été proposé par Bellcore et Cisco et valider par la suite par l'IETF pour remplacer, de manière transparente, les réseaux à commutation de circuits par des réseaux à commutation de paquets et ainsi la migration vers la voix sur IP tout en gardant les services (réseau) existants.

- Fonction

Le protocole MGCP (Media Gateway Controller Protocol) est un protocole destiné à contrôler des équipements comme les passerelles et les équipements d'accès.

- Architecture

MGCP est définit comme un système de gestion centralisé (client/Serveur) des ouvertures des sessions (appels) entre les terminaux par un serveur dédié qui pilotera des équipements de (clients). Ces serveurs centraux sont appelés Agents d'Appel (Call Agent - CA) alors que les clients sont appelés Passerelles ou équipement d'accès.

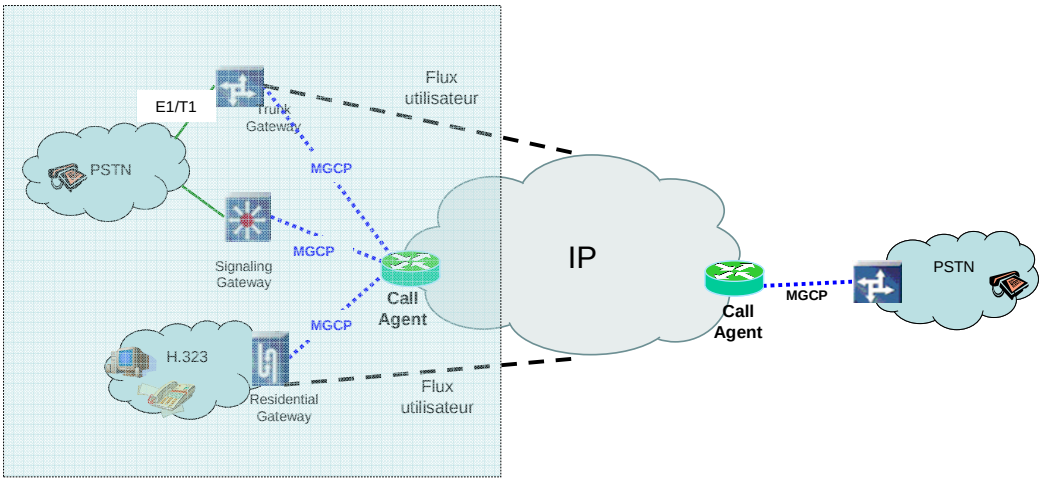


Figure B- 11: Architecture de MGCP

- Organisation

Le protocole MGCP ne concerne que le pilotage des passerelles. Il définit deux rôles : Agent de contrôle et les équipements contrôlés :

- Agent de contrôle (Call Agent): Toute l'intelligence réside dans le Call Agent. Il assure l'établissement, la maintenance, et la terminaison des appels en local (ressources sur les passerelles pour ces appels), et établit les statistiques nécessaires à la facturation. Il assure aussi l'interface entre les différentes signalisations : SIP, H.323, SS7, etc.
- Equipement contrôlé (passerelles) : Elle se limite à des fonctions de conversion de signaux, encapsulation, compression des paquets utilisateurs (données) entre les différents types de réseaux. Plusieurs types de passerelles sont définis suivant les types de réseaux, parmi lesquels :
  - Trunking gateway : fais l'interface entre le réseau téléphonique traditionnel (PSTN) et le réseau IP (VoIP).
  - Voice over ATM gateway : C'est la même passerelle que la Trunking Gateway sauf qu'elle fait l'interface avec un réseau ATM

- Residential gateway : fournit une interface entre les réseaux analogiques traditionnels (téléphone, cable TV, etc.) et le réseau VoIP.
- Business gateway: provide a traditional digital PBX interface or an integrated “soft PBX” interface to a VoIP network.

- Information

Les informations échangées avec le protocole MGCP sont sous forme de commandes en provenance du CA (Call Agent) vers les passerelles et vice-versa. Chaque commande est confirmée par un accusé de réception indiquant l’état de complétion de l’ordre. MGCP utilise huit (8) commandes, dont cinq (5) sont nécessaires pour établir, maintenir ou rompre les connexions.

Commandes	**CA vers GW**	**GW vers CA**	**Code**
Notification Request	X		RQNT
Notify		X	NTFY
Create Connection	X		CRCX
Modify Connection	X		MDCX
Delete Connection	X	X	DLCX
Audit Endpoint	X		AUEP
Audit Connection	X		AUCX
Restart in Porgress		X	RSIP

- NotifyRequest : Cette commande sert à demander à la passerelle de notifier le CA l’occurrence d’un événement particulier (par exemple la réception de tonalité), la prise de ligne (décroché d’un poste), la réception du numéro, etc.
- Notify : en réponse de la Notification Request, les notifications sont envoyées par la passerelle à travers la commande Notify. Les informations envoyées sont principalement des informations relatives aux événements détectés par la passerelle ainsi que l’identification des équipements qui les ont déclenchés.
- CreateConnection : Cette commande utilisée par la CA permet de créer une connexion entre deux points, identifiés parleurs terminaux. La création d’une connexion dans ce contexte veut dire créer un état pour une connexion entre deux points (réservation des ressources locales dans la passerelle pour cette connexion). La description des caractéristiques de l’appel (session) se fait à travers le protocole SDP.

- ModifyConnection : Cette commande utilisée par le CA sert à modifier les paramètres d'une connexion (ressources réservées dans la passerelle).
- DeleteConnection (par le CA): Cette commande utilisée par le pour mettre fin à une connexion et collecter les statistiques d'appel.
- DeleteConnection (par la passerelle) : Dans des certaine circonstances, la passerelle peut être amenée à couper une communication, soit parce que les ressources nécessaires ne sont plus disponibles, soit parce que le destinataire n'est plus à même de la maintenir. Un code indiquant la raison d'interruption de la connexion est envoyé au CA dans cette commande.
- AuditEndPoint : Cette commande utilisée par le CA sert à lancer un audit des équipements d'extrémité.
- AuditConnection : cette commande utilisée par le CA pour obtenir des informations sur une connexion.
- RestartInProgress : cette commande utilisée par la passerelle sert à indiquer que l'un des terminaux, ou un groupe de terminaux, n'est (ne sont) plus en service.

LDAP

Le protocole LDAP, développé en 1993 par l'université du Michigan, avait pour but de supplanter le protocole DAP (servant à accéder au service d'annuaire X.500 de l'OSI), en l'intégrant à la suite TCP/IP. A partir de 1995, LDAP est devenu un annuaire natif (standalone LDAP), afin de ne plus servir uniquement à accéder à des annuaires de type X500. LDAP est ainsi une version allégée du protocole DAP, d'où son nom de Lightweight Directory Access Protocol.

- Fonction

LDAP (Lightweight Directory Access Protocol) est un protocole standard permettant de gérer des annuaires, c'est-à-dire d'accéder à des bases d'informations sur les utilisateurs d'un réseau par l'intermédiaire de protocoles TCP/IP. Le protocole LDAP définit la méthode d'accès aux données sur le serveur au niveau du client, et non la manière de laquelle les informations sont stockées.

Ainsi LDAP fournit à l'utilisateur des méthodes lui permettant de:

- se connecter

- se déconnecter
- rechercher des informations
- comparer des informations
- insérer des entrées
- modifier des entrées
- supprimer des entrées

Voici la liste des principales opérations que LDAP peut effectuer:

Opération	Description
Abandon	Abandonne l'opération précédemment envoyée au serveur
Add	Ajoute une entrée au répertoire
Bind	Initie une nouvelle session sur le serveur LDAP
Compare	Compare les entrées d'un répertoire selon des critères
Delete	Supprime une entrée d'un répertoire
Extended	Effectue des opérations étendues
Rename	Modifie le nom d'une entrée
Search	Recherche des entrées d'un répertoire
Unbind	Termine une session sur le serveur LDAP

- Organisation

Le modèle adopté par LDAP est le modèle client/serveur, où les clients envoient des requêtes de recherche, inscription, modification, etc. d'enregistrement (objets) dans un répertoire. Le serveur traite ces requêtes et envoie une réponse aux clients.

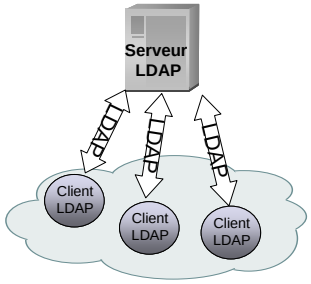


Figure B- 12: Architecture LDAP

RTCP

- Fonction

RTCP est fondé sur la transmission périodique de paquets de statistiques à tous les participant d'une conférence (reporting). La mission principale de RTCP le « feedback » de la réception des paquets RTP par un destinataire. Il fournit un retour d'information sur la qualité de réception des données transmises dans les paquets RTP. Cette information en temps réel peut être immédiatement exploitée par la source pour adapter le type de codage au niveau de ressources disponibles.

- Information

Plusieurs types de paquets sont définis par RTCP de manière à englober une grande variété de rapports. Deux types principaux de rapport sont définis par RTCP : rapport émetteur et rapport récepteur. La différence entre ces deux types de rapports est que le rapport émetteur (SR) est émis si la station a effectivement envoyé des paquets depuis le dernier rapport ; sinon un rapport récepteur (RR) est émis.

- Le rapport émetteur (SR)

Ce message est divisé en trois parties (Figure 12): En-tête, Informations émetteur (SSRC émetteur) et le rapports de réception qui contient les rapport de réception depuis le dernier rapport. Chaque bloc du rapport de réception comporte les statistiques d'une seule source de synchronisation.

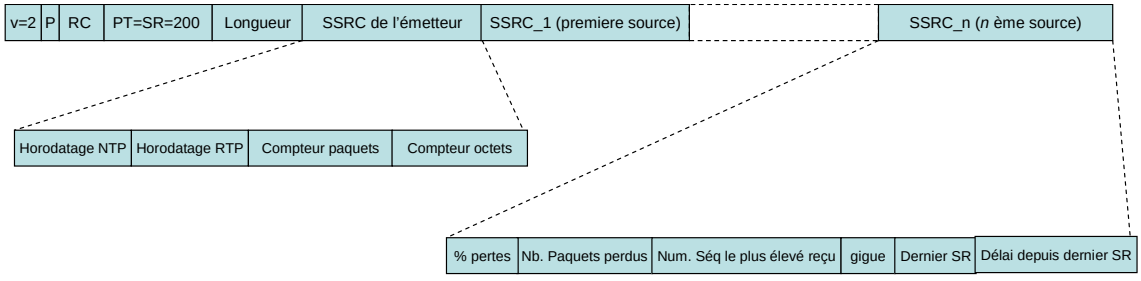


Figure B- 13: Format du message SR (RTCP)

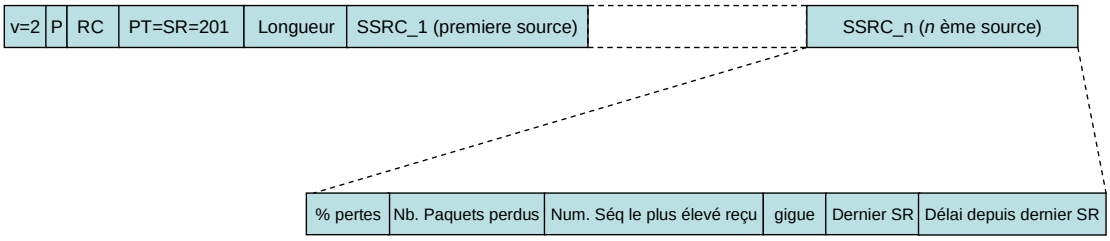


Figure B- 14: Format du message RR (RTCP)

RTSP

RTSP (Real Time Streaming Protocol), développé au sein de l'IETF, est fondé sur les standards du Web. Il définit les procédures à suivre pour le pilotage à distance des serveurs de média (streaming media) audio et vidéo. Il est devenu proposition de standard sous le RFC 2326 en avril 1998.

- Fonction

RTSP sert à établir et à contrôler la diffusion, à la demande, des flux de média continus (Streaming). Le contrôle des média se fait à travers des commandes de type magnétoscope : marche, arrêt, avance rapide, pause, etc.

- Information

Les informations échangées avec le protocole RTSP sont sous forme de commandes en provenance du client de streaming vers le serveur et vice-versa Les messages véhiculés par RTSP sont de deux types : des requêtes et des réponses :

- Requêtes : elle transporte principalement les commandes (méthodes dans le jargon RTSP) du client vers le serveur. Le tableau suivant illustre les différentes commandes employées par RTSP :

Commandes	Client vers Serveur	Serveur vers Client
DESCRIBE	X	
ANNOUNCE	X	X
OPTIONS	X	X
PAUSE	X	
PLAY	X	
RECORD	X	
REDIRECT		X
SETUP	X	

SET_PARAMETER	X	X
GET_PARAMETER	X	X
TEARDOWN	X	

- DESCRIBE : cette commande permet au client de demander la description d'un flux de médias proposée par un serveur, ou simplement d'un objet multimédia.
- ANNOUNCE (émise par le client) : cette annonce, émise par le serveur permet d'envoyer une description d'un flux au serveur.
- ANNOUNCE (émise par le serveur) : permet de changer la description d'un flux en temps réel, lors par exemple de l'introduction d'un nouveau média.
- SETUP : permet de définir le mécanisme de transport adopté par le flux de média (exemple : RTP).
- PLAY : permet de positionner le point de départ de la diffusion à un moment quelconque du flux.
- REDIRECT : cette commande est émise par le serveur pour indiquer à un client qu'il doit se connecter à un autre serveur pour recevoir le flux demandé. L'URL du nouveau serveur est présente dans cette requête.

o Réponses : ces messages sont transmis par le serveur en réponse à la commande du client. Elle contient un rapport d'exécution de ces commandes. Le tableau suivant illustre les différentes catégories de réponses du serveur :

Code	**Type**	**Description**
1xx	Informatif	Requête acceptée, traitement en cours
2xx	Succès	L'action demandée a bien été reçu, comprise et acceptée
3xx	Redirection	Une action préalable doit être entreprise afin de pouvoir répondre à la requête
4xx	Erreur Client	La requête présente une erreur de syntaxe, ou ne peut être acceptée
5xx	Erreur Serveur	La requête est valide, mais une erreur due au serveur l'empêche d'y accéder

- Temps

La caractéristique de contrôle d'une entité à distance, notamment à travers la commande PAUSE, donne aux flux RTSP une définition temporelle particulière par rapport aux autres flux de gestion. Le flux RTSP intervient avant, après, et courant l'envoi des données.

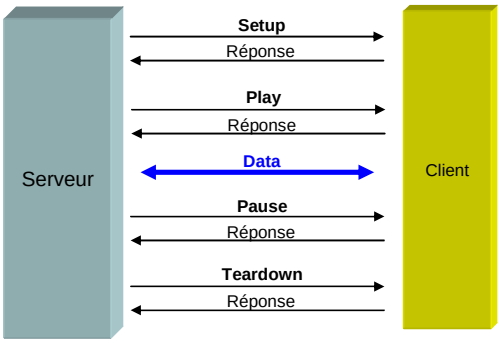


Figure B- 15: Messages échangés entre le Client et le Serveur

B.2.2 Protocoles de control

SIP

- Fonction

SIP est un protocole de signalisation de niveau applicatif qui a pour rôle d'établir, de modifier et de terminer des sessions applicatives sur un réseau IP. Ceci implique la réservation des ressources applicatives nécessaires à l'établissement des sessions. Un participant peut être invité, et un nouveau flux peut être ajouté (supprimé) à (d') une session déjà ouverte. Ce protocole offre les moyens pour gérer des sessions (transférer et arrêter des sessions, modifier les paramètres d'une session, etc.) et gérer les participants à une session : invitation, exclusion, etc.

- Organisation

Dans ce qui suit, une description de chaque élément SIP et son rôle dans le processus global d'ouverture de sessions SIP (Figure 9). Les éléments entrant dans l'établissement d'une session SIP peuvent être classifiés en deux catégories : agents utilisateur (terminaux) et serveurs intermédiaires.

- Agents utilisateur (User-Agents-UA) : Ces agents représentent les équipements des utilisateurs finaux. Une session SIP est établie entre ces agents. C'est eux qui initient (ou répondent à) (une) session.
- Serveurs Intermédiaires (Serveurs) : Ces intermédiaires sont des entités logiques à travers lesquelles transitent les messages SIP pour arriver à la destination. Ils sont utilisés pour acheminer et rediriger ces messages. Ces serveurs incluent :
  - Registar Server (RS) est une base de données qui contient les informations de localisation des UAs dans un domaine.
  - Proxy Server (PS) fait le relais entre le UA et le RS. Il accepte les requêtes émises par l'UA et cherche l'adresse du destinataire en le demandant au RS. Il achemine cette requête directement vers le destinataire final s'il est dans le même domaine, ou vers le Proxy d'un autre domaine en demandant son adresse au Redirect Server.
  - Redirect Server permet aux Serveurs Proxy (PS) de rediriger les requêtes vers des destinataires se trouvant dans des domaines différents.

La réservation des ressources applicative se fait principalement dans les terminaux.

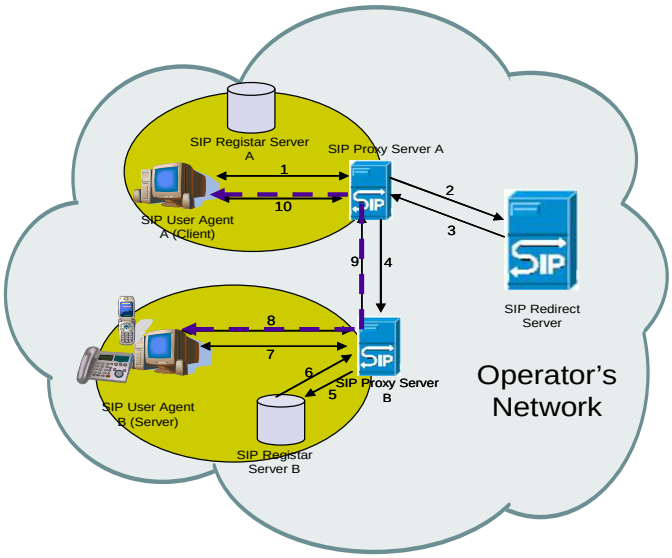


Figure B- 16: Composants SIP

- Temps

La Figure 12 illustre la procédure d'ouverture de sessions SIP. Nous avons sélectionné un exemple où une session de streaming est établie entre deux utilisateurs. RTP/RTCP étant utilisé comme protocole de transport.

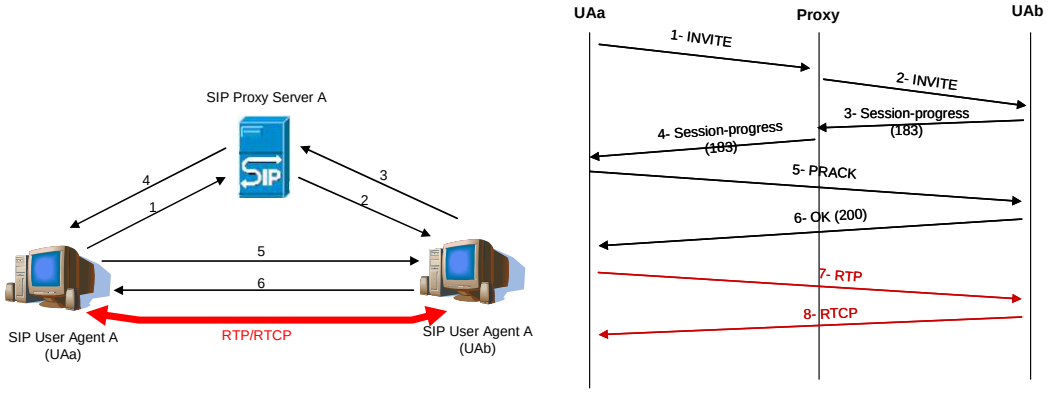


Figure B- 17: Procédure d'établissement d'une session SIP

Les messages d'initialisation (provisionning) sont les messages SIP échangés pour l'établissement de la session. En cours de cette initialisation (réservation de ressources applicatives), une négociation des type d'application, paramètres de la session (temps, etc.), et capacité des deux interlocuteurs (terminaux) est effectué. (Messages 1 à 5). C'est à la fin de cette procédure de réservation de ressources applicatives que commence la transmission des données utilisateurs (plan usager).

- Information

Comme mentionné précédemment, SIP offre un service de provisionning pour des sessions. Les messages SIP sont divisés en trois parties (Figure 13) : Ligne de démarrage (Start Line), en-tête (header) et données (body).

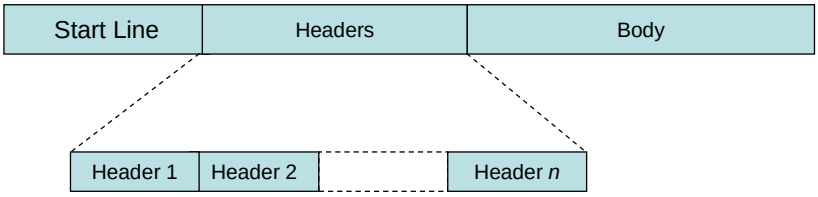


Figure B- 18: Structure d'un message SIP

RSVP

RSVP est un protocole de signalisation mis en œuvre

- Fonction

La demande de réservation de bande passante émane du destinataire à travers le message Path

Le routeur A détermine qu'il a besoin d'établir un LSP vers C. Les paramètres requis pour la session (ou les politiques administratives pour le réseau) permettent au LSR A de déterminer que la route pour le nouveau LSP doit traverser le LSR B. le LSR A génère un message PATH avec une route explicite (B, C) et les détails des paramètres du trafic pour le nouveau LSP. Le LSR A réserve les ressources qu'il a besoin pour le nouveau LSP avant de transmettre le message PATH au LSR B comme un datagramme IP.

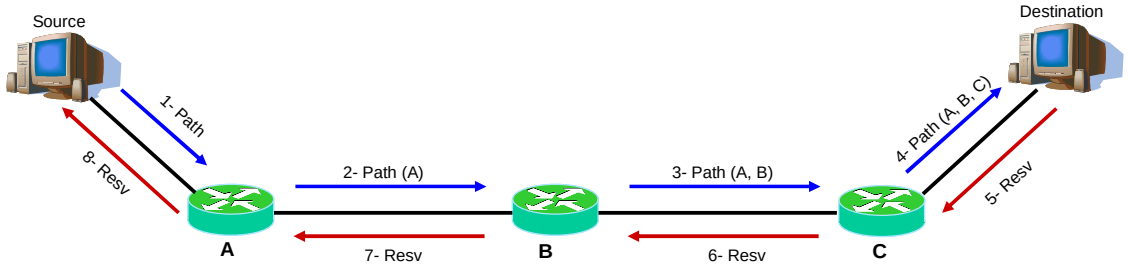


Figure B- 19: Messages RSVP

- Le LSR B reçoit le message PATH, détermine que il n'est pas le egress LSR pour cet LSP, modifie la route explicite dans le message et le transmet au LSR C.
- Le LSR C détermine qu'il est le egress LSR, détermine quelle bande passante nécessaire à réserver. Il alloue les ressources et sélectionne un label pour le nouvel LSP. Il transmet ce label à le LSR B dans le message RESV.
- Le LSR B reçoit le RESV et fait la correspondance avec le premier message (PATH) en utilisant le « LSP ID » contenu dans les messages PATH et RESV. Il détermine la quantité de ressources qu'il faut réserver au nouvel LSP, il sélectionne un label, met à jour la « Forwarding Table », et passe le nouveau label au LSR A dans le message RESV.
- Le traitement dans le LSR A est similaire, sauf qu'il ne réserve pas un label et le transmet au LSR en amont.

- Information

Les informations transportées par les messages RSVP sont des informations sur les besoins prédictifs des flux en termes de ressources réseau.

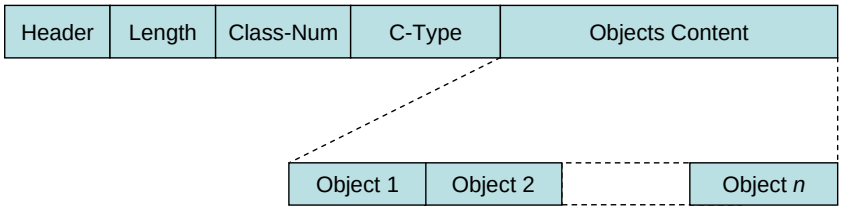


Figure B- 20: Structure d'un message RSVP

- Header : qui contient des informations sur la paquet RSVP transporté : version, type de message, checksum et taille.
- Class-Num : représente le numéro de la classe de l'objet
- Contenu de objets : représente le charge utile des messages RSVP.

- Architecture

On pourra diviser la procédure de réservation de ressources dans les routeurs en deux parties (Figure 16) : la première partie se charge de traiter les messages de réservation et d'accepter où de refuser les demandes (contrôler l'accès au réseau). La seconde partie met en œuvre cette réservation et met les paquets sur le lien de transmission.

Contrôleurs d'accès : la réception d'une demande de réservation de ressources met en action deux mécanismes : contrôle d'admission (Admission Control) et contrôle d'autorisation (Policy Control).

- Contrôle d'admission : reçoit les messages de demandes de réservation de ressources, comprenant des critères de débit ou de taux de pertes à respecter. Ce module vérifie que le nœud en dispose de suffisamment de ressources pour accepter la réservation. Dans l'affirmative, le demande de réservation (message Path et Resv), sinon un message d'erreur est envoyé (ResvErr).
- Contrôle d'autorisation : il vérifie si le demandeur a l'autorisation d'effectuer une réservation. Ce niveau de contrôle authentifie l'appelant, et permet éventuellement de procéder à la facturation des ressources réservées.

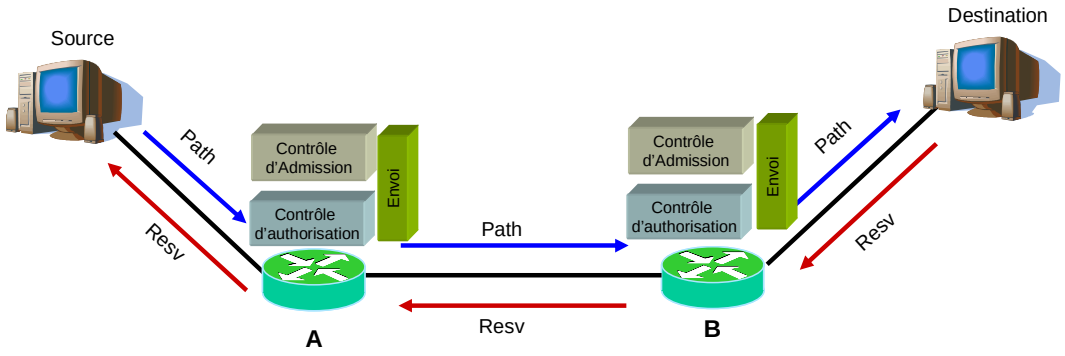


Figure B- 21: Architecture de RSVP

B.2.3 Protocoles de données

RTP

Le protocole de transport RTP représente une évolution of VAT (Visual Audio Tool) qui est une application d'audioconférence développée par le LBNL (Lawrence Berkeley National Laboratory). RTP se distingue de son ancêtre VAT par le fait que VAT est conçu spécifiquement pour les flux audio, alors que RTP support les différents types de flux : audio, vidéo, données. RTP a été approuvé par l'IETF en 1996 comme standard.

- Fonction

Le but de RTP et de fournir un moyen de transmettre de bout en bout sur IP des données soumises à des contraintes de temps réel (audio, vidéo, ...). Il est conçu pour une exploitation dans un environnement multipoint. Le rôle principal de RTP consiste à mettre en oeuvre des numéros de séquence de paquets IP pour reconstituer les informations de voix ou vidéo même si le réseau sous-jacent change l'ordre des paquets.

Plus généralement, RTP permet :

- d'identifier le type de l'information transportée,
- de reconstituer la base de temps des flux en ajoutant des marqueurs temporels et des numéros de séquence des paquets transportés.
- de contrôler l'arrivée à destination des paquets et détecter les pertes.

- Information

v=2	P	X	CC	M	PT	Seq. Num	SSRC	CSRC	Data …

Figure B- 22: Structure d'un message RTP

- CC : Nombre de CSRC. Ce nombre indique le nombre d'identificateurs de sources contributrices contenus dans la liste CSRC.
- PT : Type de contenu. Ce champ identifie le type de contenu audio et vidéo transporté par ce paquet, et son format de codage.
- Seq. Num (Sequence Number) représente le numéro d'ordre de sortie des paquets. Il est utilisé par le récepteur pour réordonner et détecter les pertes.
- Horodatage (Timestamp) : le paquet RTP est horodaté à l'instant même de l'échantillonnage du premier du premier octet le constituant.
- SSRC : il identifie la source de synchronisation, c'est-à-dire l'émetteur référence de la base de temps.
- Liste CSRC : de 0 à 15 éléments chacun. La liste CSRC nomme toutes les sources ayant contribué aux données contenues dans le paquet s'il a été créé par un mixeur. Le nombre total d'identificateurs est indiqué dans le champ CC.

- Architecture

Les média audio et vidéo sont transmis de manière indépendante. Il n'y pas de couplage direct au niveau du protocole RTP entre les sessions vidéo et audio sauf si le flux multimédia était codé ainsi (format qui mélange audio et vidéo). C'est grâce à l'identificateur de source et à l'horodatage des échantillons multimédia dans les paquets RTP que les flux sont rapprochés et synchronisés.

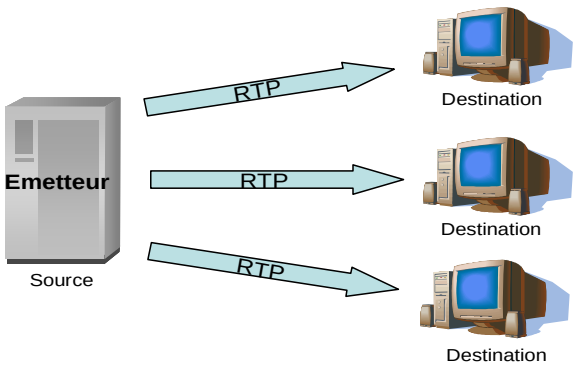


Figure B- 23: Architecture du protocole RTP

SCTP

SCTP (Stream Control Transmission Protocol) a été standardisé par le groupe SIGTRAN de l'IETF par le RFC2960. Il représente une amélioration et un renforcement de TCP de point de

vue détection des pannes (coupures) de connexions et les informations associées pour supporter le transport de données de nature temps réel (voix, vidéo, et données).

- Fonction

En plus de la fiabilité de la transmission, et le contrôle de flux comme TCP, SCTP :

- Fournit le moyen d'envoyer plusieurs flux de données entre deux extrémités de manière indépendante. Pour chaque flux les données sont transmises dans l'ordre et sans perte. Avec cette méthode, les pertes dans une donnée d'un flux n'affectent pas les autres flux. TCP, par contre, n'offre qu'un seul flux par connexion, où la perte d'un paquet entraîne un retard dans l'envoi des paquets suivants. Cet inconvénient de TCP est appelé « blocage de tête-de-ligne : Head-of-Line Blocking »
- Est un protocole orienté messages; c'est-à-dire que SCTP maintient la structure du message applicatif (appelé Chunk). TCP est un protocole orienté octets (bytes), c'est-à-dire qu'il ne préserve pas l'unité des données transmises nécessitant un calcule et regroupement en messages les paquets reçus par les niveaux supérieurs (application).
- Comprend et utilise la notion de stations « multi-homées ». Une station multihomée est une station qui possède plusieurs adresses IP. A l'initialisation des connexions SCTP, les peers échangent une liste de leurs adresses IP. De cette façon, un message qui nécessite une retransmission sera envoyé vers une adresse IP alternative afin d'augmenter la « survivabilité » d'une session SCTP en présence de pannes dans le réseau. TCP n'utilise qu'une seule interface IP par connexion, si une panne survient le peer ne sera plus joignable.

- Information

Les informations échangées entre deux points SCTP sont les données utilisateurs en plus des informations permettant la structuration et l'identification des différents flux de données. Comme tout protocole de données, un message SCTP commence par l'en-tête incluant le port source, port destination, une étiquette pour identifier l'association source destination, et un checksum.

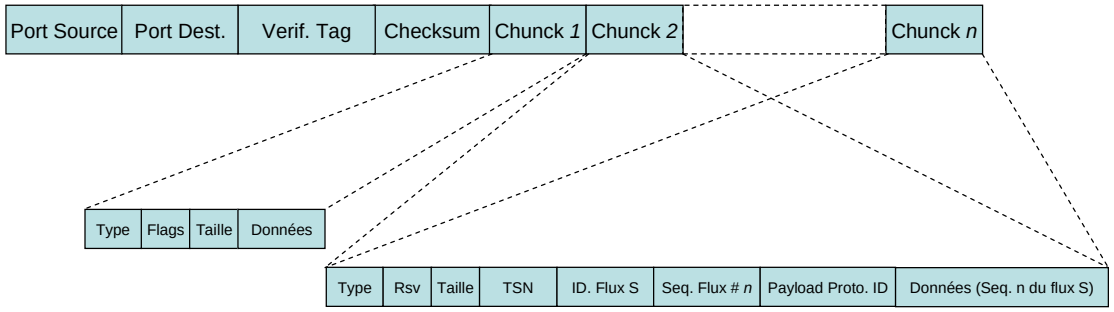


Figure B- 24: Structure d'un message SCTP

Les données utilisateurs sont transmises sous formes de messages (chunks). Un chunk transporte sois des informations sois des données utilisateur. Le nombre de chunks est limité par le MTU. On peut avoir plusieurs chunks dans le même paquet SCTP sauf pour l'initialisation (message INIT) et l'acquittement de l'initialisation (INIT ACK). Chaque chunk est composés d'un champ type de chunk, étiquette, taille du chunk et les données transportées (s'il s'agit de données utilisateur). Quatorze (14) types sont actuellement définis :

Code	Nom	Description
0	*Données*	Le chunk transporte des données utilisateur
1	*INIT*	Envoyé afin d'initialiser une association SCTP. Le réception d'acquittement à ce message ouvre l'association
2	*INIT ACK*	Acquittement du message INIT
3	*SACK*	Acquittement des données
4	*COOKIE ECHO*	Envoyé à la fin de l'initialisation pour annoncer le début d'envoi des données
5	*COOKIE ACK*	Acquittement du message COOKIE ECHO
6	*HEARTBEART*	Utilisé pour tester la connectivité (applicative)
7	*HEARTBEART ACK*	Acquittement du message HEARTBEART
8	*ABORT*	Demande de fermeture de l'association. La raison de la demande est transportée dans ce message (fermeture non prévue)
9	*ERROR*	Informer le peer d'une erreur survenue. Le type d'erreur est transporté dans ce message
10	*SHUTDOWN*	Demande de fermeture de l'association (lors de la fin d'envoi des données)
11	*SHUTDOWN ACK*	Acquittement de SHUTDOWN
12	*SHUTDOWN COMPLETE*	Conclue la fermeture de l'association. Envoyé en réponse à SHUTDOWN ACK