Université de Montréal

Gestion coopérative de la qualité de service dans les applications multimédias : Spécification et Simulation

Par
Loubna MEKOUAR

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la faculté des études supérieures en vue de l'obtention du grade de Maître ès Science (M.Sc) en informatique

Octobre 1998

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé :

Gestion coopérative de la qualité de service dans les applications multimédias : Spécification et Simulation

Présenté par :

Loubna MEKOUAR

a été évalué par un jury composé des personnes suivantes :

Rachida Dssouli	Présidente du jury
Gregor V. Bochmann	Directeur de recherche
Jan Gecsei	Membre du jury

Mémoire accepté le : 07 12. 1998

Sommaire

Pendant les dernières années, plusieurs applications multimédias ont été développées. Comme exemple de ces applications multimédias, nous pouvons citer : la téléconférence, la télédiffusion haute définition, la vidéo à la demande, les jeux interactifs et la coopération à distance. Ces applications manipulent une grande variété de médias comme le texte, l'image (fixe ou animée) et la voix. Elles diffèrent des applications classiques traditionnelles (messagerie, transfert de fichiers) par le fait qu'elles nécessitent : une grande capacité de transfert, une conservation des contraintes temporelles entre les informations, une diffusion multipoint des données et une garantie de service.

La qualité de service est devenue actuellement très importante dans les applications multimédias. Elle représente un ensemble de caractéristiques qualitatives et quantitatives d'un système multimédia distribué nécessaires pour réaliser le fonctionnement demandé d'une application. Ce fonctionnement représente la présentation des données à l'usager et la satisfaction de ce dernier. Plusieurs applications multimédias sont maintenues actuellement sur l'Internet. Ce dernier offre un service du type meilleur effort et donc, aucune garantie de service n'est fournie. Cette situation se contredit avec les besoins des applications multimédias qui nécessitent un temps de réponse court, une synchronisation des données et une réservation des ressources au niveau du réseau et des stations de travail. Le bon déroulement de ces applications nécessite une gestion efficace de la qualité de service.

Une architecture coopérative de la gestion de la qualité de service a été développée à l'Université de Montréal par Hafid et al.. Elle consiste à installer des agents au niveau des sources d'informations, des commutateurs du réseau et les différents récepteurs. Dans le cadre de notre projet de maîtrise, nous avons spécifié le protocole de gestion coopérative de la

qualité de service en utilisant le langage de description formelle SDL (Specification and Description Language). L'utilisation de ce langage permet d'éliminer les différentes erreurs introduites par l'utilisation d'un langage informel. Nous avons également amélioré les différents algorithmes proposés afin d'aboutir à de bons résultats lors de la simulation du protocole puisque la version originale souffrait de plusieurs anomalies.

Étant donné que la notion du coût est importante dans les applications multimédias, nous avons introduit la notion du coût dans les algorithmes proposés dans [Haf-97b] et nous avons modifié les algorithmes afin que le prix soit considéré. Nous avons modélisé le comportement de l'usager pour choisir une qualité donnée parmi l'ensemble des qualités disponibles. Nous avons développé également des stratégies de persuasion. Ces stratégies consistent à persuader des usagers à transiter vers une qualité de service particulière. La persuasion permet de libérer des ressources, d'augmenter le profit du système et de satisfaire les usagers. Nous avons développé également des stratégies pour calculer les prix des qualités de service après avoir réalisé une persuasion et pour exécuter d'autres opérations de persuasion.

Mots clés : application multimédia, qualité de service, gestion coopérative de la qualité de service, négociation, renégociation, adaptation de la qualité de service, agent, langage de spécification et de description SDL, persuasion.

Table de matières

Liste des acronymes

ATM	Asynchronous Transfer Mode
CD	Compact Disk
CPU	Central Processing Unit
CqosM	Cooperative Qos Management
DPE	Distributed Processing Environment
DQDB	Distributed Queue Dual Bus
EFSM	Extended Finite State Machine
Estelle	Extended State Transition Language
FDDI	Fibber Distributed Data Interface
HDTV	High Definition TeleVision
IP	Internet Protocol
ISO	International Standardization Organization
ITU	International Telecommunication Union
JPEG	Joint Photographic Experts Group
LOTOS	Language of Temporal Ordering Specification
Mbps	Mega bit per second
MM	Multimédia
MPEG	Motion Picture Experts Group
NETBLT	Network Block Transfer
ODP	Open Distributed Processing
OMT	Object Modeling Techniques
OSI	Open Systems Interconnection

Pid	Process Identifier
Qds	Qualité de service
RM-ODP	Reference Model for Open Distributed Processing
RNIS	Réseau Numérique à Intégration de services
RSVP	Resource reSerVation Protocol
RTP	Real Time Protocol
SDL	Specification and Description Language
SDT	Specification and Description Tool
ST-II	Stream Protocol
TCP	Transmission Control Protocol
TM	Tree_Manager
UDP	User Datagram Protocol
XRM	Extended integrated Reference Model
XTP	eXpress Transport Protocol

Liste des figures

Liste des tables

Remerciement

Louanges à Dieu, le tout puissant, le miséricordieux, pour son aide la plus précieuse. Je Lui présente mes remerciements les plus sincères et les plus profonds.

Je présente mes remerciements à mon directeur de recherche, le professeur *Gregor Bochmann*, pour sa supervision et ses directives tout au long de ma maîtrise et auprès de qui j'ai énormément appris.

Un grand merci pour le ministère d'éducation du Québec, mon conseiller pédagogique *Ugo Mercier Gouin* et tout le personnel du ministère qui ont mis à ma disposition tous les moyens nécessaires pour que je puisse réaliser ce travail.

Je remercie aussi tous les membres du laboratoire téléinformatique et tout le personnel du Département d'Informatique et de Recherche Opérationnelle.

Mes plus profondes gratitudes sont destinées à mon père, *Driss Mekouar*, à ma mère, *Houriya Kabbaj*, ainsi que mes chères sœurs et mon cher frère pour leurs encouragements et leur soutien inoubliables.

J'aimerais exprimer également ma grande reconnaissance à mon mari *Youssef Iraqi* pour ses conseils précieux, sa motivation et ses encouragements.

Chapitre 1.
Introduction

Les récents développements dans les technologies de traitement de l'information ont permis le développement des applications multimédias (MM). Comme exemple de ces applications MM, nous pouvons citer : la téléconférence, le télé-magasinage, l'enseignement à distance, la télédiffusion haute définition, la vidéo à la demande, les jeux interactifs et la coopération à distance. Toutes ces applications manipulent une grande variété de médias comme le texte, l'image et la voix et nécessitent des traitements complexes et variés : des systèmes d'archivage d'images fixes associés à des bases de données, des serveurs vidéo haute performance, des transmissions haut débit et des moyens d'accès multiple et à distance aux banques d'informations. Au fur et à mesure que les stations de travail gagnent en puissance et en possibilités, la demande pour ces nouvelles applications croît rapidement pour s'étendre des milieux professionnels aux particuliers. Les applications MM se caractérisent par la taille volumineuse et l'aspect temporel des médias et nécessitent donc beaucoup de ressources au niveau du réseau et des stations de travail. En plus, ces applications MM distribuées sont destinées à fonctionner sur divers types de réseaux qui relient des stations de travail possédants des plates-formes différentes et doivent par conséquent être en mesure de satisfaire les besoins des usagers.

La qualité de service (Qds) est devenue actuellement un point très important à traiter dans les applications multimédias. Elle est décrite en terme d'un ensemble de caractéristiques perceptibles par l'utilisateur et permet à une application donnée de spécifier les conditions précises de la transmission d'un flot d'information. Comme exemple de ces paramètres : le débit minimal qui doit être respecté ou le taux de pertes acceptable. Actuellement, plusieurs applications MM sont offertes sur l'Internet comme : Visual Audio Tool (vat), Network Video Tool (nv), Network Voice Terminal (nevot), etc. Mais, aucune garantie de service n'est fournie. Ceci signifie qu'un paquet émis par une source peut être perdu, sinon le temps nécessaire pour atteindre la destination peut être long. Ces situations ne sont pas convenables

aux applications MM (téléphonie ou télé-chirurgie par exemple) qui nécessitent un temps de réponse court, une synchronisation entre les flots reçus et une garantie de service.

La gestion centralisée de la qualité de service utilisée dans certaines applications MM (comme l'accès aux serveurs de données) pose certaines restrictions sur le nombre de sources et de destinations utilisées. Cette gestion a plusieurs inconvénients comme la surcharge du réseau, les goulots d'étranglement, etc. En plus, elle ne permet pas de réaliser la négociation entre chaque source et chaque participant dans le cas des applications multimédias de grande taille. Ces applications utilisent un nombre très grand de participants comme une application de télédiffusion haute définition (HDTV) ou une application de téléconférence. Une application MM doit aussi présenter à l'usager plusieurs qualités de service pour qu'il puisse choisir selon ses préférences, son budget et les capacités de sa machine. Donc, il est nécessaire de présenter à l'usager plusieurs qualités et d'assurer une gestion distribuée de la qualité de service. Cette gestion permet de préciser les valeurs appropriées des paramètres de la Qds pour les différents composants du système, de réserver les ressources nécessaires du système et de maintenir le service offert aux usagers. Parmi les fonctionnalités de la gestion de la Qds, nous pouvons citer : la négociation, la renégociation et l'adaptation de la Qds. La négociation consiste à présenter à l'usager, au début d'une session de travail, un ensemble de qualités de service et les prix correspondants afin qu'il fasse son choix. La renégociation de la qualité de service permet à l'usager de changer la qualité reçue pendant la session pour une autre qualité meilleure ou faible. Alors que le rôle de l'adaptation est de maintenir la qualité accordée à l'usager.

Plusieurs architectures ont été développées afin de gérer la qualité de service d'une manière efficace et de remédier aux problèmes présentés ci-dessus. Une architecture multi-agent a été développée par Hafid et al., à l'Université de Montréal, afin d'assurer une gestion coopérative de la Qds (CQoSM) [Haf-97b]. Elle consiste à installer un agent de Qds dans chaque émetteur, chaque commutateur du réseau et chaque récepteur participant à une application donnée. L'agent commutateur est considéré comme un gestionnaire de l'arbre de diffusion multipoint (Tree_Manager). Ces agents de qualité de service coopèrent entre eux afin de gérer la Qds d'une manière efficace. Les différentes fonctionnalités de la gestion de la Qds

comme la négociation, la renégociation et l'adaptation de la Qds sont effectuées d'une manière distribuée à travers le réseau. Par conséquent, le nombre d'usagers peut augmenter sans poser aucun problème pour le bon déroulement de l'application. En plus, la gestion coopérative de la qualité de service est indépendante des protocoles du réseau et peut être utilisée avec différentes techniques de codage et de routage.

Dans une application MM, la notion du coût est importante. Le prix à payer par un usager diffère selon la qualité reçue. En fait, plusieurs paramètres entrent en jeu pour déterminer le prix correspondant à une qualité donnée. Parmi ces paramètres, nous pouvons citer : le type du service, le type de garantie du service et la durée du service. La garantie du service est préférable dans une application MM. Elle peut être fournie si les ressources du réseau sont disponibles comme la bande passante, les mémoires tampons et les cycles du CPU. Le système doit posséder une méthode d'optimisation des ressources utilisées pour les gérer d'une manière efficace. Pour cette raison, les politiques de persuasion sont nécessaires. La persuasion dans une application MM consiste à persuader certains usagers à transiter vers une autre qualité de service. Par exemple, nous considérons une application de téléconférence où plusieurs participants utilisent des qualités de service différentes. Le système essaie de trouver la qualité qui permet d'aboutir à un profit maximum. S'il trouve cette qualité, il essaie de convaincre les participants qui reçoivent une autre qualité de recevoir la qualité de persuasion en payant un prix réduit par rapport au prix régulier. Cette technique permet au système de libérer les ressources réservées aux autres qualités de service et de profiter de la réservation des ressources effectuée pour la qualité de persuasion. Les politiques de persuasion peuvent être utilisées dans les contextes suivants :

- L'Internet n'offre actuellement aucune garantie du service. Le type de garantie fourni est meilleur effort. Pour remédier à ce problème, l'utilisation du protocole RSVP (Ressource reSerVation Protocol) va permettre une garantie de service. Les applications MM pourront offrir plusieurs Qds avec des prix différents et qui varient selon la quantité des ressources réservées. Dans ce cas, la persuasion peut être appliquée.

- Un réseau qui utilise la technologie ATM où les services MM se divisent en quatre familles : les trafics à débit constant ou continu (CBR, Constant Bit Rate), les trafics à débit variable (VBR, Variable Bit Rate), les trafics à débit disponible (ABR, Available Bit Rate) et les trafics à débit non spécifié. La gestion du trafic CBR consiste à allouer la bande passante correspondante au débit crête de la source pendant toute la durée de la communication. Par contre, le trafic VBR alterne des périodes d'activité pendant lesquelles les données sont générées à un débit très élevé avec des périodes de silence où très peu de données sont générées. La durée de chacune de ces périodes est aléatoire ainsi que le débit de la période d'activité. Dans une application MM, plusieurs Qds peuvent être offertes selon la classe du service utilisée et avec un prix différent. La persuasion peut donc être utilisée (voir chapitre 5).

La persuasion a plusieurs avantages comme l'optimisation des ressources utilisées, la satisfaction des usagers puisqu'ils reçoivent une qualité de service avec un prix réduit et l'augmentation du profit du système grâce au partage des ressources ou à leur libération.

Notre projet de maîtrise a été réalisé dans le cadre du projet de l'Institut Canadien de Recherche en Télécommunications. Dans notre projet, nous avons amélioré le protocole de gestion coopérative de la Qds, développé à l'Université de Montréal par Hafid et al. La version originale souffrait de plusieurs anomalies. Nous avons spécifié ce protocole en utilisant le langage de spécification formelle SDL. Cette spécification permet de décrire formellement le protocole afin d'éliminer toute ambiguïté et confusion, de le simuler et de le valider. Nous avons aussi introduit la notion du prix dans les algorithmes proposés vue l'importance de cette notion dans les applications MM. Nous avons également élaboré des stratégies de persuasion afin de permettre une optimisation des ressources, une augmentation du profit du système et une satisfaction des usagers. Enfin, nous avons développé des stratégies de calcul du prix des Qds après persuasion et les conditions d'arrêt d'une persuasion déjà effectuée.

Ce rapport est constitué d'un ensemble de chapitres. Il est organisé de la manière suivante :

Tout d'abord, le chapitre suivant présente les applications MM distribuées d'une manière détaillée. Une définition de la qualité de service est aussi présentée, ainsi que les exigences des applications MM en matière des systèmes de communication, des stations de travail et de la gestion de la qualité de service. Finalement, les principales architectures de Qds proposées dans la littérature sont décrites d'une manière détaillée.

Le chapitre 3 présente les différentes phases du cycle de développement d'un logiciel. Pour spécifier le protocole de gestion de la Qds, nous avons utilisé le langage de description formelle SDL. Une présentation détaillée sur ce langage est donnée dans ce chapitre.

Le chapitre 4 traite la gestion coopérative de la qualité de service dans les applications MM. Les différentes fonctionnalités de la gestion de la Qds sont étudiées et une description des différentes opérations réalisées par le protocole est présentée.

Le chapitre 5 présente l'importance de la notion du coût dans les applications MM. Le principe du mécanisme de persuasion est également traité. Les différentes politiques de persuasion élaborées sont décrites d'une manière approfondie.

Le chapitre 6 présente une modélisation par objet du protocole de gestion de la qualité de service, ainsi qu'une description des classes et des associations utilisées. Ensuite, la spécification du protocole en SDL est décrite. Cette spécification formelle est précise, claire et non ambiguë et permet de simuler et valider le protocole.

Le chapitre 7 présente les résultats des simulations que nous avons réalisées. En utilisant des diagrammes d'ordonnancement des messages (MSC), les résultats de la simulation du protocole de gestion de la Qds sont présentés. Nous présentons également les résultats de la simulation des politiques de persuasion, ainsi qu'une caractérisation de ces résultats.

Finalement, une conclusion permet de conclure le travail réalisé et de suggérer des directions futures de recherche.

Chapitre 2.
Applications multimédias distribuées et Qualité de service

Dans ce chapitre, nous présentons une définition des applications multimédias distribuées. Ces applications qui sont à la fois multimédias et distribuées nécessitent de nouvelles exigences par rapport aux applications traditionnelles classiques. La haute performance du réseau et des stations de travail est nécessaire afin de réaliser les fonctionnalités attendues des applications multimédias. Une définition de la notion de la qualité de service sera aussi introduite afin de traiter la gestion de la qualité de service dans les applications multimédias d'une manière approfondie. Par la suite, une étude détaillée des différentes architectures de la qualité de service sera présentée.

2-1 Définition des applications multimédias

Les applications multimédias permettent de manipuler une large gamme de media [Vog-95, Haf-96b, Boc-97, Haf-98]. Le terme multimédia (*MM*) réfère à la combinaison de plusieurs monomédias tels que le texte, l'image, le son et la vidéo. Le media se décompose en deux catégories :

- *Le media classique* : tel que les données informatiques ordinaires comme le texte et le graphique.
- *Le media continu* : tel que la vidéo, l'audio et l'animation. Le media continu est constitué d'une séquence continue d'un nombre fini de trames dépendantes et le terme *flot* permet d'indiquer cette séquence. Le media continu se caractérise par la dépendance temporelle qui existe non seulement entre les composants d'un flot (intra-f lot), mais aussi entre les différents flots d'un media ou entre des médias différents (inter-flot). La dépendance entre l'audio et la vidéo d'un film donné est un exemple de la dépendance temporelle. Les applications MM diffèrent des applications classiques par le fait qu'elles nécessitent un grand débit de transmission et une synchronisation entre les différents médias. La synchronisation permet de contrôler que le rythme de transmission d'un ou plusieurs flots de données est respecté. Elle permet également de gérer la notion d'ordre dans une transmission depuis ou vers un groupe d'utilisateurs. Elle peut utiliser plusieurs

mécanismes tels que la gestion d'une horloge réseau ou le contrôle de la gigue (jitter).

Comme exemple de ces applications, nous pouvons citer:

- La vidéo-sur-demande : cette application est généralement asymétrique et unidirectionnelle. Les données sont transmises depuis un serveur vidéo et sont compressées et codées dans un format vidéo. Elles sont reçues et traitées du côté utilisateur par un terminal. Ce traitement consiste à regrouper les flux, à les décoder et à les convertir en données numériques afin de les afficher sur un écran.
- La téléconférence : cette application permet une conférence entre plusieurs utilisateurs. Les informations véhiculées peuvent être de la voix, des images animées, du son, des images fixes et des documents. Cette application doit assurer une transmission temps réel de l'ensemble des médias vers différents usagers éloignés.
- Les nouvelles-sur-demande, l'enseignement à distance, la coopération à distance, etc.

Les applications multimédias peuvent être décomposées en deux types d'applications :

i. *Les applications de présentation* :

Ces applications permettent l'accès à distance aux documents MM stockés dans des dispositifs de grande capacité. Les usagers peuvent accéder à ces informations à partir des serveurs MM. Ensuite, ces informations sont transmises à travers le réseau et reçues par les usagers. Dans ce genre d'applications, la difficulté réside dans la grande vitesse de transmission des données et la conservation des dépendances temporelles du media continu. D'abord, la largeur de bande du disque impose des limitations sur le nombre de flots à transmettre simultanément et donc, le nombre d'usagers servis en même temps est limité. Ensuite, les techniques de gestionnaires de fichiers traditionnels n'assurent pas une garantie du temps-réel pour la livraison du media continu. Un exemple de ces applications de présentation est la vidéo-sur-demande.

ii. *Les applications de conversation :*

Ces applications nécessitent une communication en temps réel. L'information est créé, en temps réel, par une source (microphone et camera vidéo) et doit être transmise rapidement aux différents usagers. Les données produites sont valides pour une certaine période. Une fois ce temps écoulé, ces données ne sont plus utiles. Les applications de conversation sont plus

sensibles en terme du délai par rapport aux applications de présentation. La téléphonie et le travail coopératif assisté par ordinateur (Computer-suported cooperative Work CSCW) sont des exemples d'applications de conversation.

Les applications multimédias sont, dans la majorité des cas, des applications de présentation et de conversation en même temps. Le service de collaboration à distance combinant vidéoconférence et tableau blanc partagé (fenêtre commune visible et modifiable par les usagers) ainsi que la télé-chirurgie sont des exemples de ce type d'application [Dab-96].

L'utilisation de ces applications du media continu crée de nouvelles exigences pour transmettre, traiter et garantir une qualité de service de ces informations. Ces nouvelles exigences sont les suivantes :

a) Grande capacité de transmission

Vue la grande quantité d'information à transmettre, le système de communication doit offrir une grande capacité de transmission des données. Les techniques de compression de la vidéo, par exemple : JPEG et MPEG, peuvent accroître l'utilisation effective de la largeur de bande. Comme exemple, une vidéo non compressée qui nécessite 472 Mbps peut une fois compressée nécessiter uniquement 1,2 Mbps. La possession de techniques de compression plus performantes permettra de réduire d'avantage la quantité d'information à transmettre ;

b) *Les contraintes temporelles*

Le media continu se caractérise par la dépendance temporelle. Cette dépendance signifie qu'il y a une relation temporelle entre les données et que certaines informations doivent être délivrées à la destination en même temps [Sal-96]. Par conséquent, les applications MM doivent conserver cette dépendance pour chaque flot (synchronisation intra-flot) et entre les différents flots (synchronisation inter-flot) pendant une session donnée. Elles doivent assurer également un délai borné pour la transmission des données. Un retard de transmission va entraîner la perte des données puisqu'elles ne seront plus utiles. En plus, une synchronisation entre les participants d'une session donnée pour une application de téléconférence est nécessaire pour que tous les participants de cette application reçoivent les mêmes informations presque en même temps.

c) *La garantie du service*

Les applications MM doivent assurer le maintien du service. Le réseau et les stations de travail doivent être en mesure de maintenir ce service en conservant la vitesse demandée et la dépendance temporelle. S'ils n'arrivent pas à le faire, une dégradation de la qualité de service (*Qds*) doit être détectée. Le type de garantie de service dépend des mécanismes d'allocation de ressources utilisés. Le type de garantie peut être déterministe, statistique ou meilleur effort. Le type de garantie est déterministe s'il est applicable pour chaque unité de données transmise entre deux points d'accès de service pour la durée de service. Comme exemple, exiger que chaque image vidéo ne dépasse pas un délai de 100 ms. Si le type de garantie est statistique, une garantie de service n'est pas appliquée sur chaque unité de données transmise. Par exemple, exiger que 90% des images vidéo ne dépassent pas un délai de 100 ms. Si le type de garantie est meilleur effort, cela signifie qu'aucune garantie n'est assurée pour les services offerts. Les garanties statistiques sont plus appropriées aux applications MM qui tolèrent la perte de certaines informations par rapport aux applications classiques sans que la qualité de présentation soit inacceptable.

d) Le multipoint

La transmission multipoint permet de transmettre les mêmes données à plusieurs points dispersés géographiquement à partir d'une seule source. Chaque groupe multipoint est défini par son identificateur qui représente son adresse. Si une source désire envoyer des données aux membres d'un groupe, elle va les envoyer à l'adresse multipoint de ce groupe. Le flot reçu par le groupe est envoyé alors à tous les membres de ce groupe. Les usagers qui désirent participer aux activités d'un groupe doivent devenir des membres de ce groupe. Dans cette méthode de transmission, la source n'est pas obligée de connaître les adresses des différents usagers et il n'y a pas une connexion par usager. Cette technique est intéressante puisqu'il n'est pas nécessaire pour une application de télédiffusion, par exemple, de connaître tous les téléspectateurs avant de commencer la diffusion. Les destinataires n'ont pas besoin eux aussi de connaître l'adresse de la source pour pouvoir recevoir de l'information. Par conséquent, le processus de transmission est indépendant de la taille du groupe et ne nécessite pas l'établissement de connexions entre les participants. En plus, cette technique permet de réduire le traitement, le stockage et la largeur de bande utilisés pour la transmission de l'information.

2-2 Définition de la qualité de service

La notion de la qualité de service est apparue pour décrire certaines caractéristiques de transmission des données. Par exemple, le modèle de référence de l'OSI [ISO-84] a un certain nombre de paramètres de Qds qui décrivent les caractéristiques de la transmission tels que :

- *Le temps d'établissement de la connexion* est la durée qui s'écoule entre le moment où une demande de connexion est émise par l'utilisateur et la réception de la confirmation. Plus ce temps est court, meilleure est la qualité de service.

- *La probabilité d'échec d'établissement* mesure la chance qu'une connexion ne puisse s'établir dans un délai maximum défini.

- *Le débit de la liaison (throughput)* mesure le nombre d'octets utiles qui peuvent être transférés en une seconde.

- *Le taux d'erreur résiduel* est le rapport entre le nombre de messages perdus ou mal transmis et le nombre total de messages transmis au cours d'une période considérée.

La couche Transport du modèle OSI de l'ISO représente le garant de la qualité du service fourni par la couche Réseau. Si la qualité du service de la couche Réseau est faible, la couche Transport réalise la jonction entre ce que désire l'utilisateur, en terme de qualité et ce que la couche Réseau met à sa disposition. Le transport OSI distingue pour les paramètres de la qualité du service trois types de valeurs : préféré, acceptable et inacceptable qui sont choisis lors de l'établissement d'une connexion.

Les nouvelles applications MM nécessitent l'introduction de nouveaux services et la définition d'une nouvelle politique de mise en œuvre de la Qds pour que tous les composants du système participent et assurent une garantie de la performance. Parmi ces services : la diffusion multipoint et la synchronisation. Ainsi, la notion de la Qds telle qu'elle est définie dans le modèle OSI et qui était valable pour les applications classiques doit être révisée. La liste des paramètres qui permet de caractériser les besoins d'un utilisateur n'est pas suffisante. En plus, l'OSI ne définit aucune politique de contrôle de la Qds ou d'intervention suite à une violation de la Qds. Une politique de mise en œuvre doit être définie pour permettre un contrôle efficace de la transmission et surtout pour être réalisable.

Le modèle de référence de RM-ODP [ODP-95] définit la qualité de service comme *«un ensemble de besoins relatif au comportement général d'un ou plusieurs objets »*. Cette définition est trop générale et inclut tous les paramètres du système sans aucune distinction. La définition de la qualité de service donnée par l'ITU [ITU-89] est : *« la qualité de service est l'effet global des performances d'un service qui détermine le degré de satisfaction de l'utilisateur vis à vis de ce service »*.

[Rac-91] a défini la qualité de service comme un ensemble de caractéristiques fournies par l'usager pour exprimer la performance d'un service donnée. Ces caractéristiques sont données dans un langage compréhensible par l'usager et elles ont des valeurs objectives et subjectives. Selon Vogel et al. [Vog-95], la qualité de service représente un ensemble de caractéristiques qualitatives et quantitatives d'un système multimédia distribué nécessaires pour réaliser le fonctionnement demandé d'une application. Ce fonctionnement représente la présentation des données à l'usager et la satisfaction de cet usager.

Actuellement, plusieurs applications MM sont offertes sur l'Internet comme : Visual Audio Tool (vat), Network Video Tool (nv), Network Voice Terminal (nevot), etc. Mais, aucune garantie de service n'est fournie. Étant donné que les applications MM (téléphonie ou télé-chirurgie par exemple) nécessitent un temps de réponse court, une synchronisation entre les flots reçus et une garantie de service, plusieurs travaux ont été réalisés pour répondre à ces besoins. Un travail important a été réalisé par le groupe des services intégrés (Int-Serv) du Internet Engineering Task Force (IETF). Ce travail consiste à ajouter de nouveaux protocoles à Internet afin de pouvoir assurer une garantie de service en utilisant un protocole de réservation de ressources RSVP [Wro-97]. Ce protocole assure une réservation des ressources tout au long du trajet de transmission des flots d'informations afin de contrôler et de garantir la qualité de service.

Dans une application MM, plusieurs composants doivent assurer un certain niveau de garantie de la Qds afin de satisfaire l'usager. Ces composants sont les stations de travail, le protocole de transport, le gestionnaire des fichiers du media continu, etc. C'est pour cela qu'une gestion de la qualité s'avère nécessaire afin de ne pas dépasser un délai, une gigue et un taux de perte spécifiques et d'assurer le cheminement de l'information aux usagers.

2-3 Les exigences de la Qds dans les applications multimédias

2-3-1 La performance du système de communication

Afin de transmettre les trames de l'audio et de la vidéo par exemple pour un film, deux approches sont proposées :

a) L'utilisation d'un seul canal

Dans cette approche, un seul canal est utilisé pour la transmission des données afin de conserver la dépendance temporelle entre les différents flots pendant une session donnée.

b) ***L'utilisation de plusieurs canaux***

Cette approche permet de transmettre chaque media dans un canal séparé avec la Qds demandée. Un mécanisme de synchronisation est disponible à la destination pour reconstituer la dépendance temporelle entre les différents flots. Cette approche a plusieurs avantages par rapport à la première. Ces avantages sont les suivants :

i) La transmission de chaque media avec une Qds donnée est possible. Le fait de regrouper plusieurs médias dans un seul canal nécessite l'utilisation de la meilleure Qds pour tous les médias. Ceci a pour but de trouver un compromis entre tous les usagers et satisfaire leurs demandes. Mais, cette solution n'assure pas nécessairement une optimisation des ressources utilisées puisque l'utilisation de la meilleure qualité nécessite beaucoup plus de ressources qu'une qualité moindre.

ii) La défaillance d'un seul canal n'implique pas l'arrêt total de la transmission. Le fait d'avoir, par exemple, une défaillance dans le canal de transmission de l'audio n'implique pas l'arrêt de la transmission des autres médias. Du point de vue de l'utilisateur, il est préférable de subir une dégradation d'un media donné que de ne plus recevoir aucune information.

iii) La localisation de tous les médias sur un seul site n'est pas nécessairement toujours possible.

En plus, pour transporter des informations MM, il faut un protocole de transport adapté. Ce protocole doit supporter le multipoint, réaliser les synchronisations entre les différents médias. Il doit permettre, également, de prendre en charge un trafic intensif.

Les protocoles de transport se décomposent en deux catégories :

a) Les protocoles de transport traditionnels

TCP est un protocole de transport orienté connexion et fiable. Il se caractérise par la transmission ordonnée des trames et la retransmission en cas d'erreur ou de perte de données. TP4 est un protocole similaire à TCP. Ces protocoles ne sont pas convenables pour la transmission des informations MM pour les raisons suivantes :

i) Le mécanisme de retransmission utilisé dans le cas d'erreur ou de perte de données ne conserve pas les contraintes temporelles du media continu et il introduit la gigue. Celle-ci représente la variation du délai à cause du temps d'attente dans les files des routeurs et des serveurs de données. En plus, les données audio et vidéo sont plus tolérantes aux pertes de trames que les données traditionnelles.

ii) Le multipoint n'est pas supporté ;

iii) La synchronisation des médias n'est pas assurée ;

iv) Le délai de transmission peut être long puisque le service fourni se base sur l'approche meilleur effort ;

v) La détection de la violation et la négociation de la Qds ne sont pas supportées ;

b) *Les protocoles de transport haut débit*

Les protocoles haut débit sont plus adaptés au multimédia. Ils permettent de garantir la Qds. Comme exemple de ces protocoles haut débit : TP5, XTP, NETBLT, TP++, VMTP et [Dia-97]. Pour clarifier ce type de protocole, NETBLT est détaillé ci-dessous.

Le protocole NETBLT (*Network Block Transfer*)[Cla-87] a été développé au MIT pour les gros transferts à haut débit. Il se place au-dessus d'une couche réseau de type datagramme. Les données sont envoyées sous forme d'une série d'unités appelées *tampons*. Ceux-ci sont mis en paquet par NETBLT qui les envoie par groupe, avec un nombre prédéterminé de paquets par groupe. Une connexion est mise en place pour laquelle est négociée la taille des tampons, des paquets et du groupe. Deux stratégies de contrôle de flux ont été définies : la première est effectuée par le client sur le nombre de tampons qui peuvent être envoyés, la deuxième est interne au protocole et travaille sur le taux de transmission et la taille du groupe.

La présence de la couche d'orchestration juste au-dessus de la couche transport est nécessaire pour organiser la communication de bout en bout. Les deux fonctions principales de cette couche incluent la synchronisation des flux de données et le multiplexage de ces flux pour optimiser les ressources des couches inférieures. En plus, la présence des réseaux haut débit, au-dessous de la couche transport, est aussi nécessaire pour acheminer les informations MM avec garantie de la Qds de bout en bout. Comme exemple de ces réseaux : ATM, FDDI, et DQDB [Tha-95].

2-3-2 La performance des stations de travail

Les systèmes de transport haut débit permettent la transmission des données alors que les stations de travail réalisent les traitements sur les données. Comme exemple de ces traitements, nous pouvons citer : le codage, le décodage et la présentation de l'audio et la vidéo à l'usager. Les paramètres des stations de travail ont un grand impact sur la Qds reçue par l'usager. Par exemple, la vitesse du CPU et des bus peut influencer le taux de trames de la vidéo présentée ; et un écran noir et blanc ne peut pas afficher des images en couleur. Par conséquent, les stations de travail doivent être performantes et doivent posséder un système d'exploitation, un gestionnaire de fichiers et un système de gestion de bases de données assez performants pour traiter les données en temps réel et garantir une qualité de service. En plus, la satisfaction des usagers d'une application MM réside dans la facilité d'effectuer des manipulations telles que : la création, la modification, l'envoi et la réception de l'information [Hua-97]. Ces manipulations peuvent être réalisées à travers une interface conviviale. Cette interface permet à l'usager de spécifier ses besoins dans un langage simple et facile. L'usager spécifie le type de media à recevoir, la résolution, la qualité de l'image et du son avec des termes familiers et sans être obligé de connaître les termes techniques comme le débit, le nombre de bits par seconde et le délai. L'interface doit permettre, aussi, de sauvegarder les profiles de l'usager pour lui éviter de refaire la sélection des paramètres de la Qds au début de chaque session.

2-3-3 La gestion de la qualité de service

La gestion de la Qds s'avère nécessaire afin de satisfaire les besoins de l'usager et lui garantir le service offert avec la qualité désirée. Les fonctionnalités de la gestion de la Qds sont: la spécification, le contrôle d'admission, la réservation des ressources, la négociation, la

renégociation et l'adaptation de la Qds [Hut-95, Haf-96a, Haf-96b, Boc-97, Haf-97a, Fis-97, Alf-98]. Nous allons traiter, ci-dessous, les principales fonctionnalités qui nous intéressent et qui sont les suivantes :

a) La négociation de la qualité de service

En utilisant la terminologie ISO concernant le service de connexion de type 'peer-to-peer', les trois acteurs de la négociation sont les usagers de service et le fournisseur de service [ISO-84]. Quatre primitives sont échangées durant la phase de la négociation (voir figure 2.1).

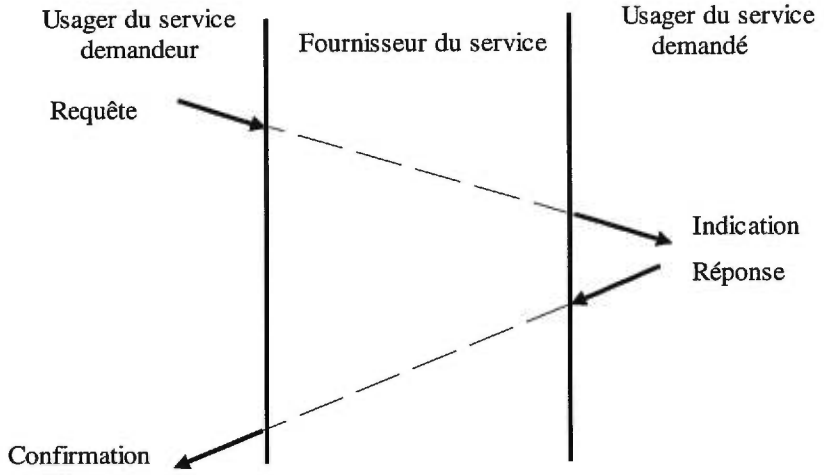


Figure 2.1. Les primitives échangées pour une négociation de la Qds

En se basant sur l'échange des valeurs des paramètres de Qds et le comportement du fournisseur du service, plusieurs variants de la négociation ont été définis [Pla-95, Haf-98] :

i. ***La négociation triangulaire*** : comprend l'usager du service demandeur 'service user calling', le fournisseur du service et l'usager du service demandé 'service user called'. D'abord, l'usager du service demandeur donne une spécification de la Qds au fournisseur de ce service par la primitive *'Requête'*. Celui-ci peut diminuer les valeurs des paramètres de la Qds. Le résultat de la spécification est transmis à l'usager du service demandé par la primitive *'Indication'*. Celui-ci peut diminuer encore les valeurs de la Qds et utilise la primitive *'Réponse'*. Le résultat final est retourné à l'usager du service demandeur en utilisant la primitive *'Confirmation'*. Le fournisseur du service et l'usager du service demandé peuvent rejeter la requête. Ce modèle de négociation est subdivisé en trois types de négociations qui sont les suivants:

- *Négociation triangulaire avec échange d'information* : l'usager du service demandeur introduit dans la primitive *'Requête'* la valeur du paramètre de la Qds (voir figure 2.2). Comme exemple, le protocole de transport de l'ISO TP4 utilise ce modèle;

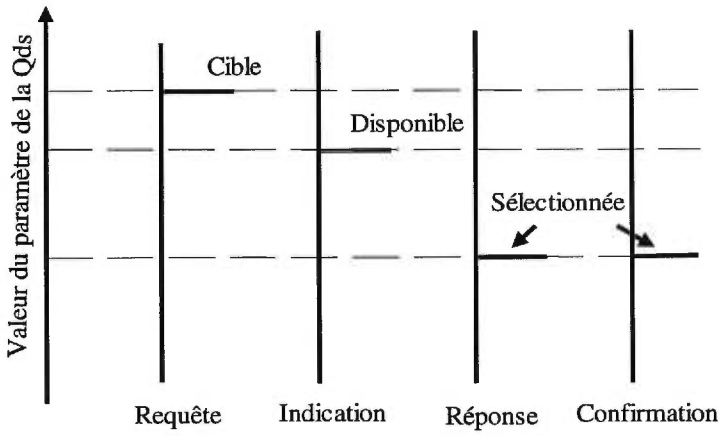


Figure 2.2. Négociation triangulaire avec échange d'information

- *Négociation triangulaire avec cible bornée* : l'usager du service demandeur introduit deux valeurs dans la primitive de la requête, la valeur ciblée et la valeur acceptée dans le pire cas (voir figure 2.3). Le protocole ST-II utilise ce modèle ;

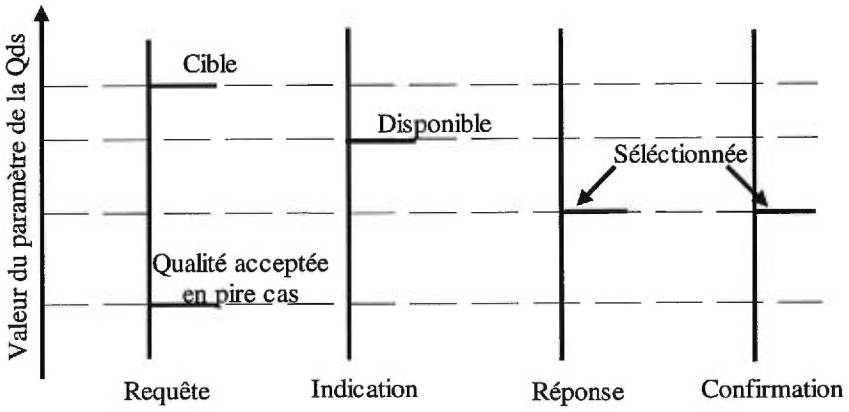


Figure 2.3. Négociation triangulaire avec cible bornée

- *Négociation triangulaire avec accord sur une valeur (Triangular negociation for a contractual value)* : Cette négociation a pour rôle de trouver un accord entre les usagers du service et le fournisseur du service. L'usager du service demandeur introduit deux valeurs dans la primitive de la requête, la valeur minimale acceptée en pire cas et la valeur limite de renforcement (bound of strengthening value). Celle-ci représente la

valeur maximale acceptée (voir figure 2.4). Le fournisseur du service et l'usager du service demandé peuvent réduire la valeur maximale si elle ne peut pas être satisfaite sans que la valeur minimale acceptée soit dégradée ;

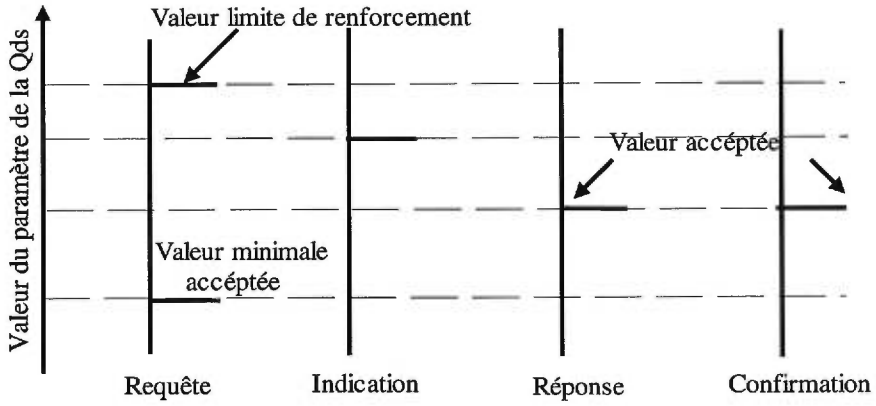


Figure 2.4. Négociation triangulaire avec accord sur une valeur

ii. ***La négociation bilatérale*** : se réalise entre les usagers du service. Le fournisseur du service ne peut pas changer la spécification de la Qds. Mais, le fournisseur du service et l'usager du service demandé peuvent rejeter la requête (voir figure 2.5).

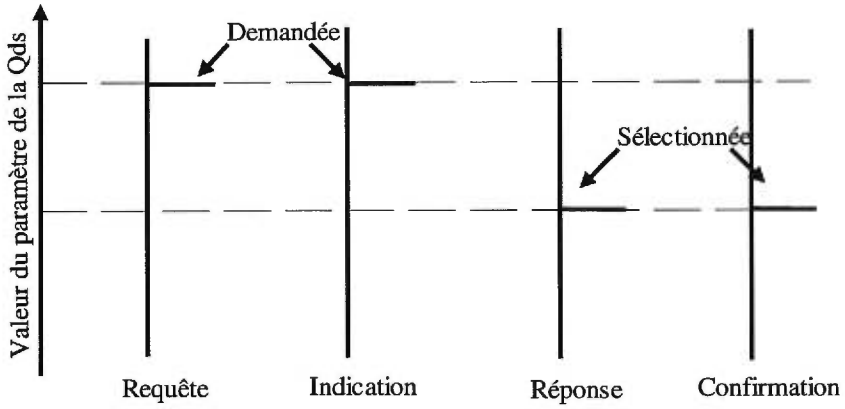


Figure 2.5. Négociation bilatérale

iii. ***La négociation unilatérale*** : L'usager du service demandeur propose une spécification de la Qds qui ne peut pas être modifiée par le fournisseur du service ou l'usager du service demandé. La négociation unilatérale se base sur l'approche «Tout ou rien» parce que la requête peut être soit acceptée soit rejetée (voir figure 2.6).

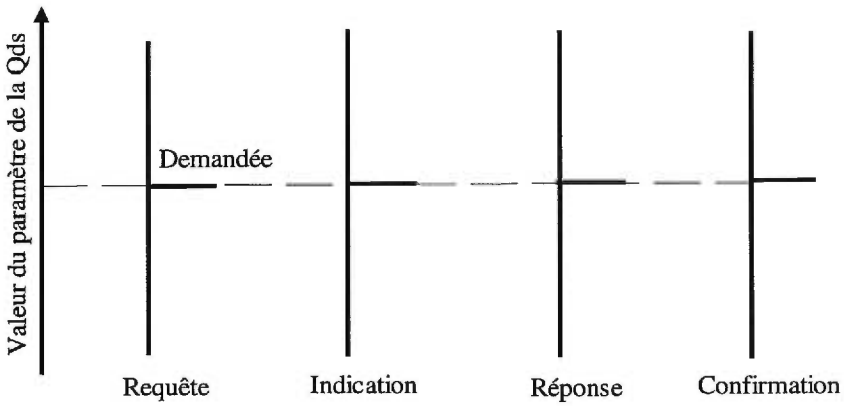


Figure 2.6. Négociation unilatérale

Le rôle de la négociation est de trouver un arrangement entre les besoins de l'usager et les ressources du système disponibles. D'abord, l'usager spécifie ses besoins en terme de Qds. Puis, le mappage (correspondance, 'mapping') des valeurs spécifiées par l'usager aux paramètres réels du système (débit, délai,... etc.) est effectuée [Hua-97]. Par exemple, l'usager choisit la vidéo en terme de résolution et de taux de trames qui vont être traduits par le débit. Ensuite, un contrôle d'admission est réalisé afin de comparer les ressources nécessaires pour satisfaire cette nouvelle demande avec les ressources du système disponibles. La décision d'acceptation dépend des caractéristiques du trafic généré par la nouvelle connexion et la disponibilité des ressources. Si les ressources sont disponibles pour satisfaire les besoins de l'usager sans entraîner une violation de la Qds aux autres usagers, une réservation effective de ces ressources est effectuée. Dans le cas contraire, la demande de l'usager est rejetée.

Lors de la négociation, le système doit indiquer à l'usager l'ensemble des Qds disponibles et les coûts associés à chaque qualité. L'usager choisit une qualité donnée selon ses préférences, la capacité de sa machine et le prix qu'il est prêt à payer. La notion du coût est importante. La négociation serait inutile si les usagers ne payaient absolument rien puisqu'ils choisiraient toujours la meilleure Qds disponible. Le coût d'une qualité de service dépend de plusieurs facteurs comme le type de service, le type de la garantie de service, la durée et la période du service. Ces différents facteurs seront traités dans le chapitre 6.

b) La supervision et l'adaptation de la qualité de service

Le mécanisme de supervision permet d'effectuer régulièrement des mesures sur la Qds fournie. Il permet de détecter une dégradation de la Qds si les valeurs de la Qds mesurées ne correspondent pas aux valeurs négociées. Le mécanisme de l'adaptation a pour rôle de maintenir les Qds négociées et de réagir aux changements du système à chaque fois qu'une violation est détectée. Si une violation est détectée à cause du partage de ressource, le mécanisme d'adaptation essaie de trouver une solution pour améliorer la qualité fournie (telle que le blocage des processus de faible priorité). S'il n'arrive pas à le faire, il réalisera une renégociation avec l'usager.

c) La renégociation de la qualité de service

La renégociation est réalisée sous la demande de l'usager, s'il veut changer de qualité pour recevoir une autre meilleure ou pire. Par exemple, un usager d'une application médicale de télé-consultation qui utilise une qualité de vidéo moins bonne, peut transiter vers une qualité meilleure pour visualiser des images à rayons X pour une certaine période. Ceci nécessite une renégociation de la Qds afin de recevoir une meilleure qualité qui permet d'accroître la largeur de bande utilisée pour recevoir les images à rayons X.

La renégociation est réalisée aussi sous la demande du système. Malgré le contrat réalisé entre l'usager et le système lors de la négociation, les valeurs de la Qds peuvent varier dans le temps à cause du partage des ressources et la congestion du système dans certains cas. Lorsque le protocole d'adaptation n'arrive plus à maintenir le service, une renégociation est réalisée pour changer de qualité. La notion du coût est aussi importante dans la renégociation et les mêmes mécanismes de contrôle d'admission et de réservation de ressources appliquées lors de la négociation, sont utilisés lors de la renégociation de la Qds. En ce qui concerne la réservation des ressources, elle doit être réalisée dans le réseau et les stations de travail pour maintenir un certain niveau de Qds. Ceci consiste à réserver une certaine quantité de ressources comme : le temps CPU, les tampons, la mémoire et la largeur de bande du réseau. Deux approches peuvent être appliquées pour la réservation des ressources :

i. *Approche optimiste* : le système réserve une quantité moyenne des ressources. Dans ce cas, une meilleure utilisation des ressources est constatée, mais, dans le pire cas, une violation de la Qds serait détectée ;

ii. *Approche pessimiste :* le système doit réserver suffisamment de ressources en se basant sur le pire cas, pour satisfaire les besoins du service demandé. Dans ce cas, la garantie du service est assurée à 100%, mais des demandes de service seront rejetées même si certaines ressources ne sont effectivement pas pleinement utilisées.

2-4 Les architectures de Qds

La majorité des travaux effectués dans le domaine de la qualité de service ont traité les composants du système d'une manière individuelle. Mais, les applications MM distribuées nécessitent que tous les composants du système distribué participent pour fournir et garantir un certain niveau de performance. Afin d'atteindre cet objectif, plusieurs architectures ont été proposées pour assurer une gestion de la qualité de service. Ces architectures définissent un cadre de travail complet et cohérent qui inclut les interfaces et les mécanismes de gestion à travers les différentes couches. Une description détaillée des différentes architectures est présenté ci-dessous [Cam-94, Haf-98, Aur-98].

2-4-1 Le modèle de Qds de Heidelberg

Le HeiProject du Centre Européen de gestion du réseau d'IBM à Heidelberg a développé un modèle complet de Qds qui fournit des garanties du service dans les stations de travail et le réseau. L'architecture de communication comprend un système de transport du media continu (HeiTS/TP) qui fournit un mécanisme pour le mappage de la Qds 'mapping' et la mise en échelle de la media ('media scalling', voir figure 2.7). La couche d'interconnexion de réseau, située sous la couche de transport, est basée sur ST-II qui garantie le service. En outre, le réseau fournit un routage basé sur la Qds (par l'intermédiaire d'un algorithme de recherche de la Qds) et un filtrage de la Qds. La clé de garantie du service de bout en bout est HeiRAT (technique de gestion de ressource). HeiRAT comporte un modèle complet de gestion de la Qds qui inclut la négociation de la Qds, le calcul de la Qds, le contrôle d'admission et la réservation des ressources. Le modèle d'Heidelberg de Qds permet une manipulation des demandes hétérogènes de la Qds de différents récepteurs et l'adaptabilité de la Qds à travers les techniques de filtrage et de mise en échelle de la Qds.

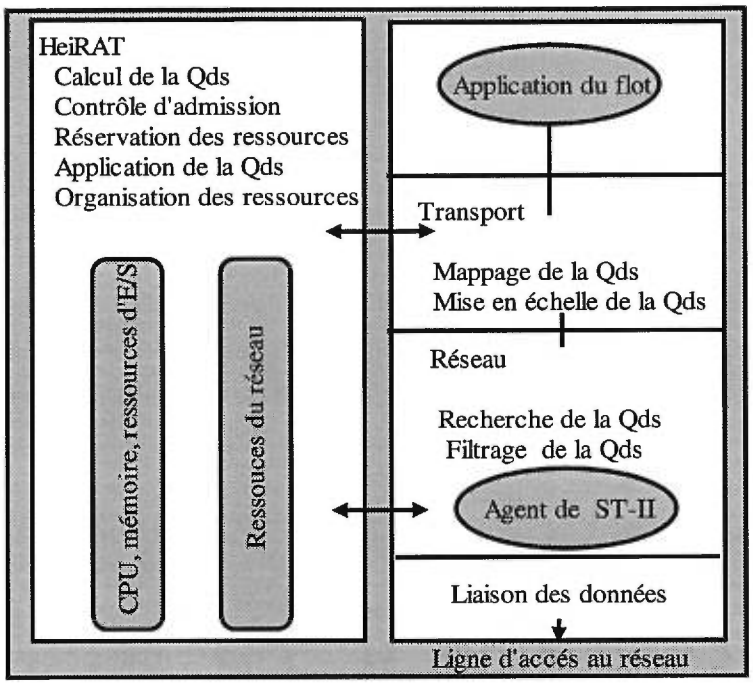


Figure 2.7. Modèle de Qds de Heidelberg : architecture du logiciel dans une station de travail

2-4-2 L'architecture XRM

Le groupe COMET de l'université de Colombie a développé un modèle de référence intégré et étendu (Extended Integrated Reference Model 'XRM') comme un modèle de gestion des réseaux de télécommunication de multimédia.

La figure 2.8 montre l'architecture XRM qui assure les fonctions suivantes :

i. Fonction de gestion, qui réside dans le plan de gestion du réseau (N-plan) et qui couvre les fonctionnalités de gestion du réseau et du système de l'ISO ;

ii. Fonction du contrôle du trafic, qui comprend le contrôle des ressources (M-plan) et les plans de la gestion et du contrôle de la connexion (C-plan). Le contrôle des ressources comprend la demande d'admission, la demande de routage dans le réseau, la gestion de la mémoire, le routage, le contrôle d'admission et le contrôle du flot dans les stations de travail;

iii. Fonction de transport de l'information, qui est située dans le plan de transport de l'usager (U-plan) et qui modélise les protocoles et les entités du media pour le transport de l'information de l'usager dans le réseau et les stations de travail; et

iv. Télébase, qui réside dans le plan d'abstraction et de gestion des données (D-plan) et qui représente l'information, les abstractions des données existantes dans le réseau et les stations de travail. Le télébase met en application le partage des données tout au long des plans de XRM.

Le XRM est conçu pour garantir la Qds dans le réseau ATM et les stations de travail. Les concepts généraux pour caractériser la capacité du réseau et les stations de travail (par exemple disques, commutateurs, etc.) ont été développés. Au niveau de la couche réseau, XRM caractérise la région de capacité d'un multiplexeur avec des garanties de Qds comme une région planifiée. Des ressources du réseau telles que la largeur de bande et la capacité de lien sont allouées en se basant sur quatre classes de trafic (classe I, II, III, et C) pour l'émulation de circuit, la voix et le vidéo, les données, et la gestion de réseau respectivement. Une classe de trafic se caractérise par ses propriétés statistiques et les besoins en Qds. Ceux-ci indiquent le taux de perte de cellules ATM et le délai. Afin de répondre efficacement aux exigences de la Qds du niveau de cellules, les algorithmes de gestion de tampons mémoires assignent dynamiquement la bande passante nécessaire pour la transmission et l'espace tampon mémoire d'une manière appropriée.

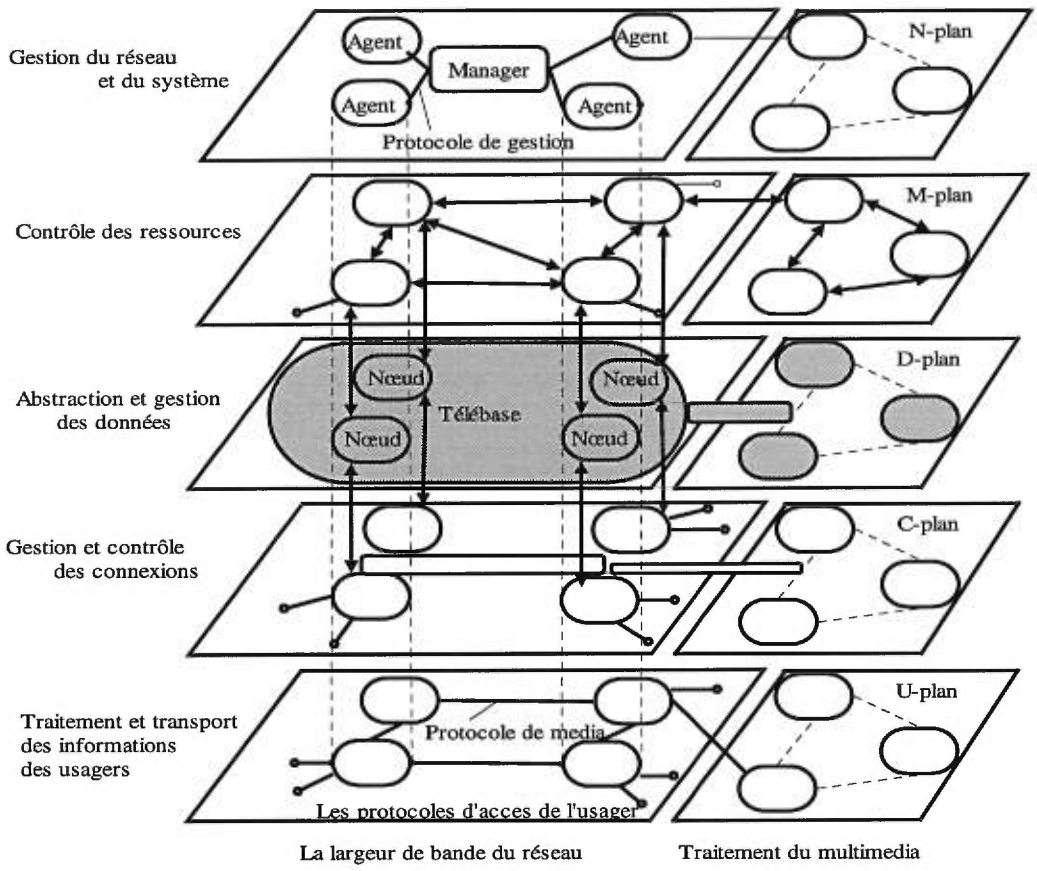


Figure 2.8. L'architecture XRM

2-4-3 L'architecture OMEGA

Au cours des trois dernières années, l'Université de la Pennsylvanie a développé l'architecture OMEGA. Celle-ci est le résultat d'une recherche interdisciplinaire qui examine la relation entre les besoins des applications en Qds (qui font des demandes de ressource rigoureuses) et la capacité de la gestion locale des ressources (le système d'exploitation) et globale (combinant la transmission et les ressources contrôlées à distance) afin de satisfaire ces demandes (voir figure 2.9). L'architecture OMEGA assume un sous-ensemble du réseau qui fournit des bornes sur le délai, le taux d'erreurs et un système d'exploitation qui est capable de fournir des garanties sur le temps d'exécution. L'importance de l'architecture OMEGA réside dans la réservation et la gestion des ressources de bout en bout. La transmission est précédée par une phase de demande de connexion. Dans cette phase, des besoins d'application, exprimés en termes de paramètres de Qds, sont négociés. En plus, des garanties sont assurées

aux différents niveaux logiques, comme entre les applications et le sous-ensemble du réseau. Ensuite, les connexions sont établies et la réservation des ressources est réalisée d'une manière appropriée.

Pour faciliter cette gestion de ressource, l'Université de la Pennsylvanie a également développé un modèle de courtage de Qds (Qos brokerage model) qui comprend le mappage, la négociation et la renégociation de Qds .

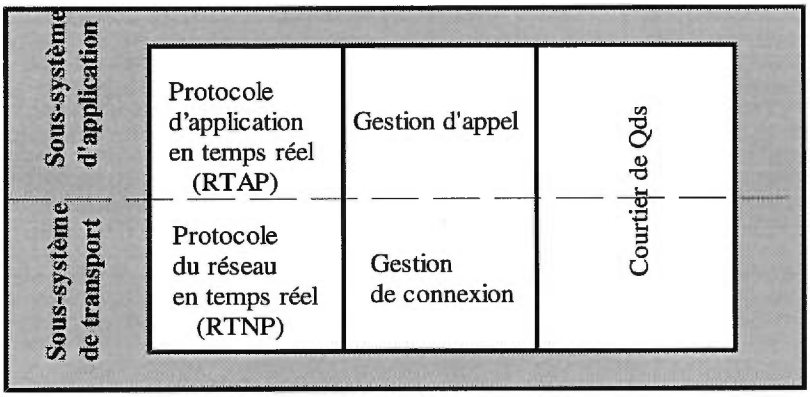


Figure 2.9. L'architecture OMEGA

2-4-4 L'architecture Int-Serv

Le travail du Groupe des services intégrés (Int-Serv) du Internet Engineering Task Force (IETF) est une contribution remarquable pour fournir un contrôle de Qds pour les applications multimédias au-dessus d'un réseau de services intégrés [Bra-94]. Le groupe a défini une architecture Int-Serv complète et un cadre de travail de Qds est utilisé pour indiquer la fonctionnalité des éléments du réseau.

Le comportement de ces éléments, qui constituent des commutateurs, des sous-réseaux et des systèmes d'exploitation, est capturé comme un ensemble de services. Chaque élément utilise une interface pour la définition de service. L'enchaînement des éléments de service fournit une Qds de bout en bout. Les services suivants sont aussi offerts en plus du meilleur effort:

i. Délai contrôlé, avec le fait que l'application choisit un niveau parmi l'ensemble des délais fournis ;

ii. Délai prédictif qui fournit une borne de délais statistiquement similaire au service statistique du groupe Tenet (voir section 2-4-7) et au service garantie de groupe COMET ; et

iii. Délai garanti qui fournit une borne absolue sur le délai.

Les flots dans une architecture Int-Serv se caractérisent par deux spécifications: une spécification du trafic et une spécification de demande de service. L'architecture Int-Serv, qui est limitée au réseau, mais applicable dans les stations du travail aussi, est composée de quatre composants:

i. Un organisateur de paquet, qui expédie des flots de paquets en utilisant un ensemble de files d'attente et de temporisateurs;

ii. Un classificateur, qui transforme chaque paquet entrant en classes de Qds;

iii. Un contrôleur d'admission, qui met en application l'algorithme de contrôle d'admission pour déterminer si un nouveau flot peut être accepté ou rejeté; et

iv. Un protocole de réservation (par exemple, RSVP [Zha-93]), qui est nécessaire pour créer et mettre à jour la spécification du flot dans les commutateurs tout au long du trajet du flot.

Le gestionnaire de qualité de service (QoS Manager, QM) est un élément de l'architecture Int-Serv dans les stations de travail. Le QM comme illustré dans la figure 2.10, présente une couche abstraite de gestion conçue pour isoler les applications des détails des services spécifiques fournis dans Internet orienté Qds (Qos-driven Internet). Un facteur de motivation derrière l'introduction d'un QM est que les applications peuvent négocier les Qds désirées sans être obligés de connaître les détails d'un service spécifique de réseau ; dans ce cas-ci, le QM fournit un degré de transparence par lequel les applications expriment les niveaux de Qds désirés dans un langage orienté application plutôt que d'utiliser des détails de la transmission de Qds. Le QM est responsable de déterminer quelles capacités de gestion de Qds sont disponibles sur le trajet de transmission de l'application, et choisit la voie la plus adaptée à l'application.

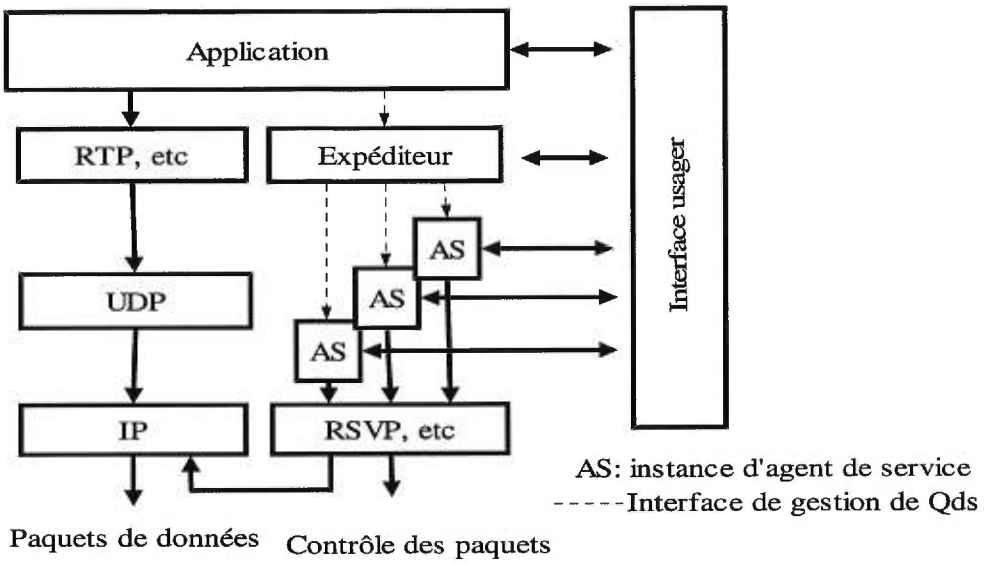


Figure 2.10. Le gestionnaire de la Qds (QM) de l'architecture Int-Serv

2-4-5 L'architecture Qos-A

L'architecture de la qualité de service (QoS-A) est une architecture qui fournit des mécanismes pour la gestion de la Qds et le contrôle de flot du media continu dans des réseaux multi-services [Cam-97]. L'architecture incorpore les notions principales suivantes:

- Flot, qui caractérise la production, la transmission et la consommation des trames de médias (unicast et multicast) avec la Qds associée;
- Contrats de service, qui sont des accords obligatoires entre les niveaux de Qds des utilisateurs et des fournisseurs; et
- Gestion de flot, qui fournit la surveillance et la maintenance des niveaux de Qds accordés.

La réalisation du concept de flot nécessite une bonne gestion de Qds et une intégration serrée entre les dispositifs de gestion, les stations de travail, les protocoles de transmission et les réseaux.

La figure 2.11 montre l'architecture QoS-A qui se compose d'un certain nombre de couches et de plans. La couche supérieure se compose d'une plate-forme d'applications distribuées avec des services pour fournir la transmission du multimédia et la spécification de la Qds dans un environnement basé sur les objets. Une couche d'orchestration, située au dessous de la plate-forme, fournit la correction de la gigue et des services de synchronisation

du multimédia à travers des flots multiples de l'application. La couche de transport contient un ensemble de paramètres de Qds qui configurent les services et les mécanismes. Au-dessous de cette couche, une couche d'interconnexion de réseaux et des couches inférieures forment la base pour le support de la Qds de bout en bout.

La gestion de Qds est réalisée dans les trois plans verticaux de la QoS-A. Le plan de protocole, qui consiste en sous-plans distincts d'utilisateur et de contrôle, est motivé par le principe de la séparation déjà présent dans RNIS et ATM. QoS-A utilise des protocoles séparés pour le contrôle à cause des différents besoins de Qds. Le plan de maintenance de la Qds contient un certain nombre de gestionnaires de Qds. Par exemple, la couche d'orchestration est intéressée à la synchronisation entre les différents flots. Par contre, le gestionnaire du transport de Qds est concerné par les intra-flots de Qds tels que la largeur de bande, la perte, la gigue et le délai. Le plan final de QoS-A concerne la gestion des flots, qui est responsable de l'établissement du flot (en incluant le contrôle d'admission de bout en bout, le routage basé sur la Qds et la réservation des ressources), le mappage de la Qds (qui traduit les représentations de Qds entre les couches) et la mise en échelle de Qds (qui est constitué du filtrage et de l'adaptation de Qds pour la maintenance du contrôle de Qds).

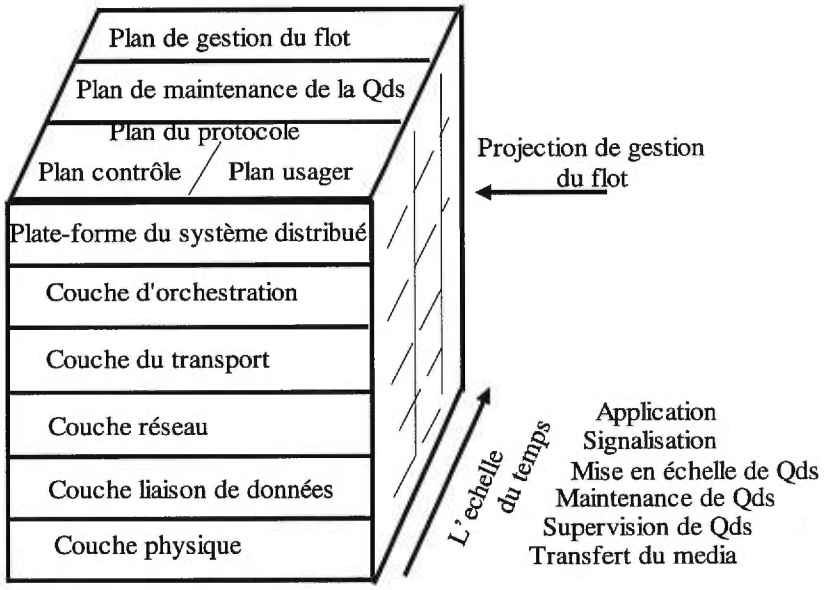


Figure 2.11. L'architecture Qos-A

2-4-6 Le cadre de travail de la Qds de l'ISO

Le cadre de travail de la Qds de l'ISO est une première contribution pour les architectures orientées Qds (QoS-driven) [ISO-95]. Il se concentre principalement sur la qualité de service de transmission de l'OSI. Le cadre de travail de la Qds de l'ISO définit largement la terminologie et les concepts de la Qds et fournit un modèle qui identifie les objets d'intérêt de la Qds dans les systèmes ouverts standards. La Qds, associée aux objets et leurs interactions, est décrite par la définition d'un ensemble de caractéristiques de la Qds. Les principaux concepts du cadre de travail de la Qds de l'ISO sont:

i. Les besoins de la Qds, qui sont réalisés par des entités de gestion et de maintien de la Qds;

ii. Les caractéristiques de la Qds, qui sont une description des mesures fondamentales de la Qds qui doivent être contrôlées;

iii. Les catégories de la Qds, qui représentent une politique régissant un groupe de besoins de la Qds spécifiques à un environnement particulier tel que des transmissions en temps-critique; et

iv. Les fonctions de gestion de la Qds, qui peuvent être appliquées à diverses caractéristiques de la Qds afin de répondre aux exigences de la Qds.

Le cadre de travail de la Qds de l'ISO se compose de deux types d'entités de gestion (à savoir entités des couches et entités du système) qui essaient de répondre aux exigences de Qds en surveillant, en maintenant et en contrôlant la Qds de bout en bout (voir figure 2.12). La tâche de la fonction de contrôle de politique (*PCF*) est de déterminer la politique qui s'applique à une couche spécifique d'un système ouvert. Elle modélise toutes les actions prioritaires qui doivent être exécutées pour contrôler le fonctionnement d'une couche. La définition d'une politique particulière est reliée à une couche spécifique et ne peut pas, donc, être généralisée. La politique peut, cependant, inclure des aspects de sécurité, de transmission en temps-critique et de contrôle de ressource. Le rôle de la fonction de contrôle de la Qds (*QCF*) est de déterminer, choisir et configurer les entités de protocole appropriées pour atteindre les buts d'une couche spécifique. L'agent de gestion du système (*SMA*) est utilisé en même temps que des protocoles de gestion de systèmes de l'OSI pour permettre aux ressources de système d'être contrôlées à distance. Le gestionnaire local des ressources (*RM*) assure le contrôle des ressources des stations de travail. La fonction de contrôle de la Qds du

système (*SQCF*) a pour but d'accorder l'exécution des entités de protocole et de modifier la performance des systèmes à distance par l'intermédiaire de la gestion de systèmes de l'OSI. L'interface de gestion des systèmes de l'OSI est fournie par le gestionnaire de gestion du système (*SMM*) pour superviser, contrôler et gérer les stations de travail. La fonction de contrôle de politique du système (*SPCF*) agit avec chaque fonction de contrôle de politique de chaque couche spécifique pour fournir une sélection globale des fonctions de la Qds.

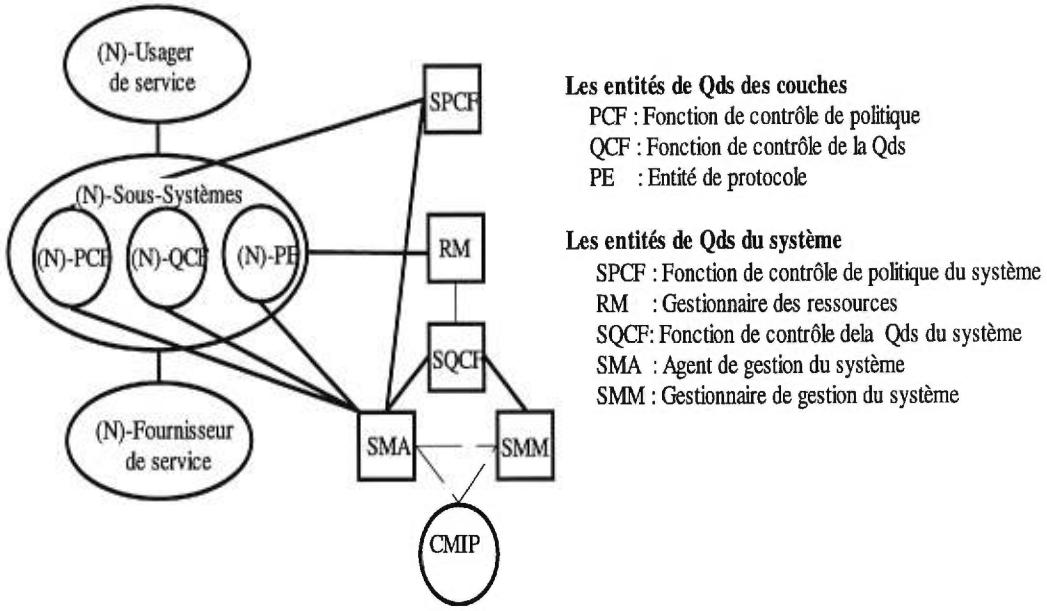


Figure 2.12. Le cadre de travail de la Qds de l'ISO

2-4-7 L'architecture Tenet

Le groupe Tenet de l'Université de Californie à Berkeley a développé une famille de protocoles qui fonctionne à travers le réseau ATM. L'architecture Tenet [Fer-92, Ban-96] inclut le protocole d'administration en temps réel (RCAP), le protocole Internet en temps réel (RTIP) et le protocole de transport du media continu (CMTP) (voir figure 2.13). Le premier fournit l'établissement de la connexion, la réservation des ressources et les fonctions de signalisation pour le reste de la famille de protocoles. RCAP couvre les couches de transport et de réseau pour la réservation globale des ressources et l'établissement de flot. CMTP est explicitement conçu pour le support du media continu. C'est un protocole qui s'exécute au dessus du RTIP et qui fournit d'une manière ordonnée et périodique les trames de media continu avec contrôle de la Qds à travers les bornes du débit, du délai et du taux d'erreur. Le

groupe Tenet fait une distinction entre les garanties déterministes et statistiques pour les flots en temps réel et du media continu respectivement. Dans le cas déterministe, les garanties fournissent une borne sévère sur la performance de toutes les cellules dans une session. Les garanties statistiques promettent que pas plus de x% de paquets auront un délai de transmission plus long que ce qui a été spécifié, ou pas plus que x% de cellules pourraient être perdues au cours d'une session donnée.

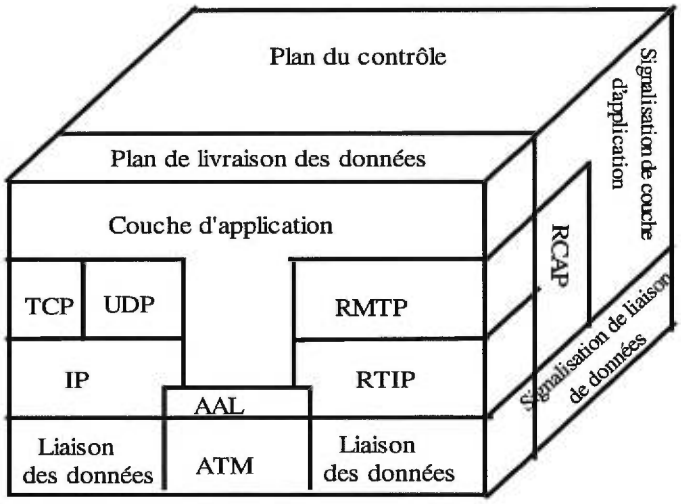


Figure 2.13. L'architecture Tenet

2-4-8 Le cadre de travail de la Qds de TINA

Le cadre de travail de la Qds de TINA décrit un cadre pour indiquer des aspects de la Qds des télécommunications distribuées dans le contexte de l'architecture de traitement. Le cadre de travail de la Qds considère les points de vue de traitement et d'ingénierie des applications distribuées de télécommunications. Il est régi par la séparation entre les applications de télécommunication et l'environnement de traitement distribué (DPE); c'est-à-dire, les services de multimédia offerts par un fournisseur utilisant le DPE et les capacités de traitement et de transmission. Du point de vue de traitement, les paramètres de la Qds exigés pour fournir des garanties aux objets sont énoncés d'une manière déclarative comme attributs du service. Dans le modèle d'ingénierie, des mécanismes de la Qds utilisés par des gestionnaires des ressources sont considérés. En énonçant d'une manière déclarative les besoins de la Qds, les applications sont libérées de l'obligation de faire face aux mécanismes complexes de gestion des ressources nécessaires pour assurer des garanties de la Qds.

2-4-9 Le modèle MASI de bout en bout

Le projet CESAME, au laboratoire de MASI à l'Université Pierre et Marie Curie à Paris, développe une architecture pour les transmissions de multimédia en mettant comme objectif principal le maintien de la Qds de bout en bout [Bes-94]. Comme l'architecture QoS-A, l'architecture MASI offre un cadre de travail de Qds générique pour définir et mettre en application les exigences de la Qds des applications multimédias distribuées qui fonctionnent sur un réseau ATM (voir figure 2.14). Le projet CESAME considère la gestion des ressources de bout en bout et qui couvre le système d'exploitation de l'hôte, le sous-système de transmission de l'hôte et le réseau ATM. Les raisons principales de développement de cette architecture sont :

i. La nécessité de correspondre les besoins de Qds de la couche d'ODP aux modules spécifiques des ressources d'une façon efficace;

ii. La nécessité de résoudre les besoins de la synchronisation de multimédia des trames multiples d'ODP ; et

iii. La nécessité de fournir un protocole de transmission approprié pour les services multimédias développés à l'Université Pierre et Marie Curie.

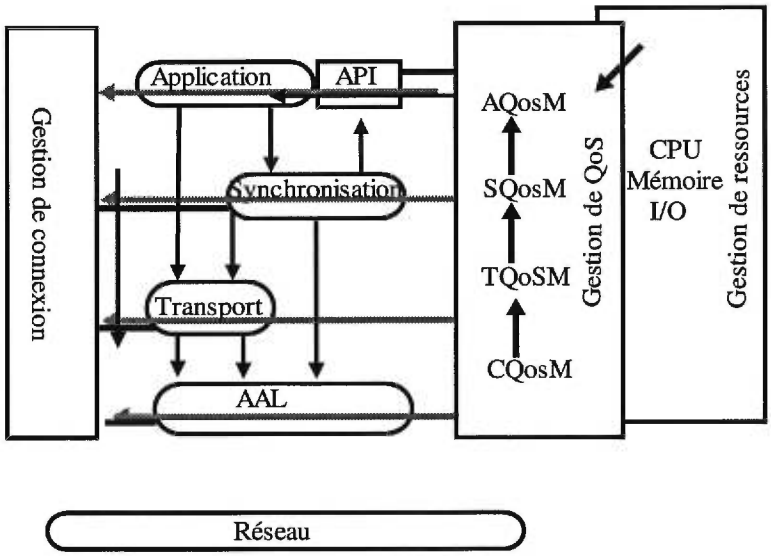


Figure 2.14. L'architecture MASI

2-4-10 Le cadre de travail de la Qds des stations de travail

À l'Université de Washington, un cadre de travail de la Qds pour fournir des garanties de la Qds pour les applications multimédias dans les stations de travail a été développé. Ce cadre de travail se compose de quatre composants: spécification de la Qds, mappage de la Qds, application de la Qds et implantation de protocole. La spécification de Qds est de haut niveau et utilise un nombre restreint de paramètres pour permettre aux applications une plus grande facilité pour qu'ils définissent leurs besoins de flots. Basées sur la spécification de la Qds, les fonctions de mappage de Qds satisfassent les besoins en ressource de bout en bout pour chaque session d'une application donnée. Les ressources considérées importantes dans le système sont l'unité centrale de traitement (CPU), la mémoire et le réseau.

La troisième composante du cadre de travail de la Qds est l'application de la Qds. L'application de la Qds est principalement concernée par fournir des garanties de traitement en temps réel pour la transmission de médias. Un appel en temps réel (Real-Time Upcall, RTU) a été développé pour des protocoles de structuration. Le dernier composant du cadre de travail est un modèle d'implémentation de protocole. Le code du protocole est structuré comme des appels en temps réel (RTU) avec des attributs déduits de la spécification de la Qds en utilisant la fonction de mappage de la Qds (mapping).

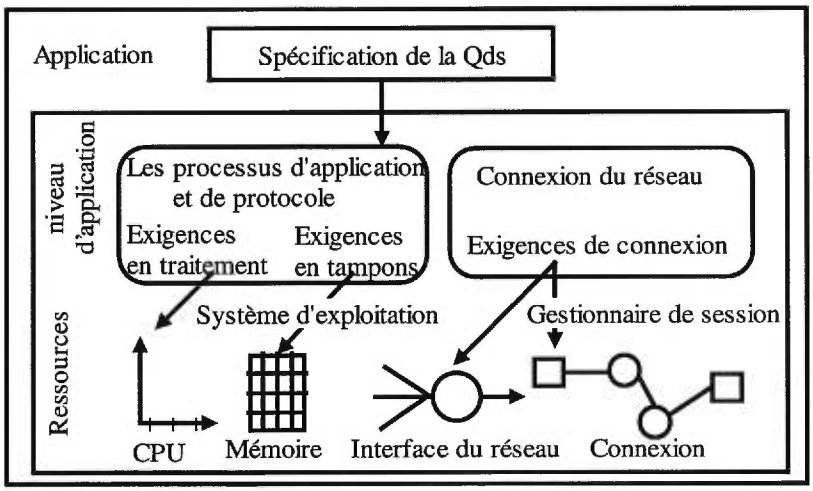


Figure 2.15. Le cadre de travail de la Qds des stations de travail

2-4-11 Le protocole de gestion coopérative de la Qds

Une architecture multi-agent de la gestion coopérative de la Qds a été développée à l'Université de Montréal par Hafid et al.[Haf-97b]. Cette architecture distribuée consiste à installer un agent de Qds dans chaque émetteur, chaque commutateur du réseau et chaque récepteur participant à une application multimédia donnée. Elle permet aux différents agents de communiquer et de coopérer entre eux afin d'assurer une gestion de la Qds de bout en bout. Les différentes fonctionnalités de gestion de la Qds comme la négociation, la renégociation et l'adaptation de la Qds sont effectuées d'une manière distribuée à travers le réseau.

Ce protocole de gestion de la qualité de service permet de surveiller les violations de la qualité de service et de reconfigurer l'arbre de diffusion afin de maintenir la qualité demandée par l'usager sans l'intervention de ce dernier. Ce mécanisme n'est pas disponible dans les protocoles existants. En plus, la notion du coût est importante dans les applications multimédias et ce protocole de gestion coopérative de la Qds peut être modifié afin de prendre cet élément en considération.

Dans le cadre de notre projet de maîtrise, nous allons spécifier ce protocole de gestion de la Qds. Ainsi, ce protocole sera largement détaillé dans le chapitre 4.

Chapitre 3.
Le langage de spécification formelle SDL

Le protocole de gestion coopérative de la qualité de service a été écrit dans un langage naturel informel. Ceci peut créer éventuellement des confusions et des ambiguïtés. Afin de pouvoir améliorer et compléter ce protocole pour vérifier son fonctionnement, nous l'avons spécifié en SDL. SDL est un langage qui permet d'aboutir à une spécification formelle d'un système, claire, précise et non ambiguë. Dans ce chapitre, nous allons présenter le cycle de vie de développement du logiciel. Ce cycle est très important puisqu'il permet de raffiner un modèle de base initial jusqu'à l'obtention d'un modèle détaillé. Les différentes phases de ce cycle permettent de faciliter la tâche des concepteurs tout en garantissant un produit final de haute qualité. Ensuite, nous allons présenter le langage de description formelle SDL.

3-1 Cycle de vie d'un logiciel

Le cycle de vie d'un logiciel est un modèle de développement qui est constitué de méthodes d'analyse, de conception, d'implantation, de test et de maintenance. Il peut être considéré comme un processus de raffinement successif qui permet de passer graduellement d'un certain niveau d'abstraction à un autre plus détaillé. Les phases du cycle de développement sont réalisées afin d'aboutir à un produit final d'une meilleure qualité. Comme illustré dans la figure 3.1, les trois phases importantes du processus de développement du système sont les suivantes:

1. Phase d'analyse des besoins

Les besoins des utilisateurs sont généralement rédigés dans un langage naturel d'une manière informelle complétés par des tables et des schémas. Ils se caractérisent par l'ambiguïté. Le passage direct à partir de ces besoins à un langage de programmation est difficile à réaliser. Dans cette phase, une validation complète de la spécification par rapport aux besoins des utilisateurs doit être réalisée ce qui permet d'avoir une spécification fonctionnelle des besoins des utilisateurs.

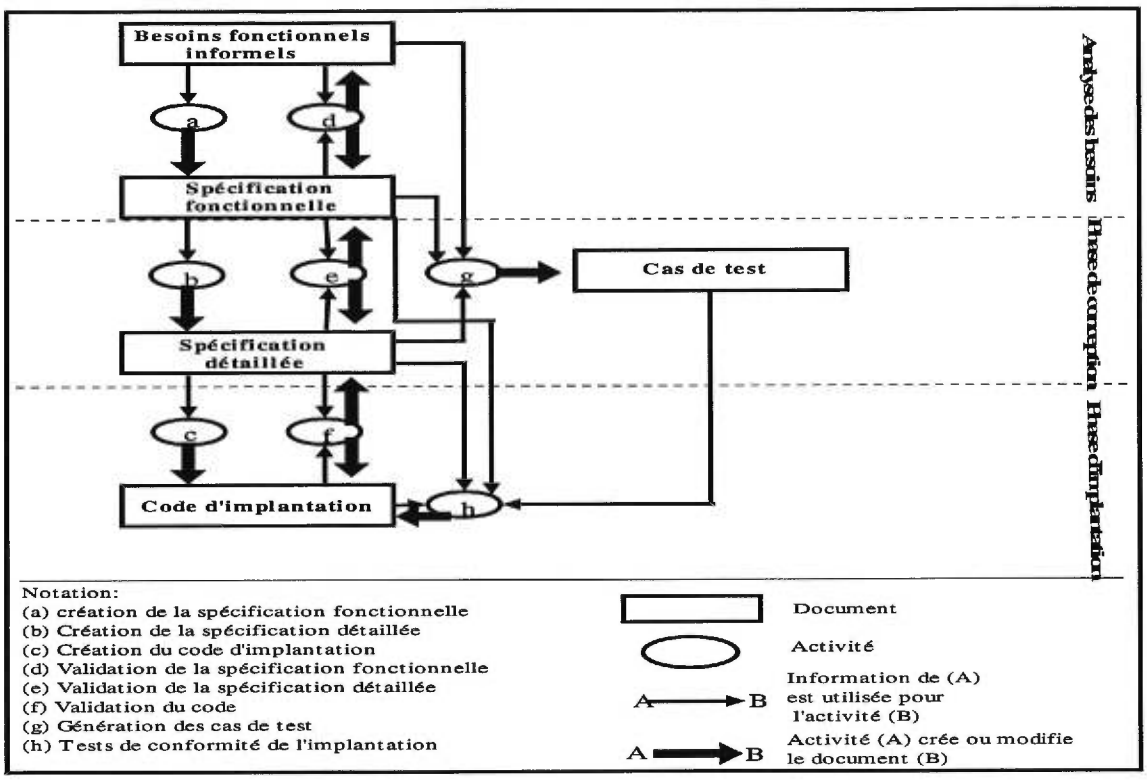


Figure 3.1. Le cycle de développement d'un logiciel

2. Phase de la conception

Cette phase consiste à produire une spécification détaillée. D'abord, une spécification formelle est réalisée à partir de la spécification fonctionnelle. L'utilisation d'une technique de description formelle est la clé de réussite pour le développement d'un système. Elle permet:

- La clarté, la précision, la non ambiguïté et la concision d'une spécification ;
- Une base complète et correcte pour la vérification d'une spécification ;
- La vérification de la conformité de l'implémentation par rapport à la spécification ;
- L'utilisation des outils informatisés pour créer, maintenir, vérifier, simuler et valider la spécification ; et
- La disposition d'un outil informatisé pour générer les applications sans faire recours à la phase de codage traditionnelle.

La spécification formelle obtenue est raffinée par la suite jusqu'à l'obtention d'une spécification assez détaillée (les structures de données, la description algorithmique des modules, etc.). Cette spécification est testée afin de détecter les différentes erreurs éventuelles telles que les situations de blocage (deadlock).

3. Phase d'implantation

La spécification détaillée obtenue à la fin de la phase de conception est traduite dans un langage de programmation. Ce passage de la description formelle au langage de programmation est généralement une activité semi-automatique. La validation de l'implantation est réalisée grâce aux activités de test. Celles-ci permettent de détecter les différentes erreurs introduites lors des phases précédentes.

3-2 Le langage SDL

Avant de passer à la spécification de notre système dans le chapitre 5, nous allons présenter, d'abord, l'histoire du langage de spécification et de description SDL, son domaine d'application, les notations et les concepts de base.

3-2-1 Historique

SDL (Specification and Description Language) est un langage de spécification formelle. Il a été développé et standardisé par l'ITU dans la recommandation Z.100. Le développement de SDL a commencé en 1972. La première version de ce langage a été publiée en 1976, suivie par de nouvelles versions en 1980, 1984 et 1988 et 1992 [Fae-94, Sar-94]. Cette dernière version a permis d'étendre considérablement le langage en introduisant le concept d'orienté objet.

SDL permet de spécifier un système en commençant par une vue globale de tout le système jusqu'aux niveaux de description les plus détaillés. Les types d'informations fournies par SDL sont le comportement et la structure [Bel-89, Bel-91].

3-2-2 Domaine d'application et notations

SDL a été développé dans le but d'être utilisé dans les systèmes de télécommunication, mais il peut être utilisé effectivement dans tout système temps réel, distribué et interactif.

SDL permet deux types de représentation. Du point de vue de la sémantique, elles sont équivalentes. Ces deux types sont: la représentation graphique (SDL_GR) et la représentation textuelle (SDL_PR). La représentation graphique fournit à l'utilisateur un ensemble de symboles graphiques pour la spécification du système et une syntaxe textuelle pour la définition de certains concepts comme les types de données abstraits. La représentation textuelle utilise uniquement la syntaxe textuelle.

3-2-3 Les concepts de base

La spécification d'un système en SDL permet de définir un ensemble de machines d'états finis étendues qui s'exécutent en parallèle. Ces machines sont indépendantes et communiquent en échangeant des messages appelés *signaux* [Bel-89, Bel-91,Fae-94, Sar-96].

3-2-3-1 La structure

La description du système constitue le plus haut niveau d'abstraction. Il s'agit d'un système (module interne) qui communique avec son environnement (module externe). La spécification du système en SDL est un ensemble de blocs (modules) interconnectés (voir figure 3.2). Chaque bloc peut être décomposé en un ou plusieurs sous-blocs, en formant ainsi une hiérarchie de blocs, ou plus précisément une structure d'arbre dont la racine est le système. La communication entre blocs ou entre le système et l'environnement se réalise par l'utilisation de canaux (*'Channels'*). Cette communication se fait de façon asynchrone par l'intermédiaire de files FIFO (First In First Out) non bornées qui contiennent les signaux transmis sur les canaux. Un canal peut être unidirectionnel ou bidirectionnel.

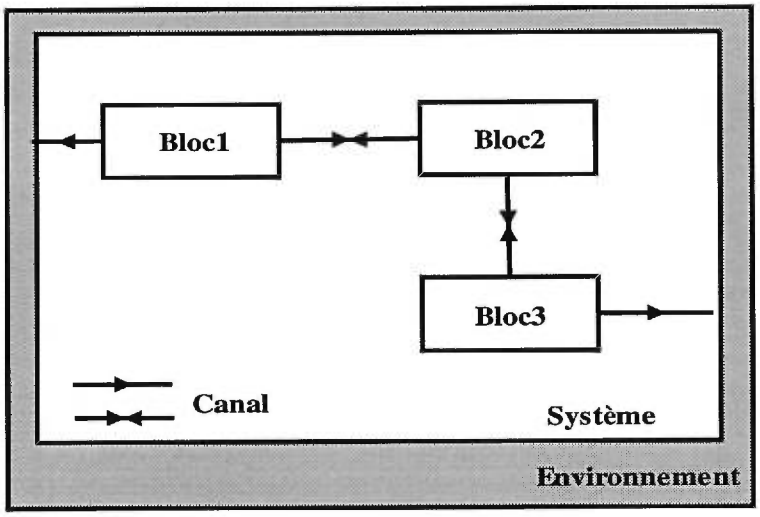


Figure 3.2. La structure d'un système SDL

Les blocs de bas niveau peuvent être décomposés en un ou plusieurs processus (voir figure 3.3). Chaque processus peut être décrit par une machine d'états finis étendue. Un processus peut communiquer avec un autre processus ou avec l'environnement à travers des chemins de signal (*'Signal route'*). Le nom du processus est suivi par deux nombres entre

parenthèses. Le premier indique le nombre d'instances de ce type pendant la création. Le deuxième indique le nombre maximal d'instances lors de l'exécution.

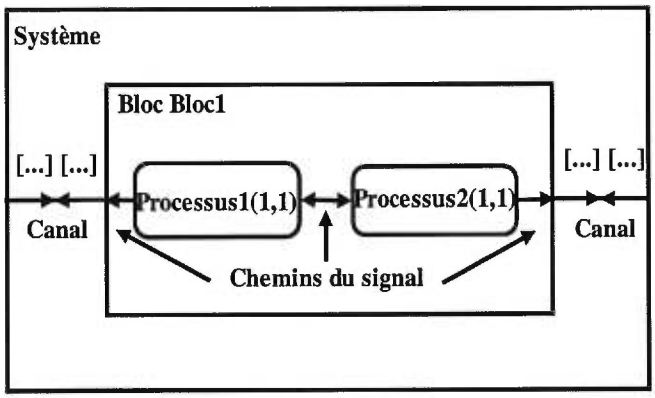


Figure 3.3. La structure d'un bloc SDL

3-2-3-2 Le comportement

Dans SDL, le comportement d'un système est défini par le comportement d'un ensemble de processus qui s'exécutent en parallèle. Le système et les blocs représentent la description statique de la structure du système. Un processus peut être crée statiquement au début du système ou dynamiquement durant l'exécution (voir figure 3.4). Chaque processus peut avoir une ou plusieurs instances. Le comportement de ces instances est identique. Cependant, chaque instance peut être dans un état différent par rapport aux autres instances. Chaque instance du processus a un identificateur unique (*Pid*). À chaque instance du processus est associé exactement une file d'entrée. Tous les signaux, destinés à cette instance et qui arrivent via différents chemins de signal, sont déposés dans la file en ordre d'arrivée FIFO (premier arrivé premier servi). Chaque instance du processus dispose, également, d'une mémoire. Celle-ci est utilisée pour le stockage des variables et des informations sur l'état courant. Chaque variable est la propriété d'une seule instance du processus. Les autres instances ne peuvent pas accéder au contenu de cette variable sans l'autorisation préalable de l'instance propriétaire.

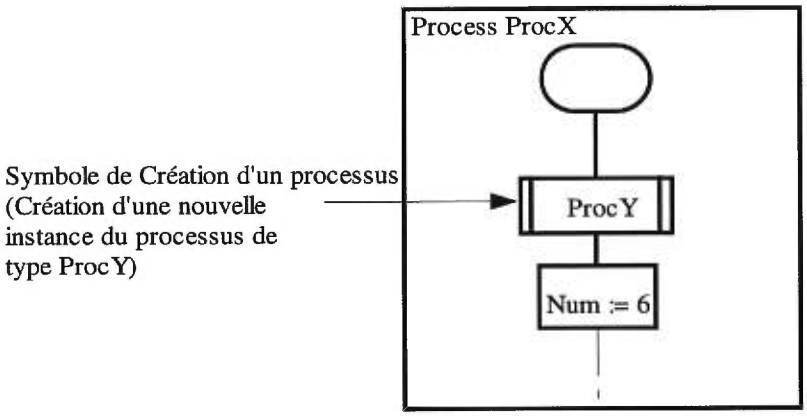


Figure 3.4 Création d'une instance du processus en cours de l'exécution.

Le comportement d'un processus est décrit par une machine à états finis étendue (EFSM), composée d'états et de transitions. À chaque instant, le processus peut soit appartenir à un état donné et il est en train d'attendre un signal d'entrée. Soit, il exécute les actions entre deux états. Pour chaque état, il y a un ensemble de signaux d'entrée *'Input'*. Si un processus attend un signal d'entrée, alors il est suspendu. La réception de ce signal est le seul événement qui peut permettre la transition vers un nouvel état. Un signal reçu peut avoir des paramètres qui vont être consommés par le processus récepteur.

La file d'entrée est utilisée par le processus de la manière suivante. Tout d'abord, le premier signal de la file est utilisé par le processus. Si ce signal peut déclencher une transition vers un nouvel état, alors ce signal est consommé par le processus. Ce type de transition est désigné par *'transition explicite'*. Sinon, le signal est tout simplement éliminé de la file et le processus reste dans le même état. Cette transition est appelée *'transition implicite'*. Dans le cas de l'utilisation de constructeur sauvegarde *'Save'*, le signal est sauvegardé dans la file pour un besoin ultérieur et le signal suivant sera considéré. La sauvegarde des signaux n'affecte pas l'ordre des signaux dans la file. Un exemple de l'utilisation de la file d'attente par le processus est illustré dans la figure 3.5.

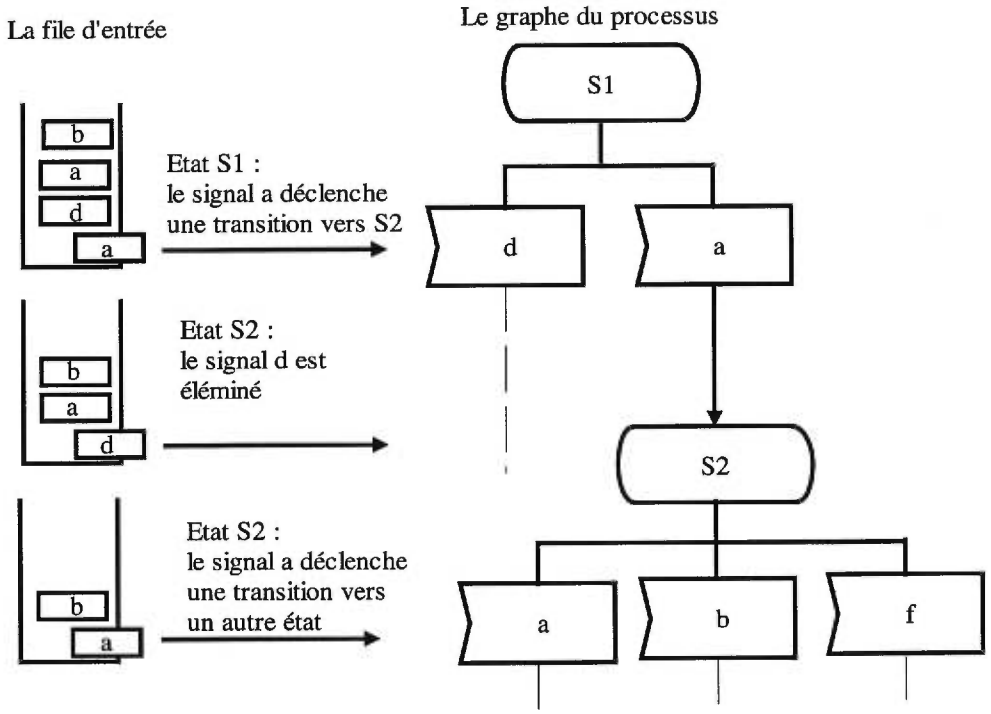


Figure 3.5. Les mécanismes d'une file d'entrée d'un processus

Le processus de la figure 3.5 est dans état S1. La file d'entrée contient les signaux 'a', 'd', 'a' et 'b' dans cet ordre. Les signaux 'a' et 'd' peuvent déclencher une transition. Le signal 'a' est le premier signal dans la file. Ce signal est consommé et la transition est réalisée vers l'état S2. Le signal 'd' devient le premier dans la file. Ce signal ne peut pas déclencher une transition explicite du processus puisqu'il n'est pas défini dans l'état S2. Le processus élimine le signal et reste dans le même état S2. Le signal suivant est 'a'. Le processus consomme ce signal et transite vers un autre état.

La transition d'un processus vers un autre état consiste à exécuter une suite d'actions. Ces actions peuvent être une manipulation de données, une prise de décisions, une création de nouvelles instances de processus, ou l'envoi de signaux de sortie à un autre processus ou à l'environnement.

3-2-3-3 La communication

La communication entre les processus est réalisée par l'échange de signaux en utilisant les chemins du signal et les canaux. La figure 3.6 présente un exemple de communication entre des instances de processus.

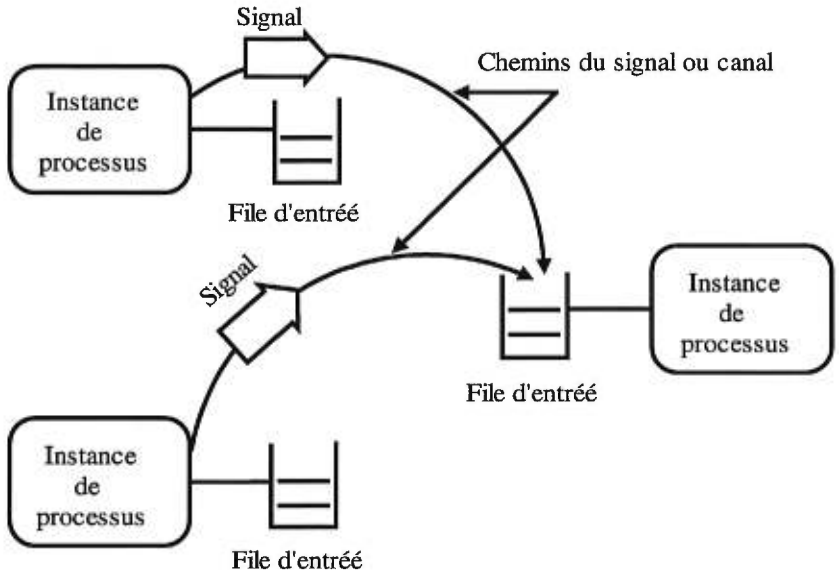


Figure 3.6. Communication entre processus

Un signal envoyé par un processus a une seule destination. celle-ci peut être spécifiée soit :

- Explicitement, en spécifiant l'identificateur du processus récepteur;
- Implicitement, en ne spécifiant aucun identificateur.

L'adressage explicite consiste à envoyer le signal de sortie 'Output' suivi du mot clé 'To' et l'adresse du processus récepteur de ce signal. Chaque processus a une adresse unique dans tout le système. Cette adresse est créé par la machine SDL. Elle diffère du nom du processus puisqu'il est possible d'avoir plusieurs instances d'un certain type de processus avec le même nom. Les adresses sont du type *Pid* qui est un type prédéfini de SDL. Pour chaque processus, quatre expressions du type Pid sont prédéfinies :

i. ***Self*** : contient l'identificateur du processus lui même ;
ii. ***Sender*** : contient l'identificateur du processus émetteur du dernier signal consommé par le processus ;
iii. ***Offspring*** : contient l'identificateur du dernier processus créé par le processus ; et
iv. ***Parent*** : contient l'identificateur du processus créateur de ce processus.

Dans l'exemple de la figure 3.7, le bloc B1 contient deux processus de type P1 et P2. Nous disposons d'une seule instance pour chaque type. Ceci est indiqué par l'expression (1,1) qui suit le nom du processus. Le processus P1 peut envoyer des signaux à travers les chemins du signal Sr1 et Sr2. Le signal A peut être envoyé sur les deux chemins du signal. Pour spécifier que le processus P2 est le processus destinataire, P1 doit connaître l'identificateur du P2. Le signal sera envoyé en utilisant le mot clé 'To' et l'identificateur du P2. Le signal B peut être envoyé uniquement à travers le chemin du signal Sr1. Par conséquent, le processus destinataire du signal B peut ne pas être spécifié explicitement par P1. L'adressage implicite est simple puisque la communication est réalisée sans que le processus connaisse les adresses des autres.

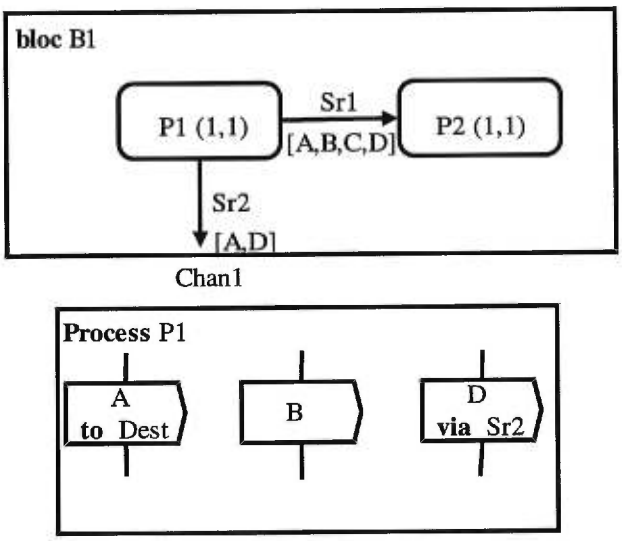


Figure 3.7. Adressage explicite et implicite

L'adressage peut être réalisé, également, par la spécification du chemin de signal ou du canal. Dans certains cas, le processus expéditeur du signal, ne dispose pas de l'identificateur du processus destinataire et le signal ne dispose pas d'une seule destination. C'est pour cela que le processus récepteur doit être spécifié par le nom du chemin du signal ou du canal à travers lequel le signal est envoyé. Ceci se réalise par l'utilisation du mot clé *'via'*. La figure 3.7 montre l'utilisation du terme Via. Le signal D peut être transmis sur les deux chemins du signal Sr1 et Sr2. L'adressage implicite n'est pas possible parce que la destination du signal D n'est pas déterminée par la structure du système.

3-2-3-4 Les types de données

Les opérations sur les variables d'un processus donné permettent de changer son état global. Les variables engendrent des transitions et peuvent, ainsi, affecter le comportement d'un processus. SDL permet de définir différents types de données. Nous pouvons citer les types suivants:

i. Types prédéfinis

SDL fournit un ensemble de types prédéfinis. Les types prédéfinis simples en SDL sont:

- *Integer* (Entier),
- *Boolean* (Booléen),
- *Real* (Réel),
- *Natural* (Naturel),
- *Character* (Caractère),
- *Time* (Temps): représente un instant du temps absolu.
- *Duration* (Durée): représente une durée de temps relative.
- *Pid* (Identificateur de processus): représente le type d'identification des instances de processus.

ii. Type Structure

L'exemple suivant permet de montrer l'utilisation du type *Structure :*

newtype Elément s*truct*

Numéro : *Integer*;

Longueur : *Real*;

endnewtype Elément

Ceci permet de définir un enregistrement 'Elément'. Ce dernier est constitué de deux champs Numéro et Longueur.

iii. Type Array

L'exemple suivant permet de montrer l'utilisation du type *Array:*

newtype Tableau

Array(integer, *Real)*

endnewtype Tableau

Ceci permet de définir un 'Tableau' qui contient des éléments réels.

iv. Changement du nom du type et sous-type

Parfois, il est préférable d'utiliser un type dans des contextes différents. Le mot clé 'Syntype' permet d'utiliser un type existent avec un autre nom. Comme exemple:

syntype Int = Integer; *endsyntype*

En plus, l'utilisateur peut définir un sous ensemble d'un type de données de la manière suivante:

syntype indice = *Integer*

Constants 1:20

endsyntype

Cet exemple permet de spécifier un sous ensemble de 20 éléments entiers.

3-2-3-5 La notion du temps

Les processus en SDL ont accès au temps par l'utilisation des temporisateurs, des variables et des expressions de type *Time* et *Duration*. *Time* définit un temps absolu, alors que *Duration* définit un temps relatif.

Un temporisateur est un objet appartenant à un processus donné. Il est du type prédéfini *Timer*. Il est soit actif soit inactif. La primitive *Set (Now + Durée, T)* permet d'activer un temporisateur. *Now* est de type *Time* et permet de donner le temps actuel durant l'exécution du système. Durée est de type *Duration* et représente la durée nécessaire pour l'expiration du temporisateur (normalement une constante). T désigne le nom du temporisateur et représente son identificateur dans le processus. Le temporisateur T supervise le temps actuel. Une fois le temps actuel dépasse le temps initialisé, un signal *de type T* est généré. Ce signal est mis dans la file d'entrée du processus et le temporisateur revient à l'état inactif *'idle'*. Un temporisateur actif est désactivé par la primitive *Reset(T)* et le signal Timer est éliminé de la file.

3-2-3-6 L'expression Export/Import

SDL ne permet pas une déclaration de variables d'une manière globale. Chaque processus définit ses propres variables d'une manière locale et les autres processus n'y peuvent pas accéder. Pour connaître la valeur d'une variable d'un autre processus, la communication par les signaux est nécessaire. Une autre méthode plus facile est l'utilisation de l'expression Export/Import. Si un processus veut rendre une variable accessible par les autres processus, il doit la déclarer en utilisant le mot clé *Exported*. Par exemple:

dcl exported Var ***Integer*** ;

Le processus propriétaire de cette variable Var, de type *Integer,* utilise le mot clé *Export* pour la rendre accessible par les autres processus de la manière suivante:

Export (Var) ;

Cette opération permet de faire une copie du contenu de cette variable. Cette copie implicite est utilisée pour répondre aux besoins des autres processus.

Le processus qui veut avoir le contenu de la variable Var effectue la déclaration suivante:

imported Var ***Integer***;

Pour visualiser le contenu de cette variable sans pouvoir modifier la variable réelle, le processus utilise cette expression:

import (Var, Proc_Exp) ;

Proc_Exp est du type *Pid* et permet de spécifier l'adresse du processus propriétaire de la variable Var. Cette adresse peut être omise si le processus propriétaire est le seule qui possède une variable avec le même nom et le même type.

3-2-4 Les constructeurs de base

La description d'un système avec la notation graphique du SDL (SDL_GR) nécessite l'utilisation des constructeurs de base. Une description détaillée des principaux constructeurs de base est donnée dans les tables suivantes en spécifiant pour chaque symbole, son nom et son rôle :

1. Les constructeurs de base d'un système

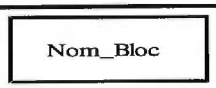
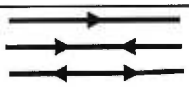
Symbole	Nom du constructeur	Explication
	Bloc	Il permet de spécifier le bloc désigné par *Nom_Bloc.*
	Déclaration de signaux	Il permet de spécifier les signaux échangés entre blocs.
	Canaux	Il spécifie les canaux unidirectionnels et bidirectionnels
[ ]	Liste des signaux	Il représente la liste des signaux transmis sur un canal.

Table 3.1. Les constructeurs de base d'un système

2. Les constructeurs de base d'un bloc

Pour la spécification d'un bloc, les symboles de déclaration de signaux et de listes de signaux de la table 3.1 sont utilisés avec les symboles suivants:

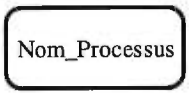
Symbole	Nom du constructeur	Explication
	Processus	Il permet de spécifier le processus désigné par *Nom_Processus.*
	Chemins du signal	Il spécifie les chemins du signal unidirectionnels et bidirectionnels.

Table 3.2. Les constructeurs de base d'un bloc.

3. Les constructeurs de base d'un processus

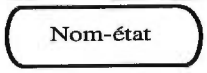
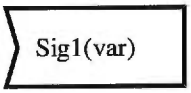
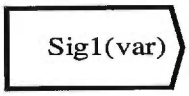
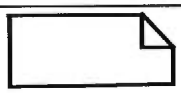
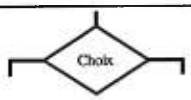
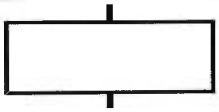
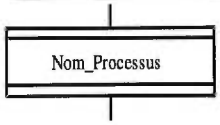
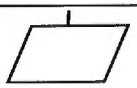
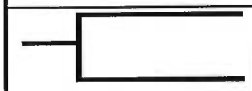
Symbole	Nom du constructeur	Explication
	Début d'un processus	Il représente le point d'entrée au processus.
	Etat	Il représente un état possible du processus nommé *Nom-état.*
	Entrée	Il représente une entrée d'information et il déclenche le début d'une transition du processus vers un autre état.
	Sortie	Il représente une sortie d'information qui est envoyé vers une destination.
	Déclaration	Il est utilisé pour la déclaration des variables.
	Décision	Il est utilisé pour spécifier un choix dans les actions à exécuter.
	Tâche	Il permet de spécifier des actions comme la manipulation des données.
	Création d'un processus	Il représente la création d'une instance du processus désigné par *Nom_Processus.*
	Sauvegarde	Il permet de sauvegarder les signaux non utilisés pour une réutilisation ultérieure.
	Extension du texte	Il dépend du symbole auquel il est attaché. Il permet l'extension du texte.
	Commentaire	Il permet de spécifier un commentaire pour rendre la spécification plus compréhensible.
	Branchement	Il permet de se brancher vers l'étiquette *Nom_Étiquette* pour compléter les actions.
	Arrêt	Il représente l'arrêt d'une instance d'un processus.

Table 3.3. Les constructeurs de base d'un processus

4. Les constructeurs de base d'une procédure

Les symboles reliés à la manipulation des données de la table 3.3 sont utilisés avec les symboles suivants:

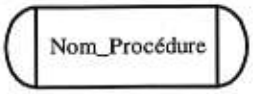
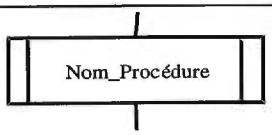
Symbole	Nom du constructeur	Explication
	Déclaration d'une procédure	Il permet de définir la procédure désignée par *Nom_Procédure*.
	Appel d'une procédure	Il spécifie un appel de la procédure désignée par *Nom_Procédure*.
	Début d'une procédure	Il spécifie le début d'une procédure.
	Fin d'une procédure	Il représente la fin d'une procédure.

Table 3.4. Les constructeurs de base d'une procédure

Le langage SDL possède d'autres constructeurs pour créer et utiliser les services, les macros, ...etc. Pour des informations plus approfondies voir [Bel-89, Bel-91, Fae-94, Sar-96].

3-2-5 Un exemple de spécification en SDL

Le comportement d'une machine à états finis sous forme d'un graphe orienté est illustré dans la figure 3.8. Ce graphe a deux états S1 et S2. Si la machine est dans l'état S1, elle peut exécuter une transition vers l'état S2 en consommant le signal 'A'. Lors de la transition, elle va envoyer le signal 'C'. La consommation du signal 'B' dans l'état S1 va engendrer une transition vers le même état et l'envoi du signal 'D'. Si la machine est dans l'état S2, elle peut exécuter une transition vers l'état S1 en consommant le signal 'A, et en envoyant le signal 'C'. La consommation du signal 'B' dans l'état S2 permettra une transition vers le même état et l'envoi du signal 'E'.

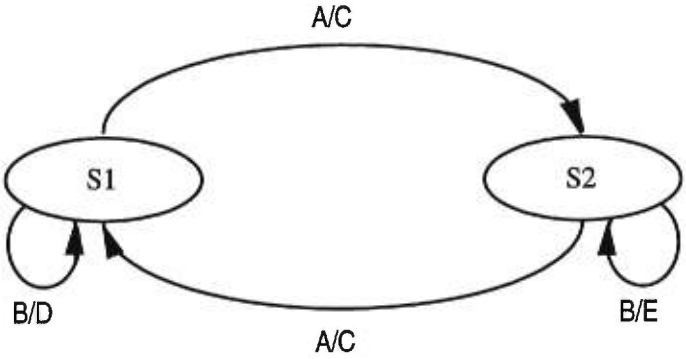


Figure 3.8. Le comportement d'une machine à états finis.

La description de la machine à états finis de la figure 3.8 sous forme d'un diagramme de processus en utilisant les constructeurs de base est illustré dans la figure 3.9.

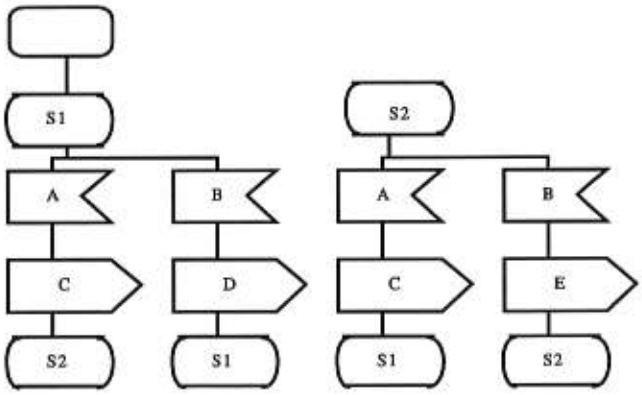


Figure 3.9. Le diagramme de processus d'une machine à états finis

Le langage de spécification formelle SDL a été utilisé pour spécifier le protocole de la gestion coopérative de la qualité de service dans les applications multimédias. Le chapitre suivant introduit le principe de base de ce protocole, les fonctionnalités de la gestion de la Qds et les différentes opérations réalisées.

Chapitre 4.
La gestion coopérative de la qualité de service dans les applications multimédias

Actuellement, la majorité des systèmes multimédias existants se basent sur l'approche 'meilleur effort'. Dans cette approche, les ressources du réseau sont partagées entre les différents participants sans contrôle d'admission ni réservation des ressources au préalable. Aucune garantie n'est assurée sur les services offerts. En plus, une gestion centralisée de la qualité de service utilisée dans certaines applications MM (comme l'accès aux serveurs de données) pose certaines restrictions sur le nombre de sources et de destinations utilisées. La gestion centralisée de la qualité de service a plusieurs inconvénients comme la surcharge du réseau, les goulots d'étranglement, etc. Elle ne permet pas aussi de réaliser la négociation entre chaque source et chaque participant dans le cas des applications multimédias de grande taille. Ces applications utilisent un nombre très grand de participants comme une application de télédiffusion haute définition (HDTV) ou une application de téléconférence. Une application MM doit aussi présenter à l'usager plusieurs qualités de service. Les systèmes existants offrent uniquement une seule qualité. Cette approche ne convient pas à tous les usagers. En effet, certains usagers sont prêts à payer cher pour obtenir une meilleure qualité alors que d'autres préfèrent payer moins cher pour une qualité moindre. En plus, pour une application de téléconférence, la réception d'une seule qualité n'est pas appropriée pour tous les participants. Certains d'entre eux peuvent utiliser des stations de travail qui ne sont pas assez performantes pour supporter la qualité reçue par la majorité des participants. Par conséquent, le système doit proposer plusieurs qualités avec différents prix. Il doit assurer le maintien du service offert afin que l'usager ne reçoive pas une qualité dégradée. La gestion coopérative de la Qds (CQoSM) [Haf-97b] permet de gérer la qualité de service d'une manière plus efficace. Les différentes fonctionnalités de gestion de la Qds comme la négociation, la renégociation et l'adaptation de la Qds sont effectuées d'une manière distribuée à travers le réseau. Par conséquent, le nombre d'usagers peut augmenter sans poser aucun problème pour le bon déroulement de l'application. En plus, elle offre plusieurs qualités de service afin que l'usager choisisse la qualité la plus appropriée.

4-1 Principes de base

La gestion coopérative de la Qds (CQosM) [Fis-97, Haf-97a, Haf-97b] a été développée en prenant en compte les besoins des applications multimédias nécessitant plusieurs participants en même temps telles que les applications de téléconférence et de télé-éducation. Dans cette approche, une connexion est établie à partir d'un point vers les autres points (point-to-multipoint). Les flots de données sont transmis par diffusion à partir de l'émetteur vers les différents récepteurs sous forme de plusieurs variantes de qualités de service: bonnes, moyennes et moins bonnes. La négociation de Qds ne se fait pas d'une manière individuelle entre l'émetteur et les récepteurs et chaque utilisateur doit choisir une qualité parmi l'ensemble des qualités proposées. Ce choix dépend des besoins et du budget de chaque utilisateur, ainsi que de la capacité de sa machine.

La gestion coopérative de la Qds est la responsabilité des serveurs de bases de données, du réseau et des stations de travail. Elle consiste à installer un agent de Qds dans chaque émetteur, chaque commutateur du réseau et chaque récepteur participant à une application donnée. À chaque application multimédia est associé un arbre de diffusion. Chaque agent commutateur de cet arbre connaît ses voisins de niveau supérieur et de niveau inférieur. Les informations concernant les voisins peuvent être facilement établies lors de la construction de l'arbre de diffusion, c'est-à-dire à la connexion et à la déconnexion d'un utilisateur. Ces agents de Qds peuvent communiquer avec les agents voisins afin de les informer sur les qualités fournies localement et sur certains problèmes de Qds.

L'exemple suivant d'une application de télé-éducation clarifie le principe de la gestion coopérative de Qds (voir figure 4.1). Dans cette application, nous disposons d'un professeur qui donne un cours, en direct, à des étudiants qui sont situés dans des régions différentes et lointaines. Le système doit assurer la transmission d'informations de type vidéo, audio, et texte sous forme de plusieurs variantes de qualités qui diffèrent selon les caractéristiques de l'image (couleur ou noir et blanc), de la résolution (faible ou élevée), de l'audio (CD ou téléphone)...etc. Le professeur constitue l'émetteur puisqu'il représente la source d'information et les étudiants sont les récepteurs.

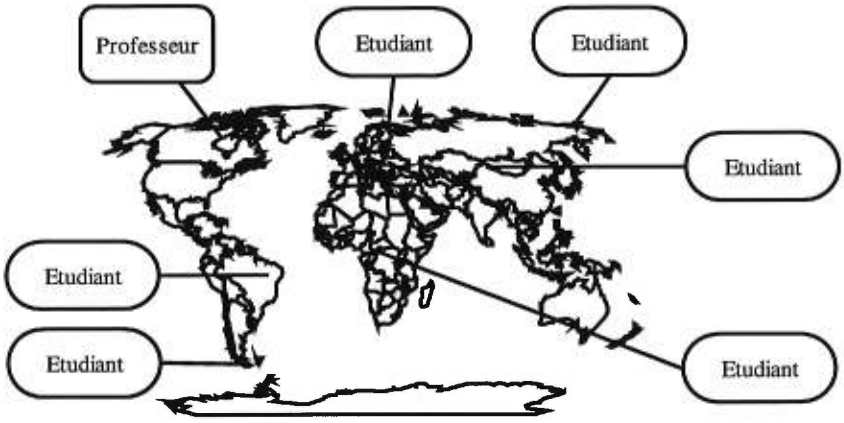


Figure 4.1. Exemple d'une application de télé-éducation

Dans cette application, la source diffuse les flots d'informations sous forme des trois qualités suivantes:

- Une qualité bonne Q3 avec des images en couleur, une résolution élevée et de l'audio de qualité CD.
- Une qualité moyenne Q2 avec des images en couleur, une résolution faible et de l'audio de qualité téléphone.
- Une qualité faible Q1 avec des images en noir et blanc, une résolution faible et de l'audio de qualité téléphone.

La figure 4.2 présente l'arbre de diffusion relatif à cette application. L'agent commutateur C3 connaît les trois agents de niveau inférieur R1, R2 et R3 aussi bien que l'agent de niveau supérieur C1. Il connaît aussi toutes les informations concernant les ressources disponibles au niveau de son commutateur, les coûts nécessaires pour les réserver et les différentes variantes de Qds disponibles pour cette application.

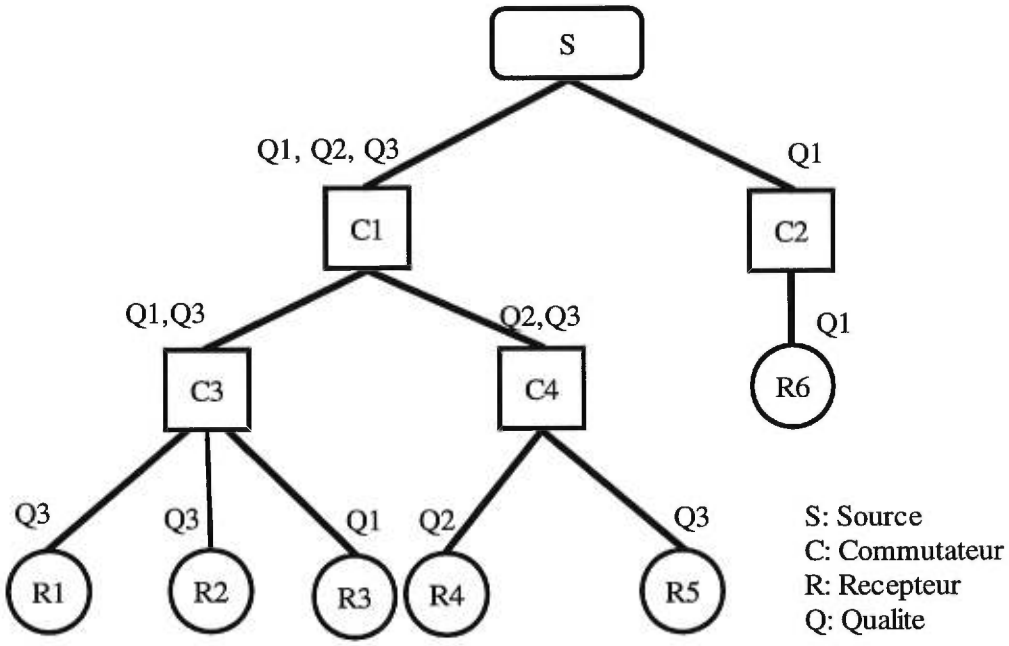


Figure 4.2. Diffusion de flots d'une application multimédia dans un réseau

4-2 L'approche multi-agents pour la gestion coopérative de la Qds

L'architecture de la gestion coopérative de la Qds dans les systèmes distribués est basée sur le concept d'agents [Haf-97b]. Un agent est installé dans chaque composant du système tel que les serveurs, les commutateurs et les machines hôtes. La fonction principale de ces agents est d'implémenter le protocole de CQosM dans n'importe quel système multimédia qui nécessite une gestion de la Qds comme la négociation, la renégociation et l'adaptation de la Qds.

Un agent est un programme capable de prendre ses propres décisions et dont l'objectif est d'atteindre certains buts [Jen-96,Woo-94]. Il est doté d'un certain niveau d'intelligence qui peut varier de simples règles prédéfinies jusqu'aux moteurs d'inférences d'auto-apprentissage à intelligence artificielle. Il agit généralement pour le compte d'un utilisateur ou un processus permettant une automatisation des tâches. Les agents peuvent opérer d'une manière autonome et peuvent communiquer avec l'utilisateur, les ressources système et/ou les autres agents si nécessaire pour réaliser leurs tâches. Les différentes caractéristiques des agents sont les suivantes:

i. *L'autonomie* : les agents opèrent sans l'intervention directe de l'humain et ils ont un certain contrôle sur leurs actions et leurs états internes ;

ii. *L'habilité sociale* : les agents interagissent avec d'autres agents via un certain langage de communication d'agents ;

iii. *La réactivité* : les agents observent leurs environnements, et réagissent aux différents changements de ces environnements au moment opportun ;

iv. *La proactivité* : les agents ne réagissent pas uniquement à la suite d'un événement, mais ils sont capables d'entreprendre des initiatives pour éviter un problème donné.

v. *La mobilité* : la capacité d'un agent de se déplacer dans un réseau ;

vi. *La véracité* : l'hypothèse que les agents ne vont pas transmettre des informations incorrectes intentionnellement ;

vii. *Le bénévolat* : l'hypothèse que les agents n'ont pas des buts conflictuels et que chaque agent va toujours essayer de réaliser ses buts ; et

viii. *La rationalité* : l'hypothèse qu'un agent va essayer de réaliser ses buts et qu'il va agir convenablement.

Dans l'architecture multi-agent pour la gestion coopérative de la Qds, les agents utilisés réalisent un certain nombre d'actions dont la supervision continuelle de l'état du réseau et l'utilisation de cette connaissance pour la prise de décisions. Les agents se caractérisent par les différentes caractéristiques décrites ci-dessus à l'exception de la mobilité et la proactivité puisqu'ils sont fixes et réactifs. Le rôle principal des agents intelligents de ce système distribué est de soulager l'opérateur du réseau de la gestion de la qualité de service et de maintenir les qualités désirées par les utilisateurs tout en essayant d'optimiser les ressources et d'augmenter le profit du système.

La figure 4.3 montre l'architecture multi-agents pour la gestion coopérative de la Qds de la figure 4.2.

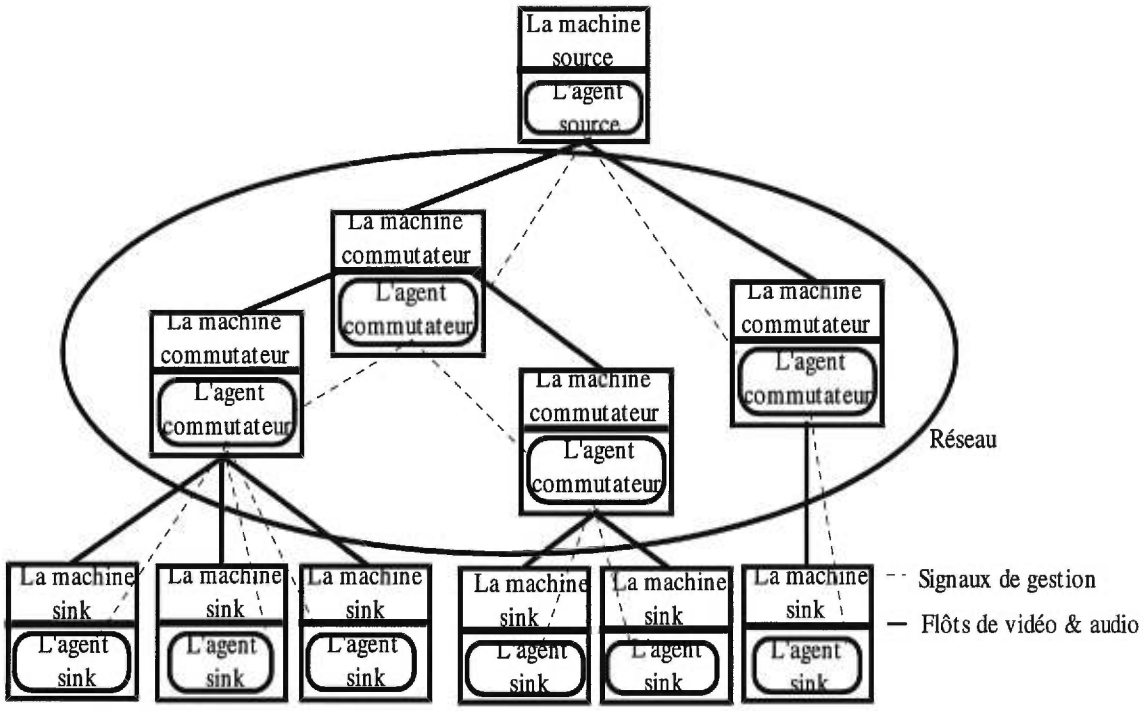


Figure 4.3. Architecture multi-agents pour la gestion coopérative de la Qds

Ce système distribué est composé de trois types d'agent: l'agent source, l'agent commutateur et l'agent Sink et chaque agent a un rôle spécifique pour la gestion coopérative de la qualité de service. Ces agents communiquent entre eux pour échanger des informations. Cette communication n'est pas réalisée, d'une manière directe, entre tous les agents du système. Les agents commutateurs peuvent communiquer avec leurs voisins de niveau supérieur et inférieur. Alors que les agents émetteurs communiquent uniquement avec les agents commutateurs de niveau inférieur et les agents récepteurs communiquent avec les agents de niveau supérieur. La structure de la communication est hiérarchique puisque elle dépend de l'arbre de diffusion. Cet arbre de diffusion impose une hiérarchie entre les différents agents du système.

(i) L'agent Source

L'agent source permet de soulager la source des opérations de gestion de la Qds. Il se charge d'attribuer et de maintenir les qualités demandées par les agents voisins de niveau inférieur. Par exemple, il demande à la source de transmettre les flots d'informations avec une certaine qualité. Dans les applications qui se caractérisent par la faiblesse des fréquences de connexion et de déconnexion des usagers, l'agent source demande à la source de transmettre uniquement les flots d'information concernant les Qds spécifiées par les usagers au lieu de transmettre inutilement toutes les Qds disponibles.

Un agent Source, installé dans une source d'information, représente la racine de l'arbre de diffusion associé à une application multimédia. L'identificateur de cet arbre, qui est unique dans tout le système, permet d'identifier également l'application. Chaque composant du système, appartenant à cet arbre, est identifié par un identificateur unique. Un agent Source peut être utilisé par plusieurs applications en même temps. Les qualités de service offerts par la source diffèrent d'une application à une autre. À chaque création d'un nouvel arbre ou modification d'un arbre de diffusion déjà existent, l'agent source met à jour la table *So_Tree_List* qui est constituée des éléments suivants pour chaque arbre de diffusion donnée :

1. L'identificateur de l'arbre : *Tree_Id* ;
2. La liste des Qds disponibles au niveau de la source : *A_List_Qos ;*
3. La liste des coûts associés aux différentes Qds: *Cout ;*
4. La liste des Qds fournies actuellement par la source : *C_List_Qos* avec *C_List_Qos* $\subset$ *A_List_Qos*;
5. Les identificateurs des agents de niveau inférieur : *D_Agent_Id*, avec les Qds fournies à chacun de ces agents : *List_Qos*.

(ii) L'agent Commutateur

L'agent Commutateur se charge de toutes les opérations de la gestion de la Qds au niveau du commutateur . Il permet ainsi de soulager le commutateur de ces tâches. À chaque création ou modification d'un arbre de diffusion, cet agent met à jour la table *R_Tree_List.* Cette table est constituée, en plus, des éléments décrits dans l'agent source, des éléments suivants :

1. L'identificateur de l'agent de niveau supérieur : *U_Agent_Id.*

2. La liste des Qds que l'agent commutateur ne peut pas recevoir à cause d'une défaillance au niveau du commutateur ou des liens qui lui sont connectés et donc, ces qualités ne peuvent pas être transmises aux voisins de niveau inférieur : *V_List_Qos1* .

(iii) L'agent Sink

L'agent Sink s'occupe de gérer la Qds avec l'usager et avec l'agent TM de niveau supérieur. Il maintient la mise à jour de la table *Si_Tree_List* qui contient les mêmes informations que l'agent commutateur à l'exception de *C_List_Qos* puisque le Sink supporte une seule Qds à la fois et (*D_Agent_Id, List_Qos*) puisque l'agent Sink ne possède pas d'agents de niveau inférieur. L'agent Sink dispose également des éléments suivants dans la table *Si_Tree_List:*

1. La qualité de service utilisé par l'usager : *C_Qos.*

2. Le prix payé par l'usager pour recevoir la Qds *C_Qos* : *Prix_Qos*

4-3 Les fonctions de la gestion coopérative de la Qds

Pour assurer la satisfaction des utilisateurs, une gestion de Qds s'avère nécessaire puisqu'elle permet de réaliser les fonctionnalités suivantes: la négociation de la Qds, la renégociation de la Qds et la reconfiguration du réseau.

4-3-1 La négociation de la Qds

Le protocole de négociation est exécuté à chaque fois qu'un utilisateur se connecte à une application donnée. L'algorithme de routage de diffusion va permettre d'envoyer la requête de connexion à un commutateur connecté à cette application. L'agent de ce commutateur va envoyer à cet usager toutes les informations concernant les qualités disponibles et les prix associés. Le prix d'une qualité donnée diffère d'une région à une autre, c'est pour cela qu'il n'y a pas un agent central qui produit ce type d'informations. Une fois que l'usager sélectionne la qualité désirée, les connexions sont établies à travers le réseau et la mise à jour des données est réalisée dans les différents agents de Qds concernés par la connexion de ce nouvel usager. La réservation des ressources est réalisée tout au long du trajet du flot afin de maintenir le service offert. Le protocole de réservation des ressources RSVP (Resource ReSerVation Protocol) [Zha-93] est utilisé. La négociation a pour rôle de trouver un

arrangement entre les préférences de l'usager et les ressources disponibles du système afin de lui satisfaire ses besoins sans entraîner une dégradation de la Qds pour les autres usagers.

4-3-2 La renégociation de la Qds

La renégociation de la Qds peut être initialisée par l'usager ou par le système. Dans le premier cas, l'usager a recours à la renégociation pour changer la qualité reçue. Si par exemple, il reçoit des images en noir et blanc, il veut peut être transiter vers une meilleure qualité en vue de recevoir des images en couleur. L'usager peut aussi changer pour une qualité moins bonne afin de diminuer le prix à payer. La renégociation est très importante dans le cas où il est préférable pour l'usager de changer de Qds tout au long de la session, puisque le fait de garder une seule qualité depuis le début de la session et jusqu'à la fin peut ne pas lui convenir. Le deuxième cas consiste à déclencher une renégociation à la demande du système. Ceci peut arriver lors d'une violation de la qualité de service à cause de la surcharge du système ou de la défaillance d'un composant du système. Puisque le système ne peut plus maintenir la Qds négociée, une renégociation est réalisée afin que l'usager transite vers une autre Qds. La renégociation est aussi utilisée en vue d'optimiser les ressources du système. Cette optimisation des ressources permet à d'autres usagers de se connecter à d'autres applications et ainsi augmenter le profit du système.

4-3-3 La reconfiguration du réseau

Le rôle de cette fonction est de maintenir la Qds accordée à l'usager. Elle consiste à s'assurer que le système fournit ses services en respectant les valeurs demandées par l'usager lors de la négociation. L'adaptation de la Qds peut être réalisée par la reconfiguration de tout l'arbre de diffusion, ce qui est extrêmement coûteux ou par la reconfiguration partielle de la partie défaillante de l'arbre de diffusion. C'est pour cela qu'il faut utiliser un mécanisme de localisation et d'identification de la partie responsable de la violation de la Qds.

Dans le cas où c'est impossible d'assurer la maintenance de la Qds, lors d'une défaillance ou une surcharge localisée du réseau, l'usager pourra subir une dégradation de la qualité reçue et une renégociation de Qds sera nécessaire pour choisir une autre qualité plus appropriée.

4-4 Description des signaux

Afin que la communication entre les différents agents soit possible, plusieurs messages doivent être échangés entre ces agents. Ceci à pour but de transmettre certaines informations qui permettent aux agents de réagir et d'entreprendre certaines actions. Si par exemple, un usager veut se connecter à une application donnée, l'agent Sink reçoit cette demande. Puis, il présente à l'usager l'ensemble des qualités de service disponibles ainsi que les prix à payer pour les recevoir. L'usager choisit, ensuite, une qualité donnée et ce choix est transmis à l'agent Sink. Ce dernier informe l'agent TM de niveau supérieur afin qu'il lui fait parvenir la qualité désirée par l'usager. Ceci est traduit par un ensemble de messages échangés entre l'usager et les agents de Qds concernés afin de réaliser l'opération demandée. La figure 4.4 montre les messages effectivement échangés afin de réaliser la connexion de l'usager.

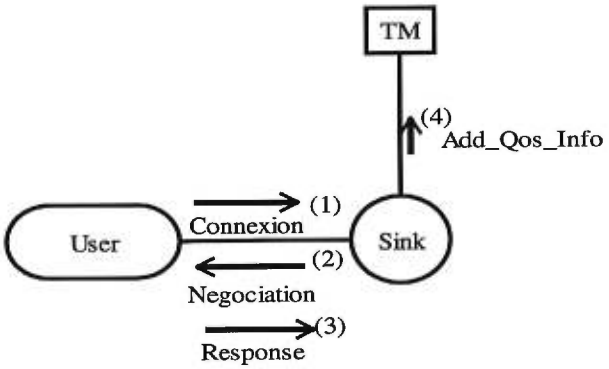


Figure 4.4. La connexion d'un nouvel usager

Nous allons présenter dans la section suivante l'ensemble des messages échangés entre l'usager et les différents agents de la qualité de service. Cet échange est réalisé de la manière suivante :

1- Un usager d'une application donnée échange avec l'agent Sink, à travers une interface, plusieurs messages tels que :

- Connexion (Tree_Id) est un signal envoyé par l'usager vers l'agent Sink afin de se connecter à une application donnée. Celle-ci est identifié par l'identificateur de l'arbre de diffusion *Tree_Id.*
- Negociation (Tree_Id, List_Qos) est envoyé par l'agent Sink vers l'usager en vue de lui informer de l'ensemble de Qds disponibles ainsi que les prix associés.
- Response (Tree_Id, Qos) indique la Qds désirée par l'usager.

- Renegociation_User (Tree_Id) est envoyé par l'usager vers l'agent Sink pour pouvoir faire une nouvelle renégociation et choisir une autre qualité de service.
- Deconnexion (Tree_Id) est un signal envoyé par l'agent Sink vers l'usager pour lui indiquer la fin de la session de travail pour cette application.

2-Les différents agents du protocole de la gestion coopérative de la Qds communiquent entre eux en envoyant des messages. Les différents messages sont les suivants:

- Add_Qos_Info (Tree_Id, Sender_Id, Receiver_Id, Qos) est un signal envoyé par l'agent identifié par *Sender_Id* vers l'agent de niveau supérieur identifié par *Receiver_Id.* La qualité Qos indique la qualité de service que l'agent *Sender_Id* veut recevoir.
- Remove_Qos (Tree_Id, Sender_Id, Receiver_Id, Qos) est un signal envoyé pour que l'agent *Sender_Id* ne reçoit plus de l'agent de niveau supérieur *Receiver_Id* la qualité *Qos.*
- Viol (Tree_Id, Sender_Id, Receiver_Id, List_Qos) est envoyé pour indiquer à l'agent *Receiver_Id* la liste des Qds *List_Qos* qui ont subi une dégradation intolérable.
- Not_Ok (Tree_Id, Sender_Id, Receiver_Id, Qos) est envoyé pour informer l'agent *Receiver_Id* que la *Qos* ne peut pas ou ne peut plus être fournie.
- Persuade (Tree_Id, Sender_Id, Receiver_Id, Qos, Prix) est envoyé pour indiquer la qualité *Qos* que l'agent *Sender_Id* préfère dorénavant envoyer à l'agent de niveau inférieur *Receiver_Id.* Le nouveau prix, associé à cette qualité *Qos*, est spécifié par *Prix.* Ce nouveau prix est inférieur au prix normal afin de persuader les usagers et les encourager à recevoir la qualité *Qos.*
- Arret_Persuade (Tree_Id, Sender_Id, Receiver_Id, Qos, Prix) est envoyé pour indiquer que la persuasion est terminée et que les agents de niveau inférieur qui bénéficiaient de la qualité de persuasion doivent payer, dorénavant, le prix régulier.

3-Les agents commutateurs (Tree_Manager) et les agents Sink du protocole de la gestion cooperative de la qualité de service échangent également des messages avec l'agent de routage à travers une interface. Ces messages sont les suivants :

- Ask_Routing est un signal envoyé par les agents Sink et Tree_Manager au protocole de routage Prot_Routing afin de savoir la disponibilité ou non d'un autre chemin de routage pour la transmission de Qds.

- Routing (valeur) est un signal émis par le protocole de routage Prot_Routing pour répondre à la demande de Ask_Routing. La réponse est soit positive lors de la disponibilité d'un chemin libre soit négative dans le cas contraire.

4-5 Description des opérations du protocole CQosM

L'échange d'informations entre les différents agents permet de réaliser les opérations de la gestion coopérative de la qualité de service [Haf-97b]. Les différentes opérations réalisées sont décrites dans les sous-sections qui suivent.

4-5-1 Connexion d'un nouvel usager

Pour se connecter à une application donnée, l'usager envoie un signal *Connexion* à l'agent Sink en spécifiant l'application MM. À la réception de ce message, l'agent Sink lui transmet toutes les informations concernant les qualités disponibles et qui peuvent être supportées localement au niveau du Sink, ainsi que les prix correspondants. Ces informations sont transmises à travers le signal *Négociation*. À la réception de ce signal, l'usager choisit une qualité donnée parmi les qualités disponibles selon ses préférences, son budget et les capacités de sa machine.

La qualité choisie par l'utilisateur est envoyée à l'agent Sink par le signal *Response*. À la réception de ce signal, si l'utilisateur n'a choisi aucune qualité, alors il veut quitter l'application et un signal *Déconnexion* lui est envoyé par l'agent Sink. Dans le cas contraire, un signal *Add_Qos_Info* est envoyé à l'agent commutateur de niveau supérieur en spécifiant la qualité que l'usager veut recevoir. La figure 4.4 montre l'enchaînement chronologique des différents signaux émis et reçus au début de la connexion d'un nouvel usager et dans le cas où l'usager choisit une Qds donnée. À la réception du signal *Add_Qos_Info*, l'agent TM exécute les tâches suivantes :

i. Si $Qos \in C\_List\_Qos$, alors l'agent TM met à jour la variable *(D_Agent_Id, List_Qos)* de *R_Tree_List* pour que l'agent émetteur du signal *Add_Qos_Info* reçoit la qualité spécifiée par *Qos* (voir cas (a) de la figure 4.5). Un test de persuasion est aussi réalisé pour voir s'il est rentable ou non d'envoyer un signal *Persuade* aux agents de niveau inférieur.

ii. Si $Qos \notin C\_List\_Qos$, alors si cette *Qos* ne peut pas être supportée localement, un signal *Not_Ok* est envoyé à l'agent qui a demandé de recevoir cette qualité pour l'informer de

l'impossibilité de lui fournir cette qualité actuellement (voir cas (b) de la figure 4.5). Sinon, l'agent TM modifie la variable *(D_Agent_Id, List_Qos)* de *R_Tree_List* pour spécifier que l'agent de niveau inférieur va recevoir la qualité demandée. La variable *C_List_Qos* est aussi modifiée puisque cette qualité *Qos* sera fournie par cet agent et un signal *Add_Qos_Info* est envoyé à l'agent de niveau supérieur qui peut être un agent TM ou l'agent Source. Ces dernières modifications sont réalisées avant de recevoir effectivement les flots d'informations correspondants à la qualité demandée puisque dans la majorité des cas, cette opération est effectuée avec succès. Si le dernier signal *Add_Qos_Info* est reçu par un agent TM, le même traitement présenté ci-dessus est réalisé.

À la réception par l'agent Source du signal Add_Qos_Info envoyé par l'agent TM, les deux cas suivants sont considérés :

a. Si Qos $\in$ *A_List_Qos*, alors:

   i. Modification de la variable *C_List_Qos* dans le cas où cette qualité n'a pas été fournie par la source.

   ii. Modification de la variable *(D_Agent_Id, List_Qos)* de *So_Tree_List*, en spécifiant que l'agent TM de niveau inférieur, qui a émis le signal *Add_Qos_Info* reçoit la qualité *Qos* (voir cas (c) de la figure 4.5).

   iii. Réalisation d'un test de persuasion pour voir s'il est rentable ou non pour le système d'envoyer un signal *Persuade* aux agents du niveau inférieur.

b. Si *Qos* $\notin$ *A_List_Qos, alors,* le signal *Not_Ok* est envoyé à l'agent TM pour lui indiquer que la *Qds* demandée n'est pas disponible au niveau de la source et donc elle ne peut pas être fournie (voir cas (d) de la figure 4.5).

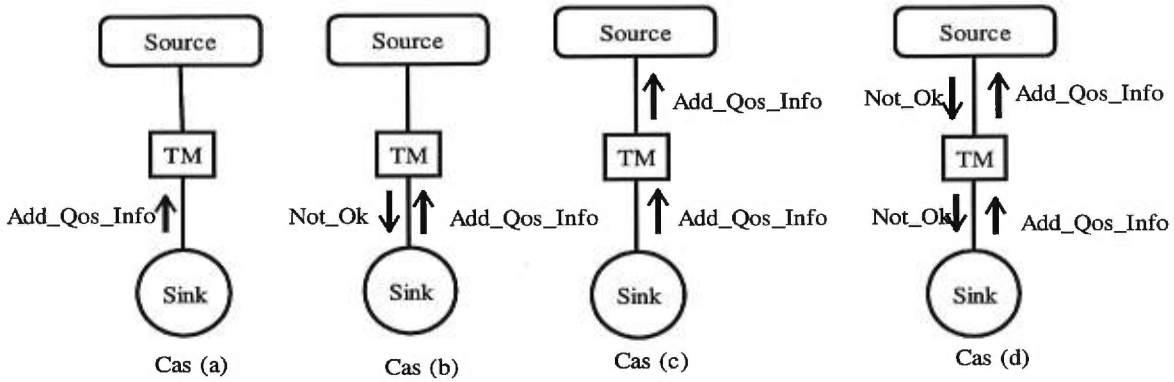


Figure 4.5. Les différents cas d'une demande de connexion de l'usager

Lorsque l'agent TM reçoit le signal *Not_Ok* provenant de l'agent de niveau supérieur, il met à jour *C_List_Qos* puisque la qualité *Qos* spécifiée dans le signal ne peut pas lui être transmise à cause de la non disponibilité ou la violation de cette qualité (voir section 4-5-2 dans ce chapitre). Par conséquent, l'agent TM modifie la variable *(D_Agent_Id, List_Qos)* de *R_Tree_List* pour l'application traitée et envoie un signal *Not_Ok* pour l'agent de niveau inférieur qui a demandé cette Qds pour lui informer de la nouvelle situation.

La réception de ce signal au niveau de l'agent Sink permet de l'informer que la qualité désirée par l'usager ne peut pas être reçue. L'agent Sink réalise une renégociation avec l'usager afin qu'il transite vers une autre Qds en lui présentant les qualités disponibles au niveau de la source et qui peuvent être supportées localement au niveau du Sink, ainsi que les prix proposés pour les recevoir.

Pendant une session de travail, si un usager veut faire une renégociation de la Qds pour changer la qualité reçue, il envoie un signal *Renegociation_User* à l'agent Sink. À la réception de ce signal, l'agent Sink lui envoie le signal *Negociation* en lui spécifiant les différentes Qds disponibles et qu'il peut supporter localement ainsi que les prix exigés pour les recevoir. L'usager fait son choix et lui répond à travers le signal *Response*. Le même traitement décrit ci-dessus est réalisé.

4-5-2 Violation d'une Qds et reconfiguration du réseau

Chaque agent du système prend, d'une manière régulière, des mesures sur les paramètres de la Qds. Il compare ces paramètres avec les valeurs de Qds négociées pour chaque qualité de service. La non conformité des ces valeurs indique que la qualité *Qos* a subi une dégradation. L'agent responsable de cette dégradation doit prendre des mesures pour restituer la qualité demandée par l'usager, ou pour initier une renégociation avec lui. Ce mécanisme de détection de violation de la Qds est implanté dans les différents agents du système : Sink, TM et Source.

À la détection de la dégradation de la Qds au niveau de l'agent Sink, celui-ci vérifie les deux cas suivants :

a) Si le Sink est responsable de la violation de la Qds, alors:

   i. Un signal *Remove_Qos* est envoyé à l'agent TM de niveau supérieur pour demander l'arrêt de transmission des flots de la qualité *Qos*. Cette qualité ne peut pas être supportée par le Sink pour une certaine période et si l'agent Sink garde cette qualité, l'usager de cette qualité va certainement constater une dégradation de la qualité reçue.

   ii. La qualité Qos est ajoutée dans la variable *V_List_Qos1*. Cette variable représente l'ensemble des Qds qui ne peuvent pas être fournies à ce niveau pour un certain temps. Cette variable est utilisée afin de savoir les qualités qui ne peuvent pas être reçues par l'agent Sink en question. Par exemple, si un usager connecté à travers ce Sink demande cette qualité *Qos*, l'agent Sink ne pourra pas satisfaire sa demande. La variable *V_List_Qos1* permet de supprimer cette qualité de l'ensemble des qualités disponibles jusqu'à ce que le Sink soit capable à nouveau de recevoir cette qualité sans engendrer aucune violation de la Qds. L'utilisation de cette variable permet d'éviter l'envoi et la réception des signaux de *Add_Qos_Info*, *Remove_Qos* et *Not_Ok* inutilement.

   iii. Une renégociation est réalisée avec l'usager afin qu'il choisisse une autre qualité puisque la qualité fournie actuellement a subi une dégradation et ne peut plus être maintenue (voir cas (a) de la figure 4.6).

b) Si le Sink n'est pas responsable de la violation, alors il envoie un signal *Viol* à l'agent TM de niveau supérieur (voir cas (b) de la figure 4.6).

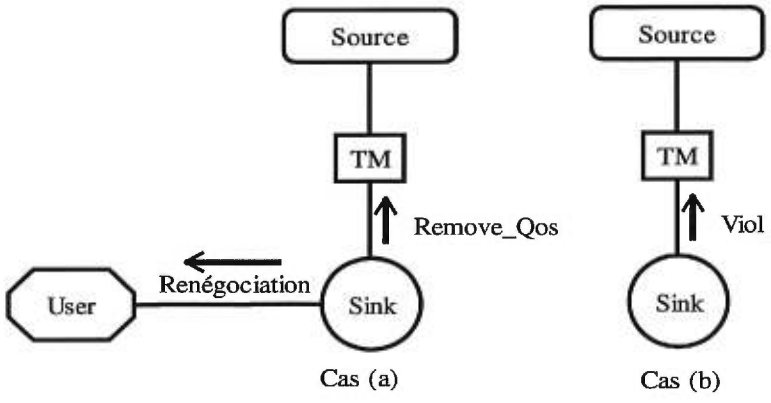


Figure 4.6. Détection de la dégradation de la qualité de service au niveau Sink

Nous venons de voir que si l'agent Sink détecte une dégradation de la Qds et qu'il n'est pas responsable, alors il émis un message *Viol* à l'agent TM de niveau supérieur en spécifiant la qualité violée. À la réception de ce signal, l'agent TM attend une certaine durée pour voir si tous les agents de niveau inférieur vont envoyer le même message. Dans une telle situation, l'agent TM envoie un signal *Viol* à l'agent de niveau supérieur en spécifiant les qualités violées. Dans ce cas, la cause de la violation des qualités est situé dans le chemin parcouru pour transmettre les flots d'information à ce TM. Dans le cas contraire, c'est le lien qui se trouve entre le récepteur du message *Viol* et son voisin, transmetteur du message, qui est défectueux et qui n'arrive pas à transmettre cette qualité. Un signal *Solve* est envoyé à chacun des agents de niveau inférieur qui ont subi une dégradation de Qds et qui ont envoyé le signal *Viol* à cet agent TM.

La réception du signal *Solve* indique que le lien qui se trouve entre l'agent émetteur de ce signal et l'agent récepteur est la cause de la violation de la qualité *Qos* et que ce lien ne permet pas de transmettre le flot d'information correspondant à cette qualité *Qos*. À la réception de ce signal par l'agent Sink, il contacte l'agent de routage pour voir s'il existe un autre chemin de secours pour la transmission de la qualité violée *Qos*. Dans le cas d'une réponse positive, la transmission est réalisée. Nous supposons dans ce cas que les traitements liés au changement de la trajectoire et à la modification de l'arbre de diffusion sont réalisés sans aucun problème. Dans le cas contraire, l'agent Sink envoie un signal *Remove_Qos* à l'agent TM de niveau supérieur pour procéder aux changements nécessaires. Une renégociation est réalisée avec l'usager en lui envoyant les qualités fournies par la source et qui peuvent être supportées localement par le Sink, ainsi que les prix exigés pour les recevoir.

Exemple 1 :

La figure 4.7 montre un exemple où les récepteurs R1 et R2 de la figure 4.2, qui recevaient la qualité Q3, ont subi une violation de Qds. Les récepteurs R1 et R2 ont envoyé un signal *Viol* à l'agent commutateur de niveau supérieur C3. Celui-ci n'a pas reçu le signal *Viol* du récepteur R3 après avoir attendu une certaine durée et par conséquent, il envoie un signal *Solve* aux agents de niveau inférieur R1 et R2. Ces deux agents ont contacté l'agent de routage et dans ce cas, ils ont transité vers une autre qualité Q2 puisqu'un autre chemin de secours

n'était pas disponible.

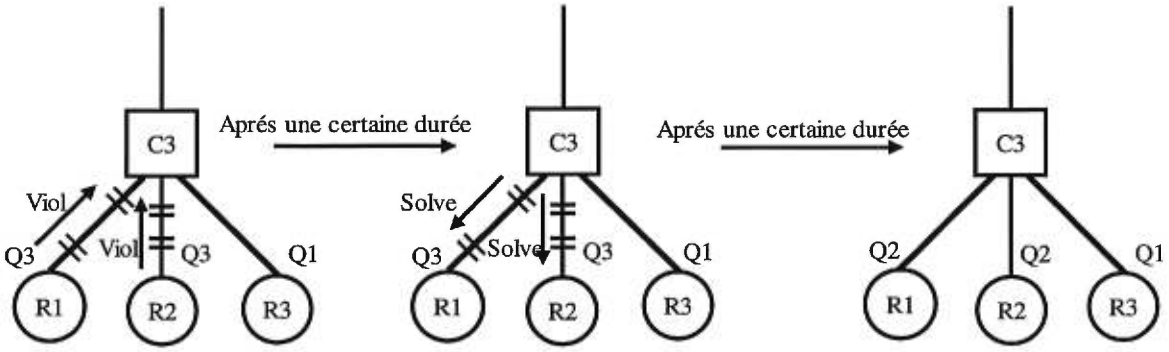


Figure 4.7. Exemple d'une violation d'une Qds avec chemin libre non disponible

Au niveau d'un agent Tree_Manager (TM), celui-ci peut envoyer un signal *Viol* à l'agent de niveau supérieur dans deux cas. Dans le premier cas, l'agent TM reçoit le signal *Viol* de tous les agents de niveau inférieur. Dans le deuxième cas, l'agent TM détecte une dégradation de la Qds et il n'est pas responsable (voir cas (b) de la figure 4.8). Si l'agent TM est responsable de cette dégradation, alors il réalise les actions suivantes (voir cas (a) de la figure 4.8):

i. Supprimer cette qualité dégradée *Qos* de *C_List_Qos*.
ii. Ajouter cette qualité dégradée *Qos* dans la liste *V_List_Qos1*. Cette variable contient la liste des Qds qui ne peuvent pas être supportées localement.
iii. Envoyer le signal *Remove_Qos* pour demander l'arrêt de transmission des flots de la qualité dégradée. Ce signal est envoyé à l'agent de niveau supérieur qui peut être soit un agent TM soit l'agent Source.
iv. Modifier la variable *(D_Agent_Id, List_Qos)* de l'entrée *R_Tree_List* qui correspond au *Tree_Id* concerné pour que tous les agents de niveau inférieur, qui utilisent cette qualité, ne la reçoivent plus. Un signal *Not_Ok* est envoyé à chacun de ces agents afin de les informer de l'impossibilité de recevoir, actuellement, cette qualité.

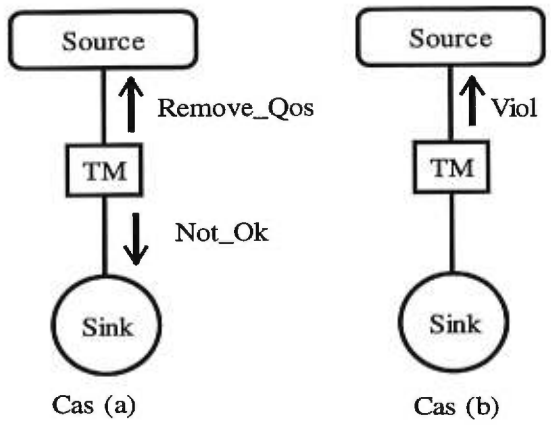


Figure 4.8. Détection de la dégradation de la qualité de service au niveau TM

Nous venons de voir que l'agent TM envoie, dans certains cas, le signal *Viol* à l'agent de niveau supérieur. Si cet agent est un TM, alors le même traitement décrit ci-dessus est réalisé. Si c'est l'agent Source qui reçoit le signal *Viol*, alors il procède de la même manière par l'attente d'une certaine durée pour voir si les autres agents de niveau inférieur vont émettre un signal similaire. Si c'est le cas, la modification des variables *A_List_Qos* et *C_List_Qos* est nécessaire puisque la source est responsable de cette violation de la Qds et donc, les qualités violées ne sont plus fournies par la source pendant un certain temps. En plus, l'agent source modifie l'entrée *(D_Agent_Id, List_Qos)* de la table *So_Tree_List*. Les agents qui ont envoyé ce signal *Viol* ne vont plus recevoir les qualités violées pour une certaine durée et un signal *Not_Ok* est émis à chacun d'eux pour leur indiquer que la qualité violée n'est plus transmise. Dans le cas où uniquement certains des agents de niveau inférieur ont envoyé le signal *Viol* à l'agent source, il envoie un signal *Solve* à chacun de ces agents.

À la réception de ce signal par l'agent TM, il contacte l'agent du routage de la même manière décrite ci-dessus. Si un autre chemin de secours n'est pas disponible, l'agent TM envoie un signal *Remove_Qos* à l'agent de niveau supérieur pour chaque qualité violée. Il modifie *C_List_Qos* puisque la qualité violée n'est plus fournie, ainsi que l'entrée *(D_Agent_Id, List_Qos)* de la table *R_Tree_List* puisque les agents de niveau inférieur ne vont plus recevoir cette qualité pour une certaine durée et pour les informer, un signal *Not_Ok* est envoyé afin qu'ils transitent vers une autre qualité de service.

Exemple 2 :

La figure 4.9 traite le même cas de la figure 4.7, mais dans ce cas, l'agent commutateur C3 reçoit le signal *Viol* de tous les récepteurs R1, R2 et R3 et donc, il doit envoyer à l'agent de niveau supérieur C1 un signal *Viol*. Cet agent attend un certain moment pour voir s'il va recevoir un message similaire de l'agent C4. Si celui-ci n'a détecté aucune dégradation des qualités fournies, alors l'agent C1 répond à l'agent C3 par l'envoi du message *Solve* qui signifie que le problème de violation se situe au niveau du lien entre C1 et C3. À la réception de ce signal, l'agent C3 contacte le protocole de routage. Si celui-ci trouve un autre chemin de secours disponible pour la transmission des données, alors les Qds demandés par les usagers sont maintenues. Ce nouveau chemin nécessite la modification de l'arbre de diffusion puisque l'agent commutateur C1 possède actuellement deux agents de niveau inférieur C4 et C5. Ce dernier possède l'agent C3 comme voisin de niveau inférieur. Les autres parties de l'arbre de diffusion ne vont subir aucune modification.

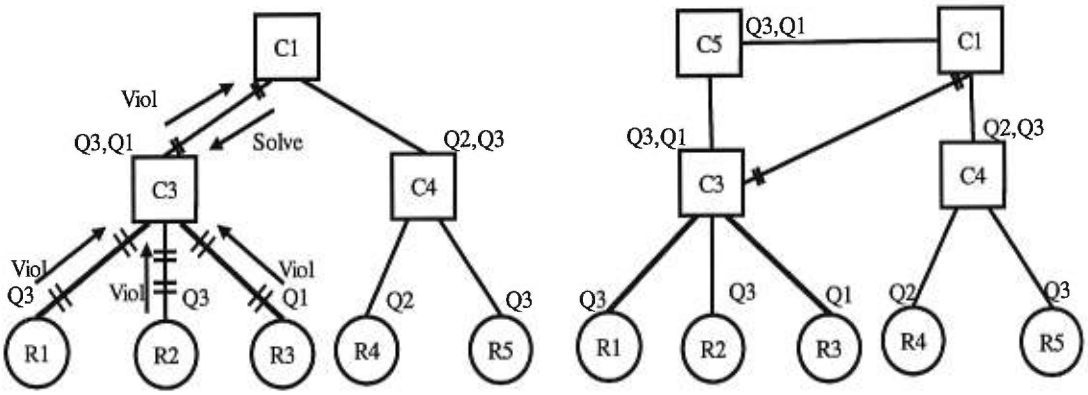


Figure 4.9. Exemple d'une violation d'une Qds avec chemin libre disponible

Exemple 3 :

Un autre exemple d'une violation de la Qds est présenté. Dans ce cas, la qualité fournie au niveau du récepteur R6 est dégradée et le récepteur R6 n'en est pas responsable. Il envoie un signal *Viol* à l'agent de niveau supérieur C2 (voir figure 4.10). Etant donné que cet agent n'a qu'un seul agent de niveau inférieur, il va envoyer un message *Viol* vers la source. Si ce dernier ne reçoit pas un message similaire de l'agent C1, il lui répond par l'envoi du message *Solve* et donc, par ce mécanisme d'identification de la partie responsable de la violation, on conclut que c'est le lien se trouvant entre l'agent C2 et l'agent source qui souffre d'une

défaillance et qui n'arrive pas à transmettre la qualité demandée par le récepteur R6. Or, il est fortement possible que le lien qui se trouve entre l'agent C2 et le récepteur R6 soit responsable de cette dégradation de la qualité. Dans ce cas, même si l'agent C2 contacte le protocole de routage pour trouver un autre chemin de secours pour la transmission des données, la qualité Q1 ne pourra pas être maintenue. Ainsi, pour les arbres de diffusion qui disposent d'un seul agent de niveau inférieur, l'identification précise et exacte de la partie défaillante devient difficile. C'est pour cela qu'il faut disposer d'un mécanisme pour vérifier les valeurs de Qds mesurées avec les valeurs négociées des informations reçues de l'agent de niveau supérieur. Dans l'exemple précédent, il faut vérifier ces valeurs au niveau de l'agent C2. Dans le cas de la non conformité des valeurs de Qds mesurées avec les valeurs négociées, il faut envoyer le message *Viol* vers l'agent source. Dans le cas contraire, l'agent C2 a suffisamment de preuve pour savoir que la violation de Qds est dû au lien se trouvant entre C2 et R6, et un signal *Solve* sera destiné au récepteur R6 afin qu'il contacte le protocole de routage pour voir la disponibilité d'un autre chemin de secours pour transmettre le flot d'informations correspondant à la qualité Q1.

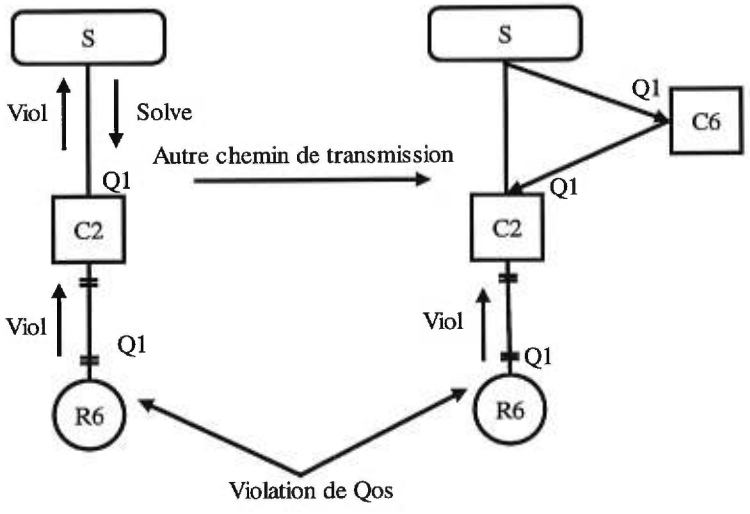


Figure 4.10. Détection de la violation de Qds

Ainsi, la gestion coopérative de Qds permet de localiser la partie responsable de la violation et par la suite, une reconfiguration partielle de l'arbre de diffusion afin d'assurer le bon acheminement des flots d'informations.

Nous avons vu, au début de cette section, que les agents TM et Sink peuvent détecter une dégradation de la Qds. De la même manière, l'agent Source peut détecter une violation de la qualité *Qos*. Dans ce cas, l'agent source procède à ces modifications :

i. Modifier la variable *C_List_Qos* puisque la qualité dégradée *Qos* ne peut plus être fournie pour une certaine période.
ii. Supprimer la qualité *Qos* de *A_List_Qos* puisque cette qualité n'est plus disponible au niveau de la source.
iii. Modifier la variable *(D_Agent_Id, List_Qos)* de *So_Tree_List* pour spécifier que les agents qui utilisaient cette qualité ne peuvent plus la recevoir.
iv. Envoyer le signal *Not_Ok* à ces agents pour les informer de la nouvelle situation et pour qu'ils choisissent une autre qualité de service.

Après avoir traité la violation de la Qds par les différents agents, nous allons présenter le mécanisme de la récupération des Qds (Recovery). En effet, lors d'une violation de la Qds, certaines qualités ne sont plus supportées par les composants du système. Dans le cas réel d'un réseau, si certains composants ne peuvent plus recevoir et transmettre certaines qualités, le gestionnaire du réseau s'occupe de changer certaines caractéristiques du réseau et de réparer certaines défaillances afin qu'ils peuvent recevoir à nouveau toutes les qualités disponibles.

Nouvelle approche pour la reconfiguration du réseau

Une autre approche a été aussi proposée pour identifier avec précision la partie du système responsable de la violation. L'algorithme proposé dans [Haf_97] se base sur le fait que si tous les agents de niveau inférieur ont envoyé le signal *Viol*, alors, s'il s'agit d'un agent TM, il va envoyer un signal *Viol* à l'agent supérieur. Sinon, il s'agit de l'agent source et c'est la source qui est responsable de cette dégradation. Par conséquent, ces qualités violées ne sont plus fournies pour les usagers pour une certaine période. Or, il serait préférable de voir si tous les agents de niveau inférieur qui utilisent la qualité dégradée ont envoyé le signal *Viol* ou non.

Afin de clarifier cette nouvelle approche, nous considérons l'exemple suivant illustré dans la figure 4.11. Nous disposons d'un agent source avec trois agents de niveau inférieur. L'agent TM1 a trois agents Sink de niveau inférieur, S1 et S3 utilisent la qualité Q3, alors que S2 utilise la qualité Q2. En ce qui concerne l'agent TM2, il a trois agents Sink de niveau inférieur: S4 qui utilise la qualité Q1 et S5 et S6 qui utilisent la qualité Q3. L'agent TM3 a un seul agent Sink de niveau inférieur S7 et il reçoit la qualité Q1.

Supposons qu'après une certaine période, la source aura des problèmes et la qualité Q3 sera mal transmise. Les agents Sink S1 et S3 vont envoyer un signal *Viol* à l'agent TM1 et les agents S5 et S6 vont envoyer un signal Viol à l'agent TM2. Selon l'algorithme proposé dans [Haf_97], l'agent TM1 va attendre une certaine durée pour voir si tous les agents du niveau inférieur ont envoyé la même signal Viol. Dans le cadre de la figure 4.11, l'agent S2 ne l'a pas envoyé et donc, l'agent TM2 va émettre un signal Solve à chacun des agents S1 et S3. Par conséquent, l'identification de la partie défaillante qui est responsable de la violation des Qds réside dans les liens se trouvant entre (TM1, S1) ainsi que (TM1, S3) et de la même manière, TM2 va réagir avec les agents Sink S5 et S6.

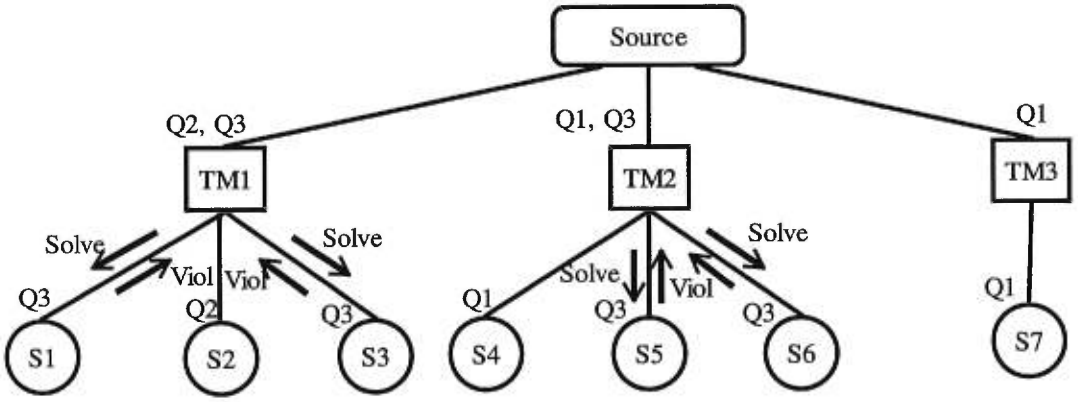


Figure 4.11. Violation de la qualité de service selon l'algorithme proposé

Or, l'algorithme amélioré consiste à voir, par exemple pour l'agent TM1, si pour tous les agents de niveau inférieur qui utilisent une qualité donnée Q3 ont envoyé le signal Viol, alors, un signal Viol est émis à l'agent source sans recevoir le signal Viol de l'agent Sink S2. De cette manière, l'agent TM2 va envoyer le signal Viol à l'agent source suite à la réception du signal Viol concernant la qualité Q3 de la part des agents Sink S5 et S6. Même si l'agent source ne reçoit pas le signal Viol de l'agent TM3, il va réaliser que la violation de la qualité

Q3 est due à la source et que cette qualité ne sera plus fournie jusqu'à ce que la source soit capable de transmettre cette qualité sans qu'elle subisse une dégradation (voir figure 4.12).

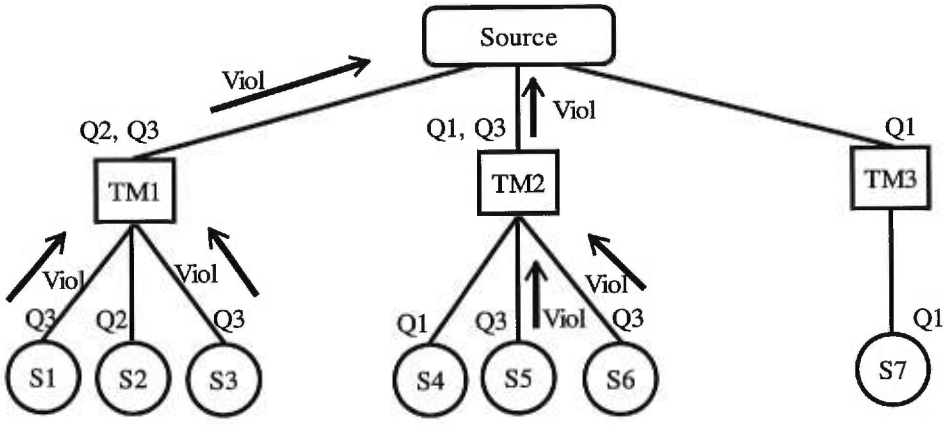


Figure 4.12. Violation de la qualité de service selon l'amélioration suggérée.

4-5-3 Persuasion

La persuasion est réalisée par le système afin d'optimiser les ressources utilisées [Fis-97, Haf-97b]. Dans l'exemple de télé-éducation présenté dans la section 2 (voir figure 4.2), le récepteur R3 est l'utilisateur exclusif de toutes les ressources réservées pour transmettre les flots de données correspondants à la qualité Q1. L'optimisation des ressources sera effectuée au niveau du commutateur de niveau supérieur C3. Celui-ci va essayer de persuader le récepteur R3 de transiter vers la qualité de persuasion Q3. Si le récepteur R3 accepte, alors les récepteurs R1, R2 et R3 pourront bénéficier tous de la meilleure qualité Q3 avec un prix inférieur au prix normal. Cette réduction du prix est due au partage des ressources utilisées pour la transmission des flots de la qualité Q3 et à la libération des ressources de la qualité Q1 qui seront utilisées pour d'autres applications. L'agent commutateur C4 va essayer de persuader, de la même manière, le récepteur R4 de transiter vers la qualité Q3 au lieu d'utiliser Q2 à la demande de l'agent commutateur C1. À l'acceptation du récepteur R4, la source va contacter l'agent commutateur C2 pour que le récepteur R6 change vers la Q3 puisque tous les agents de niveau inférieur de C1 utilisent Q3. Cette nouvelle situation permet une grande libération des ressources puisque les flots d'information concernant les qualités Q1 et Q2 ne sont plus transmis. Par conséquent, les liens seront utilisés pour d'autres applications multimédias ce qui permet une augmentation du profit du système. Elle permet, également, la satisfaction des utilisateurs puisqu'ils reçoivent la meilleure qualité Q3 avec un prix réduit et

avec une garantie de service.

Dans le cas réel d'un système, à la réception du signal *Persuade* par l'agent Sink, celui-ci vérifie s'il ne peut pas supporter la qualité de persuasion localement. Dans ce cas, la renégociation n'est pas réalisée avec l'usager. Dans le cas contraire, la renégociation est effectuée en présentant à l'usager le prix réduit de la qualité de persuasion à travers le signal *Negociation*. L'usager fait son choix et retourne la qualité choisie *Qos* à travers le signal *Response* à l'agent Sink. Celui-ci vérifie si l'usager veut quitter l'application, un signal *Remove_Qos* est envoyé à l'agent TM de niveau supérieur et un signal *Deconnexion* est envoyé à l'usager. Dans le cas où l'usager n'a pas changé de qualité, aucune action n'est réalisée et la renégociation avait pour but de l'informer de la réduction du prix de la qualité de persuasion. Le dernier cas est celui où l'usager change de Qds. Alors, un signal *Remove_Qos* est envoyé à l'agent TM de niveau supérieur puisque l'usager veut transiter vers une autre Qds. Le signal *Add_Qos_Info* est aussi envoyé à l'agent de niveau supérieur pour qu'il lui transmette la qualité choisie.

L'agent TM envoie aussi à l'agent Sink de niveau inférieur le signal *Arret_Persuade*. Ce signal permet d'indiquer que la persuasion est terminée. Par exemple, si l'opération de persuasion n'est plus rentable pour le système, le signal *Arret_Persuade* est envoyé. Ce signal permet d'indiquer que les usagers qui bénéficiaient du prix réduit de la qualité de persuasion doivent payer, dorénavant, le prix régulier. À la réception de ce signal, l'agent Sink réalise une renégociation avec l'usager en présentant les prix réguliers des différentes Qds. Selon le choix de l'usager, l'agent Sink procède aux changements nécessaires.

L'opération de persuasion sera traitée, d'une manière approfondie et détaillée, dans le chapitre suivant.

4-5-4 Déconnexion d'un usager

Si un utilisateur veut quitter l'application, il peut procéder des deux manières suivantes :

- Il envoie le signal *Renegociation_User* sans spécifier aucune qualité de service.
- Il envoie le signal *Response* sans indiquer aucune qualité de service à la suite de la réception du signal *Negociation*.

Dans ces deux cas, l'agent Sink va envoyer un signal *Remove_Qos* vers l'agent de niveau supérieur TM. À la réception de ce signal, l'agent TM accède à l'entrée de *R_Tree_List* qui correspond à *Tree_Id* spécifié dans le signal et modifie la variable *(D_Agent_Id, List_Qos)*

pour indiquer que cet agent de niveau inférieur ne va plus recevoir cette qualité. Ensuite, l'agent TM vérifie si tous les agents de niveau inférieur n'utilisent plus cette qualité. Dans ce cas, la variable *C_Tree_List* est modifiée puisque cette qualité n'est plus fournie à ce niveau et un signal *Remove_Qos* est émis à l'agent de niveau supérieur. Le même traitement correspondant à la demande d'arrêt de transmission de la qualité de la part du TM est réalisé, également, au niveau de la source. À la fin, un signal *Deconnexion* est envoyé par l'agent Sink vers l'usager concerné.

La version originale des algorithmes proposés a été présentée dans [Haf-97b]. Ces algorithmes souffraient de plusieurs anomalies. Nous avons contribué à l'amélioration de ces algorithmes afin d'aboutir à de bons résultats lors de la spécification, la simulation et la validation. Parmi ces améliorations, nous pouvons citer :

(i) La notion du prix n'a pas été introduite dans les algorithmes proposés dans [Haf-97b]. Vue l'importance du prix dans les applications MM, nous avons modifié les algorithmes afin que le prix soit considéré.

(ii) Les traitements correspondants à la negociation ou à la renegociation au niveau de l'agent Sink n'ont pas été présentés dans [Haf-97b]. Nous avons ajouté ces deux fonctions.

(iii) La mise à jour des tables *So_Tree_List*, *R_Tree_List* et *Si_Tree_List* n'a pas été réalisée. Nous l'avons ajoutée pour les différentes opérations. Ceci permet d'avoir des informations cohérentes et d'assurer une bonne communication entre les agents de Qds.

(iv) À la suite d'une dégradation d'une qualité de service au niveau Source, nous avons ajouté la suppression de cette qualité de l'ensemble des qualités disponibles *A_List_Qos* puisque cette qualité ne peut pas être transmise. Nous avons aussi remplacé le signal *Available_Qos* par le signal *Not_Ok.* Ce signal est envoyé à tous les agents TM de niveau inférieur qui reçoivent la qualité dégradée afin de les informer de la violation de la Qds et de réaliser les changements nécessaires.

(v) Dans le cas d'une violation d'une Qds au niveau Source ou Tree_Manager, nous avons ajouté un signal *Not_Ok.* Ce signal est envoyé aux agents voisins de niveau inférieur qui utilisent effectivement la qualité violée afin de les informer de la violation de la

Qds. Alors que, dans la version originale, un signal *Available_Qos* est envoyé à tous les agents de niveau inférieur sans aucune distinction.

(vi) Dans le cas où l'agent Source n'arrive à satisfaire une demande de transmission d'une qualité donnée, nous avons ajouté que l'agent Source envoie un signal *Not_Ok* à l'agent TM concerné. Ce signal lui indique que cette opération n'a pas été réalisée.

(vii) Dans le cas où des agents Sinks ou Tree_Manager sont responsables d'une violation de Qds, nous avons ajouté l'envoi du signal *Remove_Qos*. Ce signal est envoyé à l'agent voisin de niveau supérieur pour arrêter la réception de la qualité dégradée.

(viii) Dans le signal *Solve*, nous avons ajouté la variable *List_Qos*. Cette variable permet d'indiquer les qualités violées.

(ix) Dans le cas où un agent TM ne trouve pas un chemin disponible, nous avons ajouté l'envoi du signal *Not_Ok* à tous les agents voisins de niveau inférieur et qui utilisent une qualité violée. Le signal *Remove_Qos* est aussi envoyé à l'agent de niveau supérieur pour arrêter la transmission de cette qualité.

(x) À la réception du signal *Solve* au niveau d'un agent Sink et si cet agent ne trouve pas un autre chemin disponible, nous avons ajouté l'envoi du signal *Remove_Qos* à l'agent TM de niveau supérieur. Ce signal sert à arrêter la transmission de la qualité violée.

(xi) Nous avons ajouté la variable *V_List_Qos1* au niveau des agents. Cette variable représente l'ensemble des Qds qui ne peuvent être fournies ou reçues pour un certain temps. L'utilisation de cette variable permet d'éviter l'envoi et la réception des signaux de *Add_Qos_Info*, *Remove_Qos* et *Not_Ok* inutilement comme nous avons vu dans la section 4-5-2 de ce chapitre.

(xii) À chaque récupération d'une qualité de service dans [Haf-97b], le signal *Available_Qos* est envoyé à tous les agents de niveau inférieur afin de réaliser une renégociation avec les usagers. Nous avons supprimé ce signal puisque les usagers peuvent faire une renégociation en envoyant le signal *Renegociation_User*.

(xiii) À la réception du signal *Remove_Qos* par un agent TM, le test de persuasion est réalisé dans [Haf-97b] au niveau de cet agent et au niveau supérieur. Nous avons supprimé ce test dans le cas où un signal *Remove_Qos* est envoyé à l'agent de niveau supérieur.

(xiv) Dans [Haf-97b], le test de persuasion est réalisé à chaque demande de réception d'une qualité donnée ou d'arrêt de transmission de cette qualité. À la réception d'un agent Sink du signal *Persuade*, une renégociation est réalisée avec l'usager. Si cet usager transite vers une autre qualité, l'agent Sink doit envoyer les signaux *Remove_Qos* et *Add_Qos_Info* à l'agent TM de niveau supérieur. Cette nouvelle situation déclenche d'autres opérations de persuasion. Ce même scénario est répété pour les autres usagers et ainsi de suite. Ceci entraîne une avalanche de signaux puisque pour chaque test de persuasion, le système exécute plusieurs tests de persuasion d'une manière successive. Le système souffre donc d'une déstabilisation. Nous avons remédié à ce problème par le fait de réaliser le test de persuasion à des intervalles de temps bien précis.

(xv) À la réception du signal *Persuade* par l'agent Sink, nous réalisons le calcul des différents prix de Qds. À la réception du choix de l'usager, si celui-ci choisit de transiter vers une autre qualité, les signaux *Remove_Qos* et *Add_Qos_Info* sont envoyés à l'agent TM du niveau supérieur. Si l'usager veut quitter l'application, les signaux *Remove_Qos* et *Dexonnexion* sont envoyés à l'agent TM et à l'usager respectivement. Ce traitement n'a pas été spécifié dans [Haf-97b].

(xvi) Dans [Haf-97b], un usager qui veut se connecter à une application donnée contacte l'agent Sink. Celui-ci réalise la connexion et présente à l'usager l'ensemble des Qds. Nous avons ajouté que l'agent Sink échange d'abord des signaux (Info_Connexion_S et Info_Connexion_TM, voir chapitre 6 section 6-2) avec l'agent TM de niveau supérieur avant de présenter les Qds à l'usager. Cet échange de signaux permet à l'agent Sink de présenter à l'usager les prix des Qds en tenant en compte la dernière opération de persuasion réalisée.

(xvii) Les différentes politiques de persuasion n'ont pas été définies dans la version de [Haf-97b]. Nous les avons développées ainsi que des stratégies de calcul du prix des Qds après une persuasion et les conditions d'arrêt d'une persuasion déjà effectuée, comme expliqué dans le chapitre 5.

Chapitre 5.
Le principe de la persuasion

5-1 Introduction

Dans une application MM, les participants bénéficient des plusieurs services offerts. Par exemple, ils peuvent assister à une conférence en direct, ou accéder à des documents sans être obligé de se déplacer. En contre partie, ils doivent payer les services fournis pour assurer le maintien et la continuité de ces services. Le prix à payer diffère selon la qualité reçue. En fait, plusieurs paramètres entrent en jeu pour déterminer le prix correspondant à une qualité donnée. Parmi ces paramètres, nous pouvons citer : le type du service, le type de garantie de service et la durée du service. La garantie du service est nécessaire dans une application MM. Elle peut être fournie si les ressources du réseau sont disponibles comme la bande passante, les mémoires tampons et les cycles du CPU. Etant donnée que la quantité de ressources est fixe, le réseau peut subir une congestion. Pour cette raison, le réseau doit posséder une méthode d'optimisation de l'utilisation des ressources pour les gérer d'une manière efficace. En plus, les fournisseurs de services doivent satisfaire les besoins de leurs clients tout en garantissant un bénéfice pour le système. Par conséquent, une bonne stratégie de facturation est nécessaire afin de bien gérer les ressources du système. Malgré que plusieurs travaux ont été effectués pour traiter la gestion de la Qds dans les applications MM, peu d'entre eux ont traité la notion du coût.

Rupp et al. [Rup-98] ont conçu INDEX (Internet Demand Experiment) qui est un modèle d'un marché réel. Il permet de modéliser le comportement des usagers pour évaluer l'utilisation de l'Internet lorsqu'on offre plusieurs choix de Qds à ces usagers. INDEX a deux objectifs principaux :

i. Mesurer la demande de l'usager d'accès à Internet comme une fonction de Qds, une structure de paiement et une application ;
ii. Démonstration d'un système de bout en bout qui produit l'accès aux différents usagers avec une bonne combinaison qualité et prix.

Fulp et al. présentent [Ful-98] une technique distribuée et micro-économique du contrôle du flot qui modélise le réseau comme un marché compétitif. Dans ces marchés, les commutateurs facturent les liens de bande passante selon l'offre, la demande et les usagers achètent la bande passante pour maximiser leurs Qds.

Venkatasubramanian et al. [Ven-97] présentent des métriques de satisfaction des usagers et de la consommation des ressources. Ces métriques sont importantes pour l'évaluation de la qualité de service de la transmission de la vidéo.

5-2 Le coût d'un service

La notion du coût est importante dans les applications MM. En fait, plusieurs facteurs sont utilisés pour déterminer le prix correspondant à une qualité donnée. Parmi ces facteurs, nous pouvons citer :

i. *Le type de service :* un coût fixe est associé à chaque type de service. Par exemple, le coût de l'application vidéo-sur-demande diffère du coût de l'application de téléconférence. Le coût peut aussi changer selon le sujet traité ou le film à voir pour la même application de téléconférence ou de vidéo-sur-demande respectivement.

ii. *La Qds fournie :* le coût est calculé selon les différentes valeurs de paramètres de Qds fournis à l'usager. Le coût, dans ce cas, est calculé, en prenant en considération les ressources réservées dans chaque composant du système. Par exemple, le coût associé à une qualité de l'audio *'C.D'* et résolution *'HDTV'* est plus grand que celui associé à une qualité de l'audio *'téléphone'* et résolution *'TV'* .

iii. *Le type de garantie de service :* le coût diffère selon le type de garantie de service qu'il soit déterministe, statistique ou meilleur effort. Plus le service est assuré, plus la gestion et la réservation de ressources deviennent plus complexes et donc, le coût augmente.

iv. *La durée du service* : plus la durée est longue, plus les ressources sont réservées pour une longue durée et donc, le coût augmente. Afin d'encourager les connexions de grande durée, on peut accorder des rabais aux usagers qui se connectent pour une longue période.

v. *La période de réception du service* : par exemple, pendant le jour, le service coûte plus cher que la nuit.

5-3 Principe de base de la persuasion

La persuasion consiste à encourager les usagers à transiter vers une qualité donnée particulière en payant un prix réduit au lieu du prix régulier. Elle est appliquée dans le cas où les différents sinks d'un commutateur reçoivent des qualités de service différentes. Si les usagers transitent vers une seule qualité, les différentes ressources réservées pour les autres qualités seront libérées. Cette libération de ressources permettra à d'autres participants de se connecter à cette application ou à d'autres applications. Par conséquent, le nombre d'usagers servis sera plus élevé et le profit total du système va augmenter. La persuasion permet, également, le partage des coûts de la réception d'une qualité de service donnée. Ceci permet de réduire le prix à payer par les usagers. Ceux-ci seront, nécessairement, satisfaits s'ils bénéficient d'une qualité donnée avec un prix réduit. Donc, la persuasion a plusieurs avantages comme la satisfaction des usagers, la libération et l'optimisation des ressources et l'augmentation du profit du système. Un exemple détaillé de l'opération de la persuasion a été donné dans la section 4-5-3 du chapitre 4. Dans [Haf-97b], la persuasion n'a pas été détaillée. Une simple politique de persuasion a été présentée et qui consistait à persuader les usagers par la qualité reçue par la majorité d'entre eux. Nous avons modélisé le comportement de l'usager pour choisir une qualité donnée parmi l'ensemble des qualités disponibles. Nous avons défini également des politiques de persuasion qui permettent de choisir la qualité de service qui offre un profit maximal. Nous avons aussi développé des stratégies de calcul du prix des Qds après une persuasion et les conditions d'arrêt d'une persuasion déjà effectuée.

5-4 Les mécanismes de la persuasion

5-4-1 La sélection d'une qualité de service par l'usager

Chaque usager à une préférence précise 'a_i' pour une qualité donnée 'Q_i' pour tout i, i $\in$ (1..n) où n représente le nombre de Qds disponibles. Cette appréciation diffère d'un usager à un autre. Nous supposons que pour tout i, nous avons $a_i<a_{i+1}$, que la qualité Q_{i+1} est meilleure que Q_i, que les bornes des appréciations pour une qualité donnée sont fixes et que les appréciations de chaque utilisateur sont uniformément distribuées entre ces bornes. Ces appréciations sont établies lors de la connexion et elles sont fixes pour chaque utilisateur pour une session de travail donnée. Une appréciation 'a_i' d'une qualité 'Q_i' peut être considérée comme le prix attribué, d'une manière subjective, par l'usager pour évaluer la qualité 'Q_i'.

Par exemple, la figure 5.1 montre la distribution de quatre utilisateurs {U1, U2, U3, U4} selon leurs appréciations pour deux qualités Q1 et Q2 telles que $a_1 \in [0, 100]$ et $a_2 \in]100, 200]$.

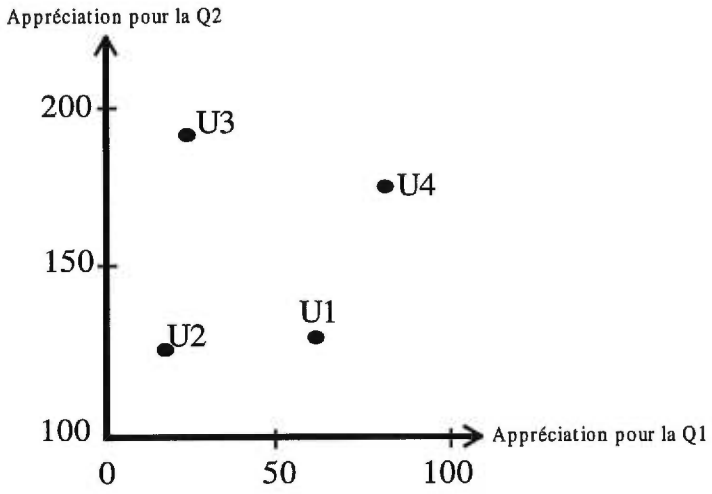


Figure 5.1. Distribution des usagers selon les appréciations

Dans le but de modéliser la manière avec laquelle un utilisateur fait son choix, nous avons proposé les deux approches suivants:

Première approche

Pour chaque qualité *'Q_i' pour tout* i, $i \in (1..n)$ où n représente le nombre de Qds disponibles, le rapport entre l'appréciation de l'utilisateur pour cette qualité *'a_i'* et le prix correspondant *'p_i'* est calculé. Nous supposons que *'p_i'* > 0 pour tout $i \in (1..n)$. La qualité choisie par l'utilisateur est celle dont le rapport *'a_i/p_i'* est le plus grand. L'utilisateur va donc bénéficier de la qualité qu'il apprécie le plus et qui se caractérise par un bon prix.

Le seul inconvénient de cette approche est le fait que les qualités fournies gratuitement aux utilisateurs ne peuvent pas être spécifiées puisque le prix d'une qualité donnée doit être nécessairement non nul.

Deuxième approche

Pour chaque qualité 'Q_i', i ∈ (1..n) où n représente le nombre de Qds disponibles, la différence entre l'appréciation 'a_i' et le prix correspondant 'p_i' est calculée. L'utilisateur choisit la qualité dont la différence 'a_i - p_i' est la plus grande. Du point de vue de l'utilisateur, l'appréciation 'a_i' pour une qualité donnée 'Q_i' représente le gain obtenu en utilisant cette qualité, alors que le prix à payer est le coût pour obtenir cette qualité et donc la différence 'a_i – p_i' représente le profit net. Par conséquent, le fait de choisir la qualité qui se caractérise par la plus grande différence signifie que l'utilisateur cherche à maximiser son profit net obtenu. Cette approche a été retenue puisqu'elle permet de spécifier les qualités fournies gratuitement aux utilisateurs.

5-4-2 Les politiques de la persuasion

Le mécanisme de la persuasion est réalisé afin de satisfaire les usagers, optimiser les ressources et augmenter le profit du système. Ce mécanisme est réalisé au sein de l'agent responsable de la transmission des flots d'une qualité ou d'arrêt de transmission de ces flots. Dans notre version du protocole, nous avons réalisé l'opération de persuasion au sein de l'agent TM dont les agents de niveau inférieur sont, tous, des agents Sink. Cet agent est désigné dorénavant par TM*. Il reçoit n qualités de service. Les deux méthodes de persuasion, que nous avons proposées et spécifiées, sont les suivantes :

1- Politique du meilleur profit *(Best_Profit)*

2- Politique de la meilleure qualité de service *(Best_Qos)*

5-4-2-1 Politique du meilleur profit

La politique du meilleur profit *(Best_Profit)* consiste à offrir aux usagers une qualité de service avec un prix réduit. Elle consiste, d'abord, à calculer le nombre d'agents de niveau inférieur à l'agent TM* qui reçoivent une qualité donnée. La variable *Nb_Ut_Q(i)* permet d'indiquer ce nombre pour chaque qualité i et la variable *Nb_Ut_T* représente le nombre total de ces agents qui reçoivent des qualités. Ensuite, le profit est calculé en se basant sur *C(i)*. *C(i)* représente le coût d'un seul lien associé à la qualité i. Il représente le coût nécessaire pour la réservation des ressources et tout le traitement nécessaire à la transmission des flots de données de la qualité de service i sur un seul lien reliant un agent donné et son agent de niveau

inférieur. Le coût de transmission de la qualité i de la source, par exemple, à l'agent TM* est égale à :

*NH * C(i)* où *NH* est le niveau de profondeur de l'agent TM* dans l'arbre de diffusion.

Pour transmettre ces flots jusqu'à l'agent Sink de niveau inférieur à l'agent TM*, le coût de transmission de la qualité i à un usager devient :

*(NH + 1) * C(i)*

Il est normal que le système doit assurer un gain sur le service offert et par conséquent, le prix d'une qualité i est donné par la formule suivante:

*Prix(i) = (NH+1) * C(i) *(1 + Gain)* où *Gain* > 0 représente le pourcentage de gain.

Le revenu total et le coût de transmission pour chaque qualité i sont donnés par les formules suivantes:

*Revenu(i) = Prix(i) * Nb_Ut_Q(i)*

Cout(i) = *C(i) * (NH + Nb_Ut_Q(i))* pour tout i tel que *Nb_Ut_Q(i)* ≠ *0.*

La figure 5.2 permet d'avoir un exemple d'agent TM* qui possède plusieurs agents Sinks.

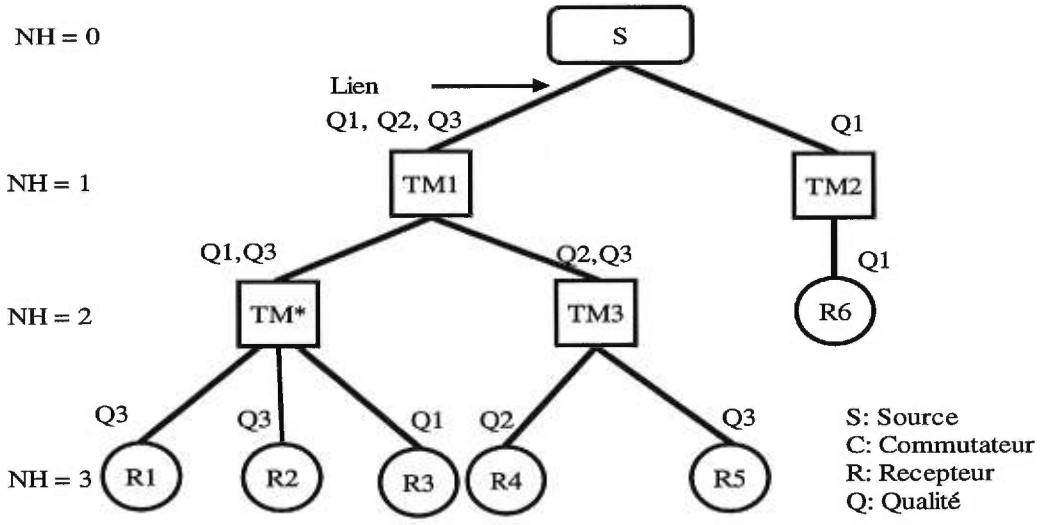


Figure 5.2. Exemple d'un agent TM*

Ainsi, le revenu total et le coût total pour cet agent TM* pour toutes les n Qds sont :

$$Revenu\_Total = \sum_{i:1..n} (NH+1) * C(i) *(1 + Gain) * Nb\_Ut\_Q(i)$$

$$Cout\_Total = \sum_{i:1..n} C(i) * (NH + Nb\_Ut\_Q(i))$$ pour tout i tel que *Nb_Ut_Q(i)* ≠ *0.*

Le coût de transmission de chaque qualité de la source à l'agent TM* est compté une seule fois. Cependant, le coût de transmission entre l'agent TM* et ses agents Sinks de niveau

inférieur est calculé autant de fois qu'il y a d'agents Sinks. Nous supposons qu'il n'y a pas de partage des coûts entre l'agent TM* et la source. Considérons l'agent TM* de la figure 5.2. Cet agent reçoit la qualité Q3. L'agent TM3 reçoit, également, la qualité Q3. Le coût de transmission de cette qualité est compté une seule fois entre la source et le TM1 pour le système. L'agent TM* ne considère pas ce partage de la transmission de la qualité Q3 entre la source et le TM1 pour calculer le coût de la transmission de la qualité Q3. Par contre, pour transmettre cette qualité aux récepteurs R1 et R2, le coût de transmission de cette qualité Q3 est considéré une seule fois de la source jusqu'à le TM*. La figure 5.3 montre la vraie architecture de la figure 5.2 que l'agent TM* prend en considération pour calculer le coût de la qualité Q3.

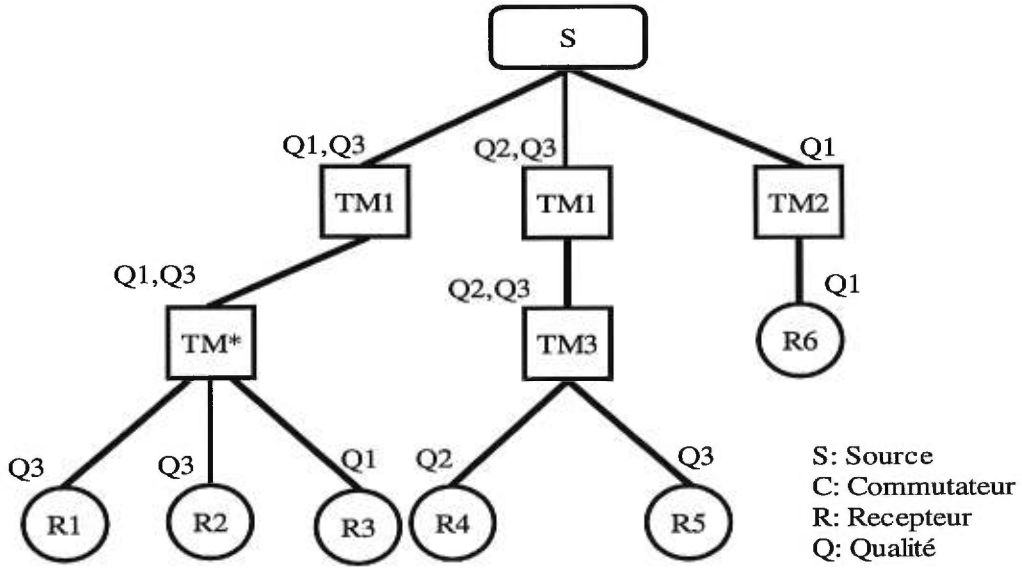


Figure 5.3. L'architecture considérée pour la figure 5.2

Cette méthode de calcul du coût a pour avantage d'augmenter le profit du système s'il y a effectivement du partage des coûts puisque plusieurs TM de niveaux différents utilisent les mêmes Qds. En plus, ceci permet de ne pas modifier les prix à chaque fois qu'un usager se connecte ou se déconnecte d'une application utilisée par un très grand nombre d'usagers.

Le calcul du profit réalisé au niveau de l'agent TM* est donné par la formule:

Profit = Revenu_Total - Cout_Total.

Dans un système réel, la réception par chaque commutateur de l'ensemble des Qds disponibles au niveau de la source pourra entraîner une congestion et une surcharge du système. Le fait

qu'un commutateur transmette uniquement certaines Qds aux différents récepteurs permettra d'utiliser les liens libres pour d'autres applications MM. La libération d'un lien est considérée, dans notre système, comme un gain. Nous avons attribué un pourcentage de gain de libération d'un chemin de transmission des flots de Qds entre la source et l'agent TM. Ce pourcentage est désigné par *Gain_Lib*. La nouvelle formule du profit devient :

*Profit = Revenu_Total - Cout_Total + (val * Gain_Lib)*

val représente le nombre de Qds, initialement disponibles par la source, et non reçues par l'agent TM*.

Ensuite, le profit estimé par la persuasion pour chaque Qds est calculé de la manière suivante:

i) D'abord, l'agent TM* s'informe sur l'ensemble des Qds disponibles au niveau de la source désigné par *A_List_Qos*. Il élimine de cet ensemble les Qds qu'il ne peut pas supporter et cet ensemble devient *V_List_Qos*. En se basant sur un échantillon de *N_Usager* usagers, normalement, très grand (≈ 1000 usagers), l'agent TM* essaie de trouver la Qds de persuasion *Qos_Pers* et le prix de cette Qds *C_Qos_Pers*. Nous avons utilisé la simulation pour trouver les résultats puisque le modèle analytique est plus difficile.

ii) Chaque usager de l'échantillon choisit une appréciation pour chaque qualité $i \in V\_List\_Qos$.

iii) Pour chaque qualité $i \in V\_List\_Qos$ telle que $C(i) > 0$, nous diminuons du prix à payer par l'usager *Prix(i)* une valeur désignée par *Alpha*. Cette valeur peut être minime. Les coûts des autres Qds ne subissent aucun changement. Les différents coûts sont présentés aux usagers de l'échantillon. Selon la méthode de choix d'un usager décrite dans la section 5-4-1, chaque usager choisit une Qds donnée. Pour chaque qualité, nous calculons le nombre d'usagers qui ont choisi cette qualité. Ce nombre est devisé, par la suite, par le nombre d'usagers total *N_Usager* pour avoir la probabilité *Prob(i)* de choisir une qualité i.

iv) Le profit estimé est calculé en utilisant les résultats de l'échantillon sur le nombre total des usagers dans notre système réel. En appliquant les changements sur les formules (voir l'étape b de cette section), nous obtenons les formules suivantes :

$Revenu\_Persuasion = \Sigma_{i:1..n}([((NH+1)*C(i)*(1 + Gain)) - Alpha]* (Prob(i)*Nb\_Ut\_T))$

$Cout\_Persuasion = \Sigma_{i:1..n} C(i)*(NH + (Prob(i)*Nb\_Ut\_T))$, avec $(Prob(i)*Nb\_Ut\_T) \neq 0$

$Profit\_Persuasion = Revenu\_Persuasion - Cout\_Persuasion + (val * Gain\_Lib)$

val représente le nombre de Qds qui ne sont pas reçues par l'agent TM*.

v) La valeur de *Alpha* est incrémentée ($Alpha \leftarrow Alpha + 1$) et les étapes iii et iv sont répétées pour chaque nouvel *Alpha* jusqu'à ce que $Alpha \geq C(i)$ initial.

vi) Le *Profit_Persuasion* maximum de la qualité i obtenu lors de cette boucle est sauvegardé dans *Tab_Profit_Pers(i)* et la valeur de *Alpha* qui réalise ce profit maximum est sauvegardé dans *Tab_Alpha(i).*

vii) Ce traitement est réalisé pour chaque Qds de *V_List_Qos*. Le fait de choisir *V_List_Qos* à partir de *A_List_Qos* et non pas de *C_List_Qos* contrairement à [Haf-97b] permet d'obtenir des meilleurs résultats. Une fois toutes les Qds traitées, nous choisissons *Tab_Profit_Pers(i)* pour $i = i\_max$ qui réalise le profit maximum *Max_Profit* sur toutes les Qds. Par conséquent, la qualité de persuasion *Qos_Pers* est *i_max.*

viii) Pour lancer la persuasion il faut satisfaire la condition suivante :

$(Max\_Profit - Profit) > Gain\_Pers$ ***Condition (1)***

Profit représente le profit réalisé sans faire appel à la persuasion. *Gain_Pers* représente la différence minimale pour persuader les usagers. Selon la condition (1), il ne suffit pas que *Max_Profit* soit supérieur au *Profit* pour persuader les usagers puisque la renégociation va engendrer des signaux qui seront échangés par les différents agents. Ces signaux de gestion et du contrôle seraient inutiles si on n'assure pas un certain gain substantiel. Si la condition (1) est vérifiée, un signal *Persuade* est envoyé à tous les agents de niveau inférieur. Ce signal spécifie que l'agent TM* veut dorénavant transmettre la Qds *i_max* afin de libérer les autres liens et optimiser les ressources. Les usagers ont intérêt, dans la

majorité des cas, à transiter vers la qualité *i_max* puisque l'offre est motivante. Le prix *C_Qos_Pers* obtenu par *(Prix(i_max) - Tab_Alpha(i_max))* est réduit par rapport au prix régulier. Une fois la persuasion est réalisée, la variable e*tat* est initialisée à 1 pour indiquer que l'état actuel est anormal puisqu'il est dû à la persuasion. La variable *etat_Qos* reçoit la valeur de *i_max* afin de garder trace de l'opération de la persuasion réalisée.

L'opération de la persuasion permet de réduire le prix d'une seule Qds à la fois. Les autres prix des Qds ne subissent aucune modification. Ceci a pour but de motiver les usagers à choisir une seule qualité et par la suite, les ressources réservées pour la transmission des autres Qds seront éventuellement libérées.

Dans le but de bénéficier des avantages de la persuasion, L'agent TM* effectue périodiquement un test de persuasion. Le fait d'exécuter ce test à chaque connexion ou déconnexion d'un usager va entraîner une divergence du système. La stabilité du système est nécessaire pour aboutir à de bons résultats lors de la persuasion. Si des usagers transitent vers la qualité *i_max* au cours de la session, ou un nouvel usager se connecte à l'application, ils doivent savoir les prix de différentes Qds. Nous avons proposé plusieurs approches pour déterminer le prix à payer pour recevoir la qualité *i_max* en sachant que les prix réguliers des autres Qds ne changent pas. Parmi ces approches, nous pouvons citer :

a) Présenter les prix réguliers pour toutes les Qds :

   Dans cette approche, nous présentons les Qds avec les prix réguliers. Même si une persuasion a été réalisée, le prix réduit de la qualité de persuasion ne sera pas présenté aux usagers. Mais, un nouvel usager qui se connecte pourra choisir une autre Qds non utilisée par l'agent TM* de niveau supérieur. Cette situation entraînera des nouveaux coûts surtout après la libération de certaines ressources grâce à la persuasion et par la suite, une diminution du profit du système. Cependant, cette approche garantie que si le système ne gagne pas de plus, alors, il ne va pas perdre puisque chaque usager assume, par le prix régulier, ses propres coûts.

b) Présenter le prix réduit pour obtenir la qualité de persuasion pendant une certaine période:
Cette approche consiste à présenter le prix réduit de la qualité de persuasion pendant une certaine période. Ainsi, si un usager fait une renégociation ou un nouvel usager se connecte, il pourra obtenir la qualité de persuasion avec un prix réduit. Donc, elle permet aux usagers de bénéficier de la qualité de persuasion pour une certaine période. Cette période peut, par exemple, être suffisante pour atteindre un certain nombre d'usagers qui utilisent la qualité de persuasion. Cette approche pourra avoir les mêmes inconvénients que l'approche a) après avoir dépassé la durée spécifiée.

c) Augmenter le prix réduit de la qualité de persuasion par un certain pourcentage :
Dans cette approche, le prix réduit de la qualité de persuasion n'est pas présenté. Si un usager fait une renégociation ou un nouvel usager se connecte, il pourra obtenir la qualité de persuasion avec un prix supérieur au prix réduit et inférieur au prix régulier. Ce nouveau prix peut être calculé d'une manière fixe ou dynamique. Nous devons savoir par quelle méthode le nouveau prix de la qualité de persuasion sera calculé. Il est possible que le nouveau prix ne soit pas suffisamment motivant pour que les usagers transitent vers la qualité de persuasion. Par conséquent, le système ne va pas bénéficier des avantages de la persuasion.

d) Présenter le prix réduit de la qualité de persuasion jusqu'à une nouvelle persuasion :
Dans cette approche, nous présentons le prix réduit de la qualité de persuasion jusqu'à la réalisation d'une autre opération de persuasion. Pour les autres Qds, nous présentons les prix réguliers. Cette approche permet de motiver les usagers à choisir la qualité de persuasion et d'augmenter le profit du système. En se basant sur la formule du calcul du profit obtenu par la qualité de persuasion, le nombre d'usagers qui utilisent cette qualité augmente, ainsi que le profit. Le prix réduit de la qualité de persuasion est présenté aux usagers jusqu'à la réalisation d'une nouvelle persuasion où les nouveaux prix des Qds, qui peuvent être réguliers ou réduits, sont déterminés. Cette approche a plusieurs avantages par rapport aux autres. Elle permet d'encourager les usagers à choisir la qualité de persuasion au lieu d'utiliser une Qds qui a été libérée au cours de la persuasion. Elle

permet également le partage du coût de la qualité de persuasion par plusieurs usagers. Par conséquent, cette approche est à retenir.

À la réalisation d'une nouvelle persuasion, les mêmes étapes de l'algorithme présenté ci-dessus sont réalisées. Nous traitons le système de nouveau en calculant le profit réalisé sans persuasion et le profit réalisé avec la persuasion par une qualité donnée. Selon la condition (1):

(Max_Profit - Profit) > Gain_Pers ***Condition (1)***

Si la condition (1) est vérifiée, la persuasion est réalisée. Sinon, nous procédons par la manière suivante:

- Qos_Pers → 0 et *C_Qos_Pers → 0* puisque aucun signal de persuasion ne sera envoyé.
- Si *etat = 0*, aucune persuasion n'a été effectuée avant et aucun changement n'est réalisé. Dans le cas contraire, des usagers sont en train de bénéficier du prix réduit de la qualité de persuasion. Etant donnée que la condition (1) n'est plus vérifiée, aucune Qds ne pourra permettre une augmentation du profit du système. Les usagers qui utilisaient la qualité *Qos_Pers* doivent payer, dorénavant, le prix régulier au lieu du prix réduit puisque le système n'assure aucun profit par rapport à la situation normale. Un signal *Arret_Persuade* est envoyé aux agents Sink de niveau inférieur qui utilisent la qualité de persuasion avec un prix réduit pour les informer de la nouvelle situation. Les variables *etat* et *etat_Qos* sont initialisées à zéro puisque la persuasion est terminée et le système revient à la situation normale.

Cette politique du meilleur profit *(Best_Profit)* permet de présenter une qualité avec un prix réduit aux différents usagers d'une manière équitable. Une autre variante de cette politique qui est aussi intéressante est le fait de réduire le prix de la qualité de persuasion uniquement pour les usagers qui ne reçoivent pas cette qualité. Cette méthode se base sur le fait que les usagers qui ont choisi la qualité de persuasion sont satisfaits de ce choix et ils sont prêts à payer le prix régulier. Cette méthode a plusieurs avantages. Elle permet de présenter la qualité de persuasion avec un prix beaucoup plus réduit que le prix proposé par la première méthode ce qui garantie que plusieurs usagers vont transiter vers cette politique et par la suite, la libération des ressources réservées aux autres Qds. Elle permet également une augmentation du profit pour le système plus que la première méthode.

5-4-2-2 Politique de la meilleure qualité

Cette politique de persuasion est désignée par la politique de la meilleure qualité de service *(Best_Qos)*. Elle est effectuée par l'agent TM* périodiquement. D'abord, l'agent TM* doit vérifier la stabilité du système pour effectuer l'opération de la persuasion. Cette politique est utilisée surtout pour les applications qui se caractérisent par des connexions à long terme. Le principe de base de cette politique est de présenter à l'usager une meilleure qualité avec le même prix qu'il paie normalement pour une qualité moins bonne. Dans cette politique le revenu de l'agent TM* ne change pas puisque les prix des Qds ne subissent aucune modification. Les usagers qui utilisent une qualité meilleure que la qualité de persuasion vont continuer à recevoir cette qualité. Les autres usagers qui utilisent des Qds moins bonnes vont transiter vers la qualité de persuasion puisqu'il est bénéfique pour eux de recevoir une meilleure Qds avec le prix d'une qualité moins bonne. Par exemple, un usager qui reçoit des images en noir et blanc va toujours accepter de recevoir des images en couleur pour une certaine période sans qu'il ne paie plus. Nous supposons que tous les usagers acceptent de recevoir la qualité de persuasion. Cette politique utilise le fait que le coût d'une qualité de persuasion utilisée par plusieurs usagers est réduit grâce au partage du coût de transmission des flots de la qualité de persuasion de la source jusqu'au TM* et à la libération des ressources utilisées par les autres Qds moins bonnes. La politique de meilleure qualité de service consiste à réaliser les actions suivantes :

a) Le calcul du nombre d'usagers par qualité de service et du profit sans aucune persuasion de la même manière que celle décrite dans la politique du meilleur profit.

b) Pour chaque qualité $i \in C\_List\_Qos$, nous calculons le coût de réception de la qualité i par tous les usagers qui reçoivent une qualité moins bonne que i. Les usagers qui reçoivent une qualité meilleure que i ne vont pas changer de qualité.

c) Pour chaque qualité i $\in C\_List\_Qos$, nous calculons le profit selon la formule suivante :

   $Profit(i) = Revenu - Cout(i) + (val * Gain\_Lib)$

   val représente le nombre de Qds qui étaient utilisées avant de persuader par la qualité i.

e) Recherche du profit maximum réalisé *Max_Profit*. La qualité de service utilisée pour obtenir ce profit est la qualité de persuasion *Qos_Pers*.

f) Vérification de la condition suivante :

Max_Profit > *Profit* ***Condition (2)***

Si le profit maximum obtenu est plus grand que le profit sans persuasion, l'agent TM* persuade les agents Sink de niveau inférieur qui utilisent une Qds moins bonne que la qualité *Qos_Pers* pour qu'ils transitent vers cette qualité sans augmenter le prix payé en envoyant le signal *Persuade*. La variable *etat* reçoit 1 puisque l'état du système a changé et une persuasion a été appliquée. La variable *etat_Qos* reçoit la valeur de *Qos_Pers*. Dans le cas contraire, aucune persuasion n'est réalisée.

Après avoir effectué l'opération de la persuasion, le système attend une certaine période pour effectuer une autre persuasion. Pendant cette durée, si un nouvel usager se connecte, ou un usager du système veut transiter vers une autre Qds, ces usagers doivent connaître les prix des différentes Qds. Etant donné que les prix réguliers des Qds n'ont pas changé et que l'usager doit recevoir une qualité donnée d'abord avant de bénéficier de la persuasion, les approches proposées dans la politique du meilleur profit ne peuvent être appliquées à l'exception de la 1ère approche. Ainsi, les prix réguliers des différentes Qds sont proposés. Après une certaine période, l'agent TM* va effectuer une autre persuasion pour s'assurer du bénéfice du système. L'agent TM* vérifie si *etat=1*. Dans ce cas, une persuasion est déjà réalisée et l'agent TM* procède à la réalisation des tâches suivantes:

a) L'arrêt de la persuasion qui a été déjà réalisée par l'envoi du signal *Arret_Persuade* à tous les agents Sink de niveau inférieur qui ont bénéficié de la persuasion.
b) La variable *etat* et *etat_Qos* sont initialisées à zéro puisque le système est revenu à son état normal, i.e. aucune persuasion n'est en cours.
c) Attente d'une certaine durée jugée suffisante pour que les agents Sinks de niveau inférieur à l'agent TM* reçoivent le signal *Arret_Persuade* et procèdent aux changements nécessaires. Ces changements consistent à faire une renégociation avec les usagers. Une fois les usagers changent de Qds et le système se stabilise une nouvelle persuasion est réalisée de la même manière décrite ci-dessus.

Chapitre 6.
Spécification du protocole de gestion de la qualité de service

Ce chapitre est consacré pour la modélisation par objet du protocole de gestion de la qualité de service. La spécification de ce protocole en utilisant le langage de spécification formelle SDL sera aussi présentée.

6-1 Le modèle OMT

Une modélisation par objets (OMT) des différents agents du protocole de gestion coopérative de la qualité de service est présentée dans la figure 6.1.

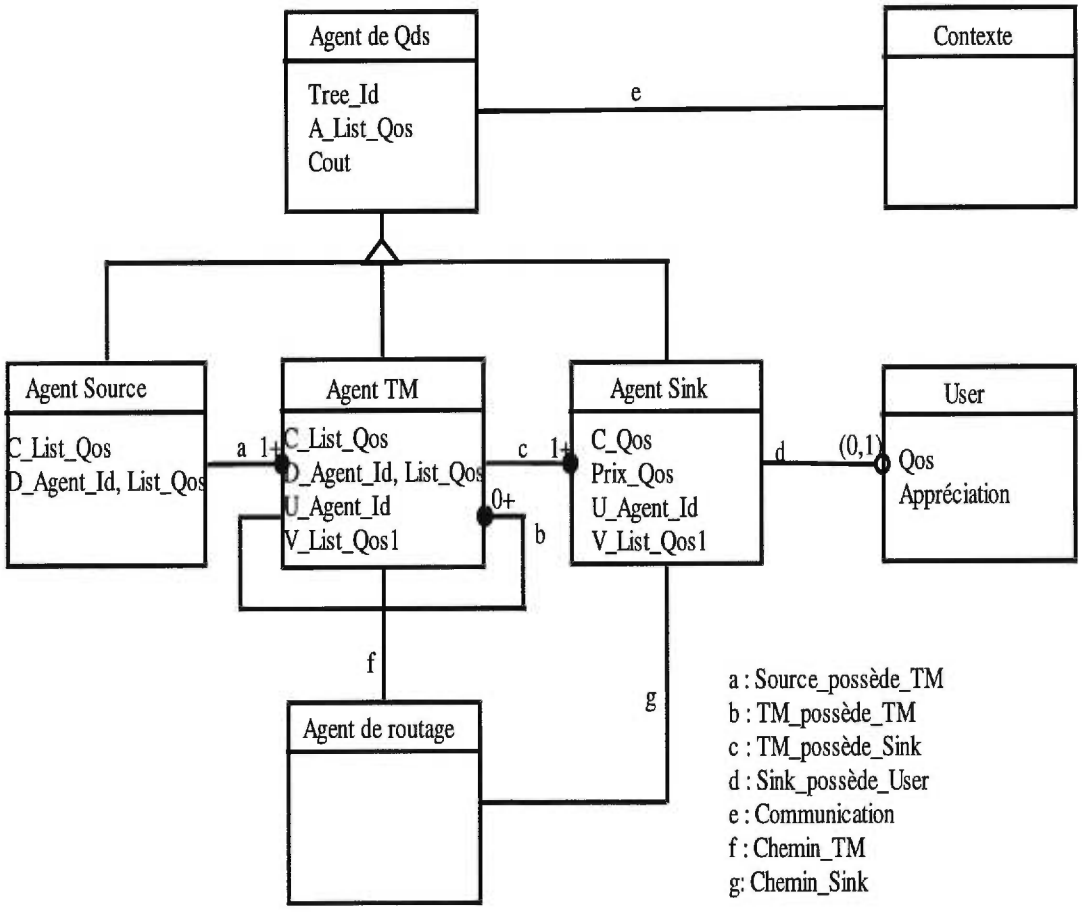


Figure 6.1. Le modèle OMT du protocole de la gestion de la Qds

6-1-1 Description des classes

Dans notre modélisation par objet, nous disposons de l'ensemble des classes suivantes :

a) *Agent de Qds* : Cette classe désigne l'ensemble des agents utilisés dans le protocole de la gestion coopérative de la qualité de service. En utilisant la notion d'héritage, les classes suivantes héritent de la classe *Agent de Qds* :

   i) Classe *Agent Source* : cette classe représente l'agent Source.

   ii) Classe *Agent TM* : cette classe représente l'agent Commutateur. Dorénavant, nous allons designer cet agent par agent Tree_Manager (TM).

   iii) Classe *Agent Sink* : cette classe représente l'agent Sink.

En plus des données héritées de la classe d'*Agent de Qds,* chacune de ces classes a ses propres données. Une description détaillée de ces données a été présentée dans la section 4-2 du chapitre 4.

b) Classe *User* : cette classe représente un usager du système.

c) Classe *Contexte* : cette classe représente l'environnement des différents agents Source, TM et Sink.

d) Classe *Agent de routage* : elle représente l'agent de routage.

6-1-2 Description des associations

Le modèle OMT du protocole de la gestion coopérative de la qualité de service possède des classes et des associations. Celles-ci ont pour rôle de montrer la relation qui existe entre les différentes classes. Les associations du modèle OMT du protocole sont les suivantes :

a) L'association *Source_possède_TM* permet d'indiquer qu'une instance d'*Agent Source* dispose nécessairement d'une ou de plusieurs instances d'*Agent TM.*

b) L'association *TM_possède_TM* indique que chaque instance d'*Agent TM* peut avoir zéro ou plusieurs instances d'*Agent TM.*

c) L'association T*M_possède_Sink* indique qu'une instance *Agent TM* possède nécessairement une ou plusieurs instances d'*Agent Sink.* La classe Agent TM peut posséder comme agent de niveau inférieur soit des agents TM, soit des agents Sink.

d) L'association *Sink_possède_User* indique qu'une instance d'*Agent Sink* peut avoir ou ne pas avoir une instance *User.*

e) Les associations *Chemin_TM* et *Chemin_Sink* indiquent les relations entre l'*Agent de routage* et les *Agents TM* et *Sink* respectivement.

f) L'association *Communication* est établie entre les *Agents de Qds* et le *Contexte*.

6-2 Le principe d'opération du protocole

Le protocole de gestion de la qualité de service permet de gérer, d'une manière coopérative, toutes les opérations de la Qds. Les différentes classes du modèle OMT représentent les agents de ce protocole. Ces agents communiquent entre eux pour réaliser une connexion d'un usager à une application donnée, de faire une négociation de la Qds avec l'usager ou de reconfigurer le réseau lors d'une violation de Qds. Cette communication nécessite l'utilisation des signaux décrits dans la section 4-4 du chapitre 4. En plus de ces signaux, nous avons utilisé les signaux suivants :

- Info_Connexion_S (Tree_Id, Sender_Id, Receiver_Id) est envoyé par l'agent Sink identifié par *Sender_Id* vers l'agent TM de niveau supérieur identifié par *Receiver_Id* pour demander certaines informations concernant la qualité de persuasion et le prix associé.
- Info_Connexion_TM (Tree_Id, Sender_Id, Receiver_Id, Qos, Prix) est un signal destiné à l'agent Sink *Receiver_Id*. Celui-ci a déjà envoyé le signal *Info_Connexion_S* et il reçoit par ce signal les informations demandées.
- Detect_Change(Tree_Id, Qos) est un signal envoyé par le Contexte vers les agents Source, TM et Sink. Il permet d'indiquer que l'environnement de ces agents a détecté une dégradation de la Qds.

La communication est réalisée entre les différents processus par l'échange de signaux de la manière suivante :

1. L'agent Source communique avec l'agent Tree_Manager en lui envoyant plusieurs signaux. Le signal *Not_Ok* lui informe que la source n'arrive pas à transmettre ou à maintenir une qualité de service donnée. Le signal *Solve* indique que la source n'est pas responsable de la violation de certaines Qds et que le TM doit assurer la transmission de ces qualités. Le signal *Persuade* est aussi envoyé pour indiquer la Qds que la source préfère dorénavant transmettre et le prix correspondant à cette qualité.

2. L'agent Tree_Manager communique avec les autres agents de la manière suivante:
    a) L'agent TM envoie à l'agent Source les signaux : *Add_Qos_Info* pour une demande de transmission d'une Qds, *Remove_Qos* pour l'arrêt de transmission d'une Qds et *Viol* lors d'une violation d'une ou plusieurs Qds.
    b) L'agent TM envoie à l'agent Sink de niveau inférieur le signal *Persuade* afin d'indiquer la qualité la plus rentable que le TM préfère transmettre. Si le TM n'arrive pas à envoyer une Qds au Sink, ou pour lui informer d'une violation d'une Qds, un signal *Not_Ok* lui est adressé. Le signal *Solve* est envoyé si le TM n'est pas responsable de la violation de certaines Qds et que le Sink doit assumer la responsabilité en communiquant avec l'agent du routage pour trouver un autre chemin ou établir une renégociation de Qds avec l'usager. Le signal *Info_Connexion_TM* a pour rôle d'informer le Sink de la dernière mise à jour réalisée au niveau TM en ce qui concerne la qualité de persuasion et le prix associé.
    c) Le TM envoie un signal *Ask_Routing* à l'agent de routage pour demander un autre chemin libre pour la transmission des Qds violées.

Etant donné que l'arbre de diffusion considéré permet d'avoir des commutateurs dont les agents voisins de niveau supérieur ou inférieur sont également des commutateurs, alors la communication entre un TM et le TM de niveau supérieur se fait de la même manière que celle entre le TM et la Source et entre un TM et le TM de niveau inférieur correspond à celle qui est entre le TM et le Sink.

3. L'agent Sink échange des messages avec les autres agents de la façon suivante :
    a) L'agent Sink envoie à l'agent TM de niveau supérieur le message *Add_Qos_Info* pour recevoir une Qds donnée. Le signal *Remove_Qos* a pour but de demander l'arrêt de transmission d'une Qds donnée. Il lui envoie également le signal *Viol* pour lui informer de la violation de certaines Qds et afin de recevoir des informations nécessaires pour la connexion d'un usager, il envoie le signal *Info_Connexion_S*.

b) L'agent Sink envoie à l'agent User le signal *Négociation* pour lui informer des différentes Qds disponibles et les prix associés. Ce signal est envoyé lors de la négociation ou la renégociation avec l'usager. Ce dernier est déconnecté du système lors de la réception du signal Déconnexion.

c) L'agent Sink envoie, également, à l'agent de routage le signal *Ask_Routing* pour demander si un autre chemin est disponible afin de transmettre les Qds violées.

4. L'agent User représente le comportement d'un utilisateur du système. Si un utilisateur veut se joindre à une application donnée, il doit d'abord envoyer le signal Connexion à l'agent Sink. À la réception du signal *Négociation*, il choisit une Qds et envoie le signal *Response* qui indique son choix. Si l'usager veut changer la qualité reçue par une autre, il envoie à l'agent Sink le signal *Renégociation_User*.

5. Le Contexte représente l'environnement des agents Source, TM et Sink. Il connaît l'identificateur de chaque agent et tous les voisins de niveau supérieur à chaque TM et chaque Sink et ceux de niveau inférieur à la source et à chaque TM. Cet agent est capable de vérifier les paramètres de Qds mesurés avec les paramètres de la dernière renégociation effectuée avec l'usager. Si ces paramètres diffèrent, le signal *Detect_Change* est envoyé à l'agent responsable et qui peut être soit l'agent Source, un agent TM ou un agent Sink. Ce signal indique que la Qds fournie a subi une dégradation non tolérable.

6. L'agent de routage communique avec l'agent TM et l'agent Sink. Il répond à leur demande de disponibilité d'un autre chemin libre pour la transmission des Qds violées par une réponse positive ou négative selon la situation.

6-3 La spécification du protocole

Afin de développer une spécification du protocole de gestion de la qualité de service en SDL, nous avons utilisé l'outil SDT. SDT est un outil qui permet la spécification d'un système avec le langage de spécification formelle SDL sous les deux formes de notation : textuelle et graphique. Il permet de créer, maintenir et vérifier une spécification en respectant la syntaxe et la sémantique du langage. Cet outil permet également la simulation, la validation et la

génération du code de l'application. Il possède aussi plusieurs librairies qui fournissent plusieurs fonctions C pour manipuler toutes les structures utilisées par le code généré. Tout ceci est présenté à l'utilisateur à travers des interfaces usagers conviviales et avec l'appui de l'aide pour lui faciliter la tâche.

La spécification détaillée du protocole de gestion de la Qds en utilisant la notation graphique du langage de description formelle SDL est donnée dans l'annexe A.

6-3-1 Description des types de données utilisés

Lors de la spécification du protocole de gestion de la qualité de service, nous avons utilisés les types de données suivants:

(i) Vecteurs

Tab : vecteur d'entiers qui indique le nombre de processus fils d'un processus.

Tab_Reel : vecteur de réels qui indique le niveau de profondeur de chaque processus.

TabPid : vecteur d'éléments du type *Pid* qui indique l'identificateur de chaque processus.

Streams : vecteur d'entiers qui indique les qualités disponibles ou non.

Real_Streams : vecteur de réels qui indique les prix des différentes qualités de service.

List_Agent_Qos : vecteur d'enregistrements de type *Agent_Qos* qui indique pour chaque processus Source ou Tree_manager l'ensemble des identificateurs des processus voisins de niveau inférieur ainsi que les qualités reçues par chaque processus.

Probabilites : vecteur de réels qui indique le pourcentage d'usagers qui ont choisi une qualité donnée.

Appreciation : vecteur de réels utilisé pour spécifier les différentes appréciations attribuées par l'usager pour les Qds.

Des sous-ensembles énumérés ont été spécifiés pour indiquer les sous-ensembles d'indices des types de vecteur utilisés.

(ii) Structures

Agent_Qos est un enregistrement de deux éléments: *Agent* de type *Pid* et *List_Qos* de type *Streams*. Il permet d'indiquer les différentes qualités de service *List_Qos* reçues par le processus identifié par *Agent*.

6-3-2 Description des variables utilisées

Chaque agent du modèle OMT a été remplacé par un processus SDL. Les différents processus du système manipulent des données afin de transiter d'un état à un autre. Plusieurs variables, appartenant à des types différents ont été utilisées pour spécifier une opération de gestion ou d'établissement de l'arbre de diffusion. Une description détaillée des différentes variables utilisées par les processus les plus importants est présentée ci-dessous.

(i) Au niveau du processus Source

- *Tree_Id* : de type *Integer*. Elle représente l'identificateur de l'arbre de diffusion associé à l'application.
- *A_List_Qos* : de type *Streams*. Elle représente la liste des qualités disponibles au niveau de la source. *A_List_Qos(i) = 1* indique que la qualité numéro i est disponible. Dans le cas contraire, *A_List_Qos(i) = 0.*
- *C_List_Qos* : de type *Streams*. Elle représente la liste des qualités fournies actuellement par la source. *C_List_Qos(i)=1* indique que la qualité i est fournie actuellement, sinon *C_List_Qos(i) = 0.*
- *Cout* : de type *Real_Streams*. Elle indique les coûts des différentes Qds. *Cout(i)* indique le coût associé à la Qds i.
- *Downstream_Agents* : de type *List_Agent_Qos*. Elle indique l'ensemble des agents de niveau inférieur à l'agent source ainsi que la liste des Qds reçues par chaque agent.
- *NQ*: représente le nombre de Qds disponibles pour une application donnée.
- *T, T_rec, Tpers* sont des temporisateurs utilisés pour spécifier les contraintes temporelles.
- *T_Valeur,* de type *Duration,* représente la durée nécessaire pour l'expiration d'un temporisateur.
- *NR* : représente le nombre de processus TM de niveau inférieur.

(ii) Au niveau du processus TM

Le processus TM dispose des différentes variables utilisées par le processus Source à l'exception de la variable *NR*. Le processus TM possède en plus les variables suivantes :

- *V_List_Qos1* : de type *Streams*. Elle permet d'indiquer l'ensembles des Qds qui ne peuvent pas être supportées au niveau du TM à cause d'une certaine défaillance. *V_List_Qos(i) = 1* indique que la qualité i ne peut être reçue par le TM.
- *NS* : représente le nombre de processus TM ou Sink de niveau inférieur.
- *TM_Sink* est une variable Booléenne qui est égale à *'vrai'* si le processus TM a des processus Sink comme processus de niveau inférieur et *'Faux'* dans le cas contraire.
- *NH* : représente le niveau du processus TM dans l'arbre de diffusion.
- *Qos_Pers* : représente la qualité que le processus TM préfère transmettre aux autres processus de niveau inférieur.
- *C_Qos_Pers* est le prix à payer pour recevoir la qualité *Qos_Pers* par les processus de niveau inférieur. Ce prix est normalement inférieur au prix régulier afin d'encourager les usagers à recevoir la qualité *Qos_Pers*.

(iii) Au niveau du processus Sink

Le processus Sink dispose des variables *Tree_Id, A_List_Qos, cout, NQ, T_rec, T_Valeur, NH, V_List_Qos1* de la même manière décrites ci-dessus. Il dispose, également, des variables suivantes:

- *C_Qos* représente la qualité reçue par le Sink.
- *List_Prix_Qos* est une variable de type *Real_Streams*. Elle représente la liste des prix des Qds présentées à l'usager. *List_Prix_Qos(i)* indique le prix de la qualité i. Si *List_Prix_Qos(i)=-1*, la qualité i ne peut pas être fournie à l'usager. Dans le cas contraire, *List_Prix_Qos(i) >= 0.*
- *Qos_P* et *C_Qos_P* représentent la qualité de persuasion et le prix de cette qualité respectivement.

6-3-3 Description des opérations

6-3-3-1 Etablissement de l'arbre de diffusion

Dans le cas réel d'un système, nous disposons de toutes les informations nécessaires pour établir la communication entre les différents agents. À l'aide de l'agent de routage, l'arbre de diffusion est établie. Celui-ci permet à chaque agent de connaître ses agents de

niveau supérieur ou inférieur. Nous avons spécifié l'arbre de diffusion par l'envoi de certains signaux entre les processus Source, TM, Sink et Contexte.

Représentation graphique du système

Tout d'abord nous introduisons l'architecture du système sous forme d'un graphe dont la racine est le processus Source. Celui-ci a le numéro 1. Les autres processus sont numérotés par niveau de gauche à droite. La figure 6.2 montre un exemple du système décrit dans la figure 4.2 du chapitre 4.

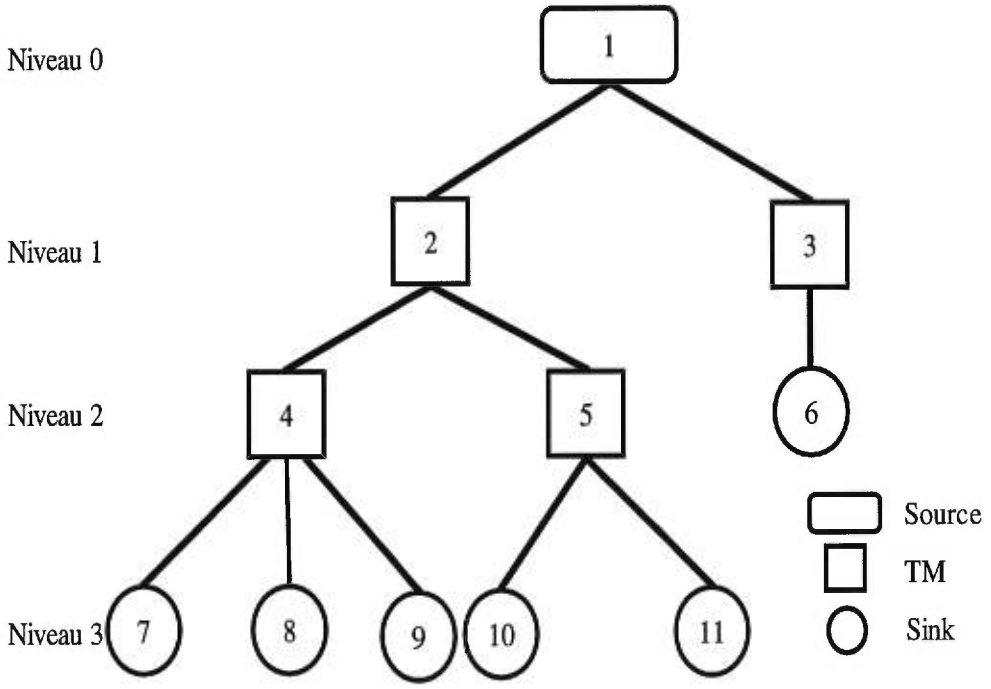


Figure 6.2 Représentation d'un système sous forme d'un graphe

L'arbre de diffusion est établie par le processus Contexte. Celui-ci connaît le nombre total des processus à créer indiqué dans la variable *Nb_Agent*. Il possède, également, trois vecteurs nécessaires pour la spécification de n'importe quel système. Ces vecteurs sont les suivants:

i. Vecteur *T* qui contient pour chaque i le nombre de processus de niveau inférieur avec i de 1 à N. N représente le nombre de processus qui possèdent des processus de niveau inférieur.

ii. Vecteur *TK* qui indique pour chaque processus i le numéro à partir duquel il va commencer à numéroter ses processus de niveau inférieur.

iii. Vecteur *NH* qui présente le niveau de profondeur de chaque élément i dans l'arbre de diffusion. Le processus Source a un niveau 0 (NH = 0) puisqu'il est la racine de cet arbre.

L'exemple suivant permet de clarifier la spécification de l'architecture du système de télé-éducation présenté dans la figure 6.2. L'indice i des vecteurs *T*, *TK* et *NH* varie de 1 à 5 et l'indice 1 représente le processus Source. Nous disposons des informations suivantes :
$T = (2, 2, 1, 3, 2)$, $TK = (2, 4, 6, 7, 10)$ et $NH = (0, 1, 1, 2, 2)$.
T(1) indique que le processus Source dispose de deux processus TM de niveau inférieur. Le premier processus TM de ces deux processus a comme numéro 2 puisque $TK(1) = 2$ et le deuxième a le numéro 3, c'est à dire, $TK(1) + 1$. Ce numéro indique l'indice i de ces trois vecteurs. Ainsi, le processus TM du numéro 2 est de niveau 1 puisque $NH(2) = 1$ et il a, lui aussi, deux processus TM de niveau inférieur puisque $T(2) = 2$. Le premier a le numéro 4 puisque $TK(2) = 4$ et le deuxième a le numéro $TK(2) + 1 = 5$. Ces deux agents ont la niveau 2 dans l'arbre de diffusion car $NH(4) = NH(5) = 2$ et ainsi de suite.

Dans notre système, nous disposons de deux types de processus TM :

- Processus TM dont les processus de niveau inférieur sont tous des processus TM
- Processus TM dont les processus de niveau inférieur sont tous des processus Sink

Pour savoir si un processus TM possède des processus TM ou Sink comme processus de niveau inférieur, nous utilisons l'astuce suivant. Nous multiplions l'élément correspondant dans le vecteur *TK* par cent, par exemple. À condition que nous ne dépassons pas ce nombre dans la numérotation des processus. Le vecteur *TK* devient : $TK = (2, 4, 6, 700, 1000)$. Ceci signifie que le processus TM de numéro 4 dispose de 3 processus de niveau inférieur. Ces processus sont des processus Sinks puisque $TK(4) = 700 = 7 * 100$. Le premier de ces éléments a le numéro 7 et ainsi de suite.

Échange de signaux entre processus

Le processus Contexte envoie au processus Source un signal *NbrS*. Celui-ci contient deux éléments : le nombre de processus de niveau inférieur au processus Source et le premier numéro pour la numérotation de ces processus. Le processus Source reçoit ce signal et il lui retourne son adresse dans le signal *Adr_Source*, l'adresse de chaque processus TM créé, ainsi

que son numéro dans le signal *Adr_TM*. À la réception de ces signaux, le processus contexte met l'adresse de chaque processus créé en accédant par son numéro comme indice dans le vecteur *TPid*. Ce vecteur contient toutes les adresses des différents processus et il permet au processus Contexte de connaitre l'architecture du système. La réception du signal *Adr_TM* par le processus Contexte va déclencher l'envoi du signal *Nbr* au processus TM créé. Ce signal contient le nombre de processus d'agent de niveau inférieur, le numéro à partir duquel il va commencer la numérotation, et son niveau de profondeur dans l'arbre de diffusion. À la réception de ce signal, le processus TM met à jour la variable booléenne *TM_Sink*. Cette variable est égale 'True' s'il va créer des processus Sink et 'False' dans le cas où il va créer des processus TM. Ces deux cas sont les suivants :

i. Dans le premier cas, le processus TM envoie au processus Contexte le signal *Adr_Sink* à chaque création du processus Sink. Le signal *Adr_Sink* contient l'adresse du processus Sink créé et son indice dans le vecteur *TPid*. Le processus Contexte reçoit ce signal et met à jour le vecteur *TPid*. Le processus TM envoie au processus Sink créé un signal *Commence* en indiquant le niveau de profondeur de ce processus créé. Celui-ci crée le processus User à la réception de ce signal et lui envoie le signal *Init*.

ii. Dans le deuxième cas où le processus TM crée des processus TM, il envoie un signal *Adr_TM* au processus Contexte. Celui-ci reçoit tous les signaux qui lui sont envoyés par les différents processus jusqu'à ce que le nombre des processus créés soit égale à *Nb_Agent* qui représente le nombre total d'agents.

6-3-3-2 Violation d'une qualité de service

En se basant sur la section 4-5-2 du chapitre 4, la détection de la dégradation de la Qds est réalisée par chaque agent. Celui-ci prend des mesures sur les valeurs des paramètres de Qds et les compare avec les valeurs négociées. Si elles ne sont pas conformes, il procède à l'identification de la partie responsable de cette dégradation. Le processus Contexte a été conçu pour représenter l'environnement des différents agents. Il permet de simuler l'occurrence des pannes susceptibles de toucher les agents. Ces pannes déclenchent une violation de la Qds et une dégradation de la qualité reçue par l'usager. Pour un besoin de simulation, une fonction aléatoire est utilisée pour indiquer le processus responsable de la

violation de la Qds ainsi que la qualité dégradée. Le signal *Detect_Change* est envoyé à ce processus pour qu'il procède à appliquer le mécanisme de reconfiguration du réseau (voir chapitre 4 section 4-5-2). À la réception de ce signal par le processus Sink ou TM, ces processus vérifient s'ils sont effectivement responsables ou non. Dans le cas réel, ceci est réalisé par la vérification de la conformité des flots reçus par l'agent de niveau supérieur avec ceux qui sont fournis à l'agent de niveau inférieur. Ce comportement a été spécifié par l'utilisation d'une fonction aléatoire qui permet d'indiquer si le processus est responsable ou non. La figure suivante est la représentation graphique de la spécification de ce mécanisme au niveau de Tree_Manager.

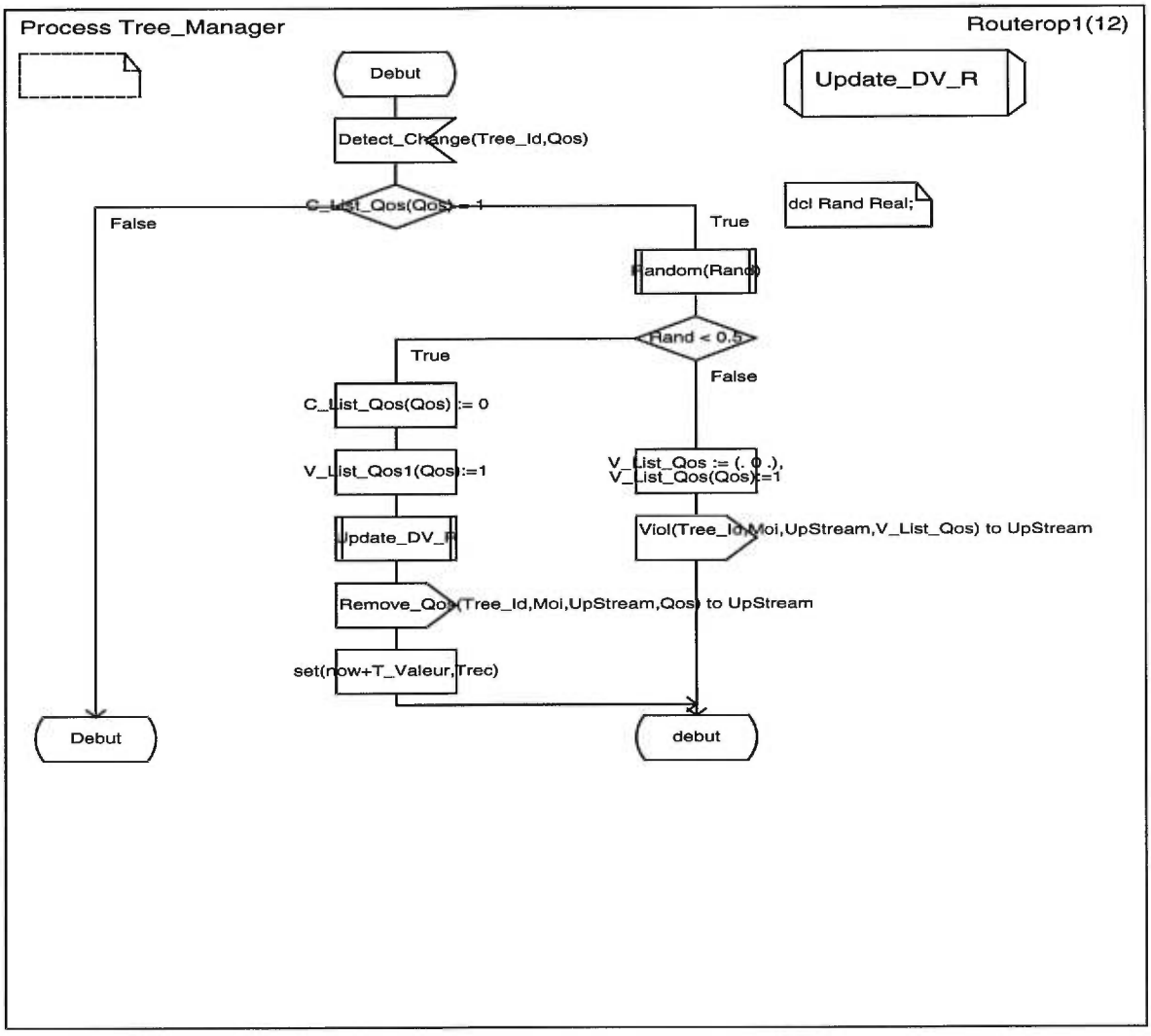


Figure 6.3. Mécanisme de détection d'une violation de Qds au niveau TM

En ce qui concerne le mécanisme d'identification de la partie responsable d'une violation de Qds, un agent TM ou Source qui reçoit un signal *Viol* doit attendre un certain moment pour voir s'il va recevoir un message similaire de tous les agents de niveau inférieur. Nous avons utilisé un temporisateur pour spécifier cette durée d'attente. À l'expiration de ce temporisateur, la durée d'attente est terminée et le processus procède à l'exécution des autres actions. Au niveau du processus Source, s'il reçoit un signal *Viol* de tous les processus de niveau inférieur, un temporisateur est activé. À l'expiration de ce temporisateur, la récupération des Qds est réalisée.

Dans un système réel, l'occurrence d'une panne dans un composant du réseau entraîne une violation de la Qds. Ce composant ne peut pas recevoir la qualité dégradée pour une certaine période. La récupération (Recovery) de la Qds violée est réalisée par la réparation de la défaillance. Pour spécifier la récupération d'une Qds par un processus, nous avons utilisé un temporisateur. À chaque fois qu'une violation de la Qds est détectée par un processus responsable de cette violation, ce temporisateur est activé. La durée nécessaire pour son expiration est jugée suffisante pour la réparation de la défaillance. À l'expiration de ce temporisateur, une fonction aléatoire est utilisée, pour fin de simulation, afin d'indiquer la qualité récupérée. Cette qualité est supprimée de la variable *V_List_Qos1*. Nous rappelons que *V_List_Qos1* représente la liste des Qds qu'un agent TM ou Sink ne peut pas recevoir à cause d'une défaillance et donc ces qualités ne peuvent pas être transmises aux voisins de niveau inférieur. La figure 6.4 est la représentation graphique de la spécification de ce mécanisme au niveau de Tree_Manager. Au niveau du processus Source, la récupération de la Qds est réalisée par la modification de la variable *A_List_Qos* par la même méthode. Ainsi, une qualité violée, qui n'était plus disponible, est fournie de nouveau par la source.

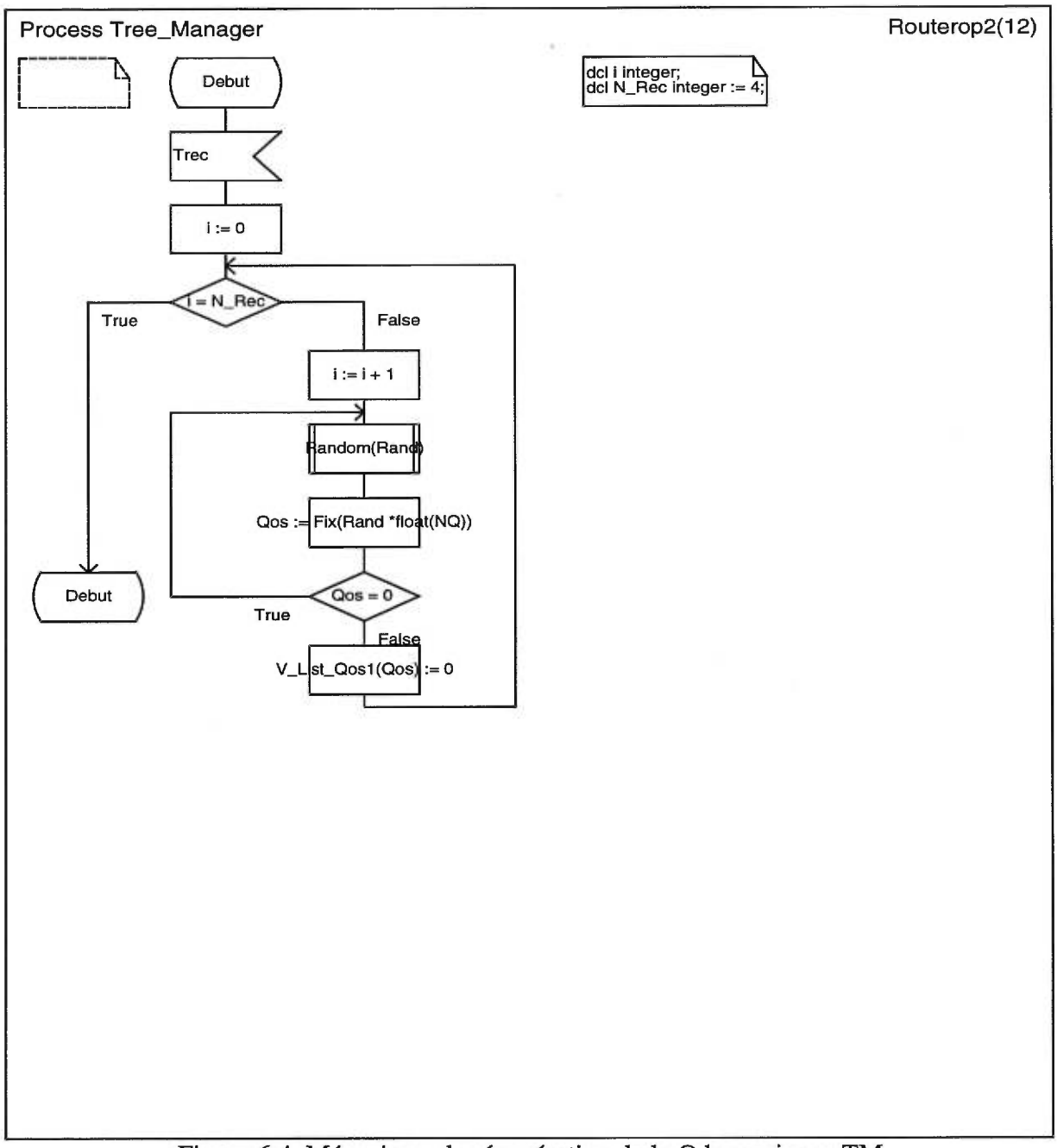


Figure 6.4. Mécanisme de récupération de la Qds au niveau TM

Dans le cas où les agents TM ou Sink communiquent avec l'agent de routage pour trouver un autre chemin disponible, l'agent de routage vérifie la disponibilité de ce chemin pour la transmission des flots des Qds. Ce comportement a été spécifié par l'utilisation d'une fonction aléatoire. Si le processus Prot_Routing qui représente l'agent de routage trouve un chemin libre, la réponse est positive. Dans le cas contraire, la réponse est négative.

6-3-3-3 Négociation ou renégociation d'une qualité de service

À la réception d'une demande de connexion, une négociation est réalisée avec l'usager. Une renégociation est en fait une négociation, mais, au lieu d'être exécutée au début d'une session de travail, elle est réalisée pendant une session. La négociation ou la renégociation de la Qds, qui se réalise au niveau du processus Sink, consiste à effectuer les étapes suivantes:

i. La réception de l'ensemble des Qds disponibles au niveau de la Source.
ii. L'annulation des Qds qui ne peuvent pas être supportées localement par le Sink
iii. La réception de l'ensemble des coûts pour chaque Qds.
iv. Le calcul du prix de chaque Qds en se basant sur des informations locales.
v. L'attribution d'un prix négatif aux Qds qui ne peuvent être pas fournies afin de les distinguer des Qds fournies gratuitement aux utilisateurs.
vi. Modification des prix selon les informations de la persuasion.

La liste des prix des différentes Qds est fournie à l'usager. L'usager choisit une qualité donnée parmi l'ensemble des qualités proposées en se basant sur certains critères. Pour modéliser ce choix, l'usager exprime ses préférences en terme d'appréciation. Nous utilisons une fonction aléatoire afin d'attribuer à chaque usager, lors de son initialisation, une appréciation pour chaque qualité en respectant les bornes fixes des appréciations. Selon la méthode de la sélection d'une Qds de la section 5-3-1 du chapitre 5, l'usager fait son choix et répond par le signal *Response*. Selon sa décision, l'agent Sink procède aux changements nécessaires.

6-3-3-4 Persuasion

L'opération de persuasion est réalisée dans le cas d'un système réel à des intervalles de temps bien précis. Pour spécifier la durée nécessaire entre deux persuasions successives, un temporisateur a été utilisé. À l'expiration de ce temporisateur, l'opération de persuasion est réalisée. Pour un besoin de simulation, une fonction aléatoire a été aussi utilisée. Cette fonction permet d'attribuer à chaque usager de l'échantillon des appréciations pour chaque Qds.

Le processus de gestion a été conçu pour permettre une gestion automatique d'un très grand nombre d'usagers pour la simulation. Ce processus permet également de réaliser certaines statistiques sur l'opération de la persuasion. Le processus de gestion reçoit des informations du processus TM. Ces informations sont : le profit sans persuasion, le profit estimé après persuasion, le profit réalisé après persuasion, le nombre total d'usagers, le nombre d'usagers par Qds, etc. Les statistiques réalisées sont : le cumul des bénéfices réalisés par la persuasion, le taux de satisfaction des usagers. Ce taux indique le nombre d'usagers qui ont accepté de transiter vers la qualité de persuasion et donc, ils sont satisfaits de la persuasion.

6-4 Remarques sur l'utilisation de SDL

La difficulté de la spécification du protocole de la gestion de la qualité de service résidait dans les algorithmes décrits dans [Haf-97b]. Ces algorithmes souffraient de plusieurs anomalies (voir chapitre 4). Une fois que ces algorithmes ont été améliorés, la spécification est devenue moins difficile surtout avec la notation graphique du langage de description formelle SDL. Cependant, l'aide de SDL ne permet pas d'accéder facilement à certains sujets. La manipulation des fonctions aléatoires est difficile puisqu'elles ne permettent pas de générer un germe diffèrent à chaque simulation ce qui entraîne des suites de termes similaires. La spécification de la partie de la construction de l'arbre de diffusion nécessitait plus d'effort pour aboutir à de bons résultats. Ceci est dû à la création dynamique des processus et à l'échange d'un certain nombre de signaux.

Chapitre 7.
Simulation de la performance

Nous avons présenté dans les chapitres précédents la spécification du protocole de la gestion coopérative de la qualité de service. Nous avons présenté également les politiques de persuasion. Afin de pouvoir vérifier le bon fonctionnement du protocole, nous l'avons simulé en utilisant l'outil SDT. Cette simulation permet de détecter les différentes erreurs introduites dans les phases d'analyse des besoins et de conception. Nous avons également simulé les politiques de persuasion afin de déterminer si elles sont bénéfiques à appliquer. Dans ce chapitre, nous allons présenter les résultats de la simulation de la spécification du protocole de gestion de la qualité de service et de l'application de la persuasion.

7-1 Types de simulation

Les types de simulation utilisés pour simuler une spécification donnée sont: la méthode exhaustive, la méthode guidée et la méthode aléatoire.

i. Méthode de simulation exhaustive :

Dans cette méthode, le simulateur permet de générer le graphe d'accessibilité correspondant au système SDL. Ce graphe contient tous les états accessibles du système. Cette méthode de simulation consiste à vérifier exhaustivement tout le graphe en explorant tous ses états. Cependant, cette méthode de simulation est difficile à réaliser à cause de l'explosion combinatoire des états.

ii. Méthode de simulation guidée

Ce type de simulation permet à l'utilisateur de contrôler la simulation du système. L'utilisateur peut changer l'état d'un processus ou certaines variables afin qu'il vérifie la partie du graphe d'accessibilité qu'il préfère.

iii. Méthode de simulation aléatoire

Ce type de simulation permet une vérification aléatoire du système. Le simulateur exécute une suite de transitions de manière aléatoire. Cette méthode permet la vérification uniquement de certaines parties du graphe d'accessibilité et non pas la totalité du graphe puisque la simulation est faite d'une manière aléatoire.

7-2 Simulation du protocole de gestion de la qualité de service

Après avoir spécifié le protocole de gestion de la qualité de service, nous l'avons simulé avec les deux méthodes de simulation : aléatoire et guidée. Les différentes opérations de la gestion décrites dans la section 4-5 du chapitre 4 ont été exécutées avec succès. Comme exemple de simulation, nous considérons dans ce que suit un agent Source qui possède un seul agent TM. Celui-ci possède deux agents Sinks S1 et S2. Nous utilisons les diagrammes d'ordonnancement des messages (MSC) comme trace de simulation. Ces diagrammes décrivent les interactions entre les différents composants du système. Le diagramme MSC correspondant à l'établissement de l'arbre de diffusion est présenté dans l'annexe B. Le diagramme suivant (voir figure 7.1), obtenu par l'outil SDT, montre la connexion de deux usagers à cette application. D'abord l'usager 1 envoie un signal *Connexion* à l'agent Sink1. Ce dernier lui présente les Qds avec les prix correspondants à travers le signal *Negociation* après avoir reçu certaines informations de l'agent TM de niveau supérieur. L'usager 1 fait son choix et réponds par le signal *Response*. L'agent Sink1 envoie un signal *Add_Qos_Info* à l'agent TM du niveau supérieur afin de recevoir la Qds demandée. De la même manière, la connexion de l'usager 2 est réalisée avec l'agent Sink2.

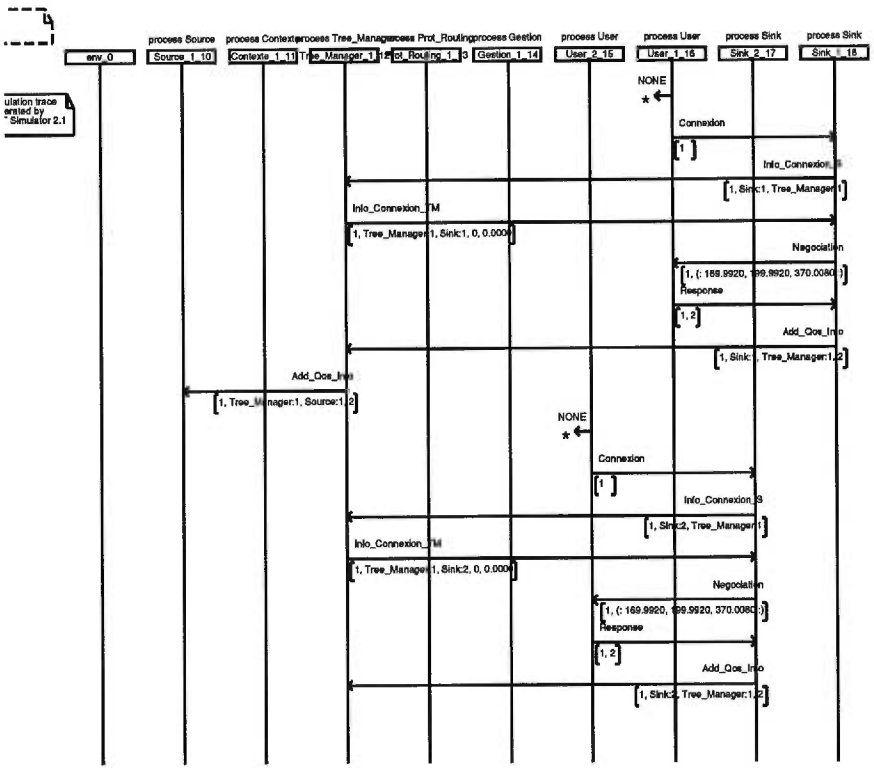


Figure 7.1. Le diagramme MSC de la connexion des usagers

Ensuite, nous considérons qu'un signal *Detect_Change* est envoyé à l'agent Sink1 par le processus Contexte pour lui informer de la détection d'une dégradation d'une Qds. Dans ce cas, l'agent Sink1 n'est pas responsable. Il envoie un signal *Viol* à l'agent TM. Celui-ci lui répond par le signal *Solve* puisque l'agent Sink2 n'a pas envoyé un signal *Viol* à l'agent TM. Le diagramme correspondant à cette situation est le suivant:

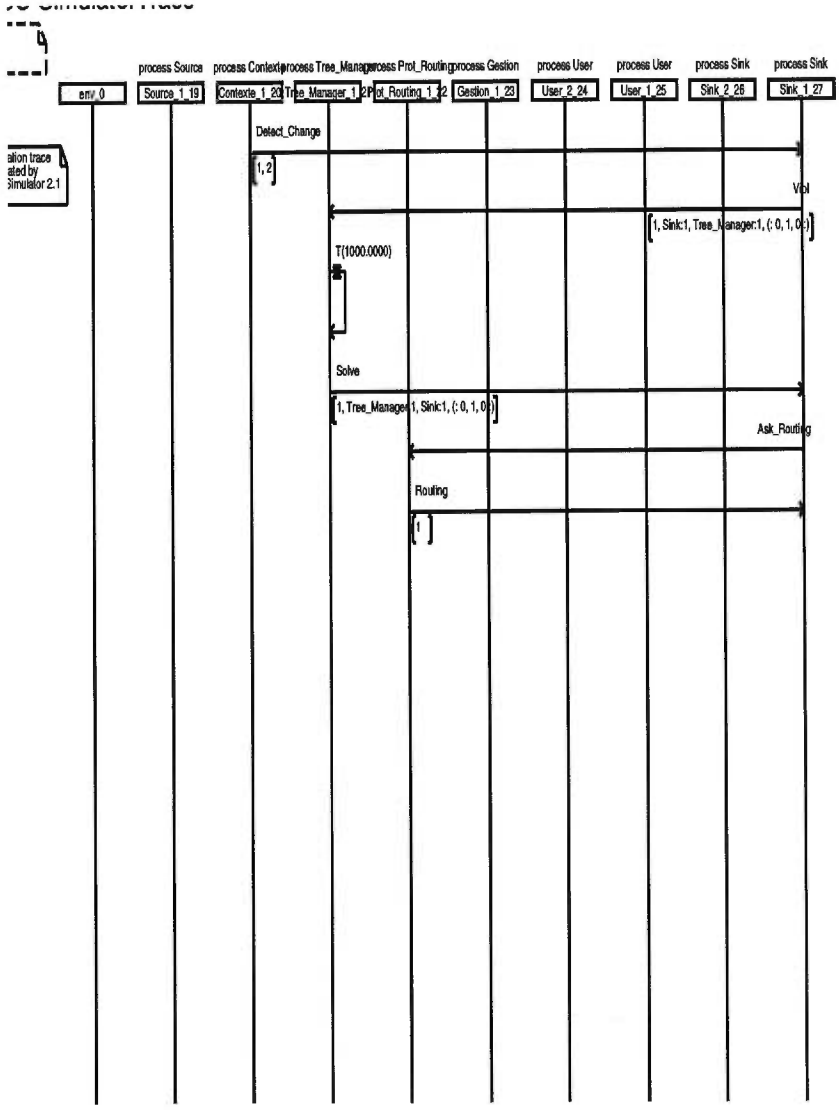


Figure 7.2. Le diagramme MSC de la dégradation de la Qds au niveau Sink1

Le diagramme suivant représente le mécanisme de la violation de la Qds au niveau Sink. Les deux agents Sinks reçoivent le signal *Detect_Change* de la part du processus Contexte. Les deux agents ne sont pas responsables de cette violation de la Qds et par conséquent, ils envoient un signal *Viol* à l'agent TM. Celui-ci envoie également ce signal à l'agent Source. Ensuite, une renégociation est réalisée avec les usagers à la suite de la

réception du signal *Not_Ok* par l'agent Source. La récupération de la Qds est aussi effectuée au niveau de l'agent Source après l'expiration du temporisateur Trec.

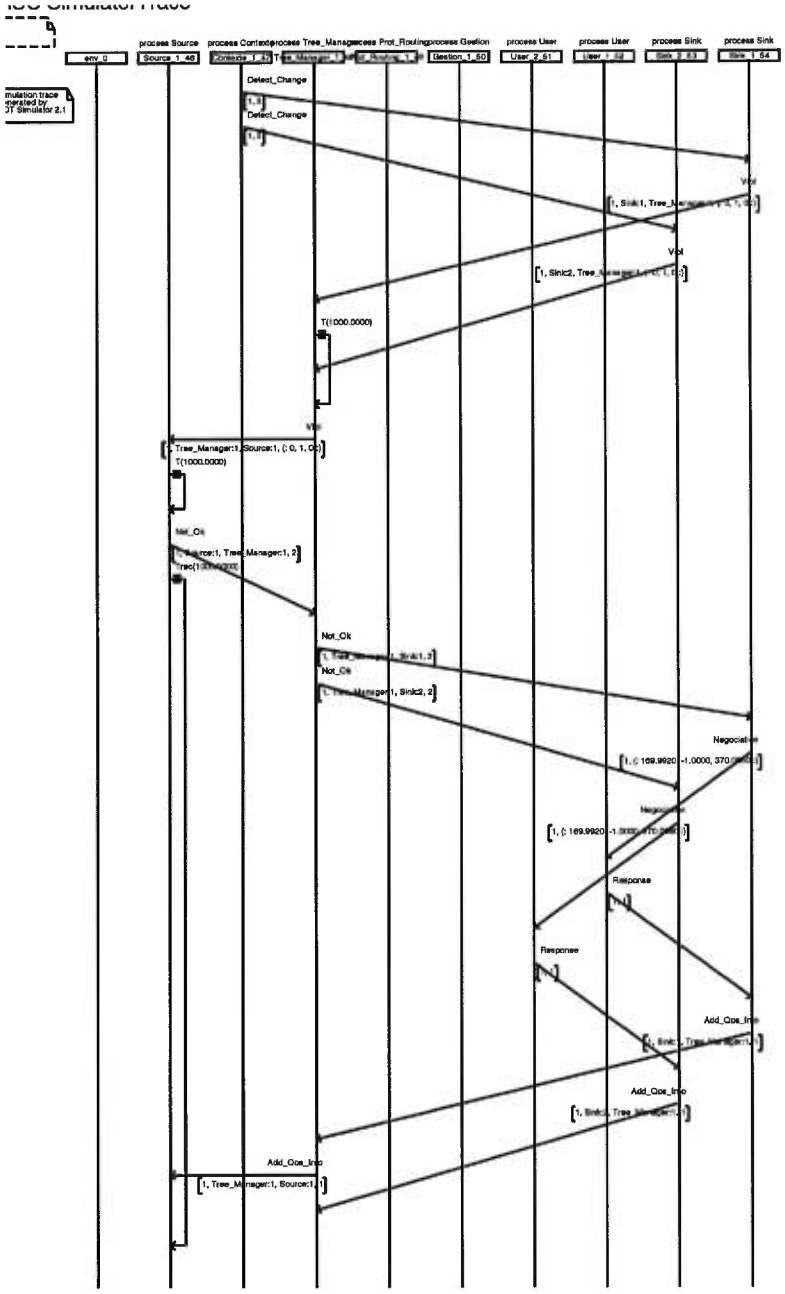


Figure 7.3. Le diagramme MSC de violation de la Qds au niveau des Sinks

D'autres diagrammes MSC relatifs à cette exemple sont présentés dans l'annexe B. Ces diagrammes représentent les situations suivantes :

(i) Un signal *Detect_Change* est envoyé à l'agent Sink2. Celui-ci est responsable de la violation de la Qds. L'agent Sink2 procède aux changements nécessaires et réalise une renégociation avec l'usager2.

(ii) Un signal *Detect_Change* est envoyé à l'agent TM. Celui-ci informe les agents Sinks afin qu'ils réalisent une renégociation avec les usagers et un signal *Viol* est envoyé à l'agent Source. La récupération de la Qds est aussi effectuée au niveau de l'agent TM.

(iii) Un signal *Detect_Change* est envoyé à l'agent Source. Celui-ci ne fournit plus la qualité violée pendant une certaine période. Une opération de renégociation est effectuée avec les usagers afin qu'ils transitent vers une autre Qds.

7-3 Simulation des politiques de la persuasion

Lors de la simulation de la version originale du [Haf-97b], plusieurs anomalies ont été détectées. Ces anomalies ont été traitées d'une manière détaillée dans la dernière section du chapitre 4. L'amélioration du protocole a permis d'avoir une spécification correcte et cohérente.

SDT est un outil de simulation robuste et efficace pour simuler le comportement d'un protocole avec échange de signaux. Nous l'avons utilisé pour simuler le protocole de gestion de la qualité de service. Nous l'avons aussi utilisé pour simuler l'opération de persuasion. Mais, il ne permet pas de présenter les données sous forme graphique. Pour avoir des résultats très détaillés, le temps d'exécution est très long (plusieurs heures). En plus, la fonction aléatoire utilisée (écrite en code C) génère le même germe ce qui entraîne des suites de termes similaires. L'opération de persuasion consiste à exécuter des milliers d'opérations et d'utiliser des fonctions mathématiques d'où la nécessité d'utiliser un outil spécifique. Nous avons opté pour MATLAB afin de simuler l'opération de persuasion. Les commandes de MATLAB sont faciles puisqu'elles sont exprimées dans un langage similaire au langage utilisé dans les mathématiques. Le temps d'exécution est plus court et il dispose d'un ensemble de fonctions prédéfinies. Il permet aussi d'avoir des représentations graphiques qui résument visuellement les données.

Les politiques de persuasion meilleur profit (*Best_Profit*) et meilleure qualité de service (*Best_Qos*) ont été simulées avec MATLAB. Ces politiques sont présentées dans les sections 5-4-2-1 et 5-4-2-2 du chapitre 5 respectivement. Nous allons présenter ci-dessous les résultats obtenus après simulation de ces politiques de persuasion.

7-3-1 Exemple de l'utilisation de la politique Best_Profit

Cette politique consiste à diminuer les prix réguliers des qualités jusqu'à l'obtention d'un prix d'une qualité qui permet d'augmenter le profit du système. L'exemple de l'application de télé-éducation présenté dans le chapitre 4 section 4-1 permet d'avoir des résultats concrets sur la simulation en utilisant la politique Best_Profit.

L'arbre de diffusion associé à cette application est constitué d'un agent Source qui dispose d'un seul agent TM. Celui-ci dispose d'un millier d'agents Sinks. Un millier d'usagers sont connectés à cette application MM. La majorité de ces usagers ont choisi la qualité Q1 avec un pourcentage de 99,6%. La qualité Q2 n'a pas été choisie par aucun usager et 0,4% des usagers ont reçu la qualité Q3. Nous disposons également des informations suivantes :

- C = (1; 27; 37) où C(i) représente le coût par lien pour la qualité Qi et $i \in [1,2,3]$.
- Prix = (2.4; 64.8; 88.8) où Prix(i) représente le prix à payer par l'usager pour recevoir la qualité i. Le pourcentage de gain utilisé '*Gain*' est égal à 20% du coût.
- Les bornes des appréciations pour chaque qualité Q1, Q2, Q3 sont $a_1 \in [0, 30]$ et $a_2 \in]30, 60]$ et $a_3 \in]60, 90]$ respectivement.
- L'échantillon utilisé est constitué de mille usagers.
- L'unité de monnaie utilisée pour exprimer les prix, les coûts et les profits est désignée par *UM* (unité de monnaie).

Dans cette politique, nous allons comparer les résultats obtenus du profit initial et du profit estimé après la persuasion. La figure suivante résume la manière d'obtenir ces profits.

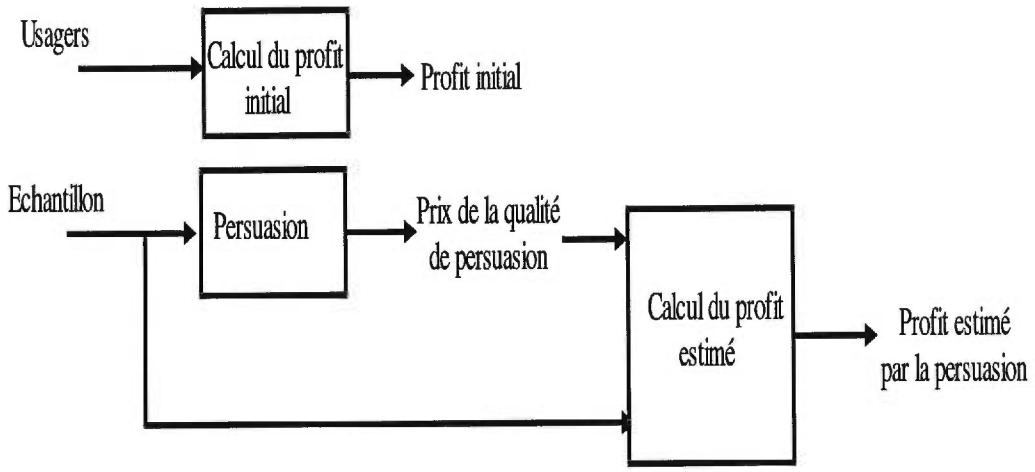


Figure 7.4. Schéma illustrant le calcul du profit initial et estimé

Après avoir réalisé la connexion de tous les usagers, l'agent TM attend une certaine durée pour s'assurer de la stabilité du système. Dans cette politique, nous disposons de deux méthodes pour présenter le prix de la qualité de persuasion:

(i) Présenter le prix réduit de la qualité de persuasion à tous les usagers d'une manière équitable.

(ii) Présenter le prix réduit de la qualité de persuasion à tous les usagers qui n'utilisaient pas cette qualité avant la persuasion. Les usagers, qui recevaient cette qualité avant la persuasion, ne vont pas bénéficier de cette réduction du prix.

Utilisation de la première méthode

En appliquant la première méthode sur les usagers de l'application de télé-éducation, nous obtenons les résultats suivants :

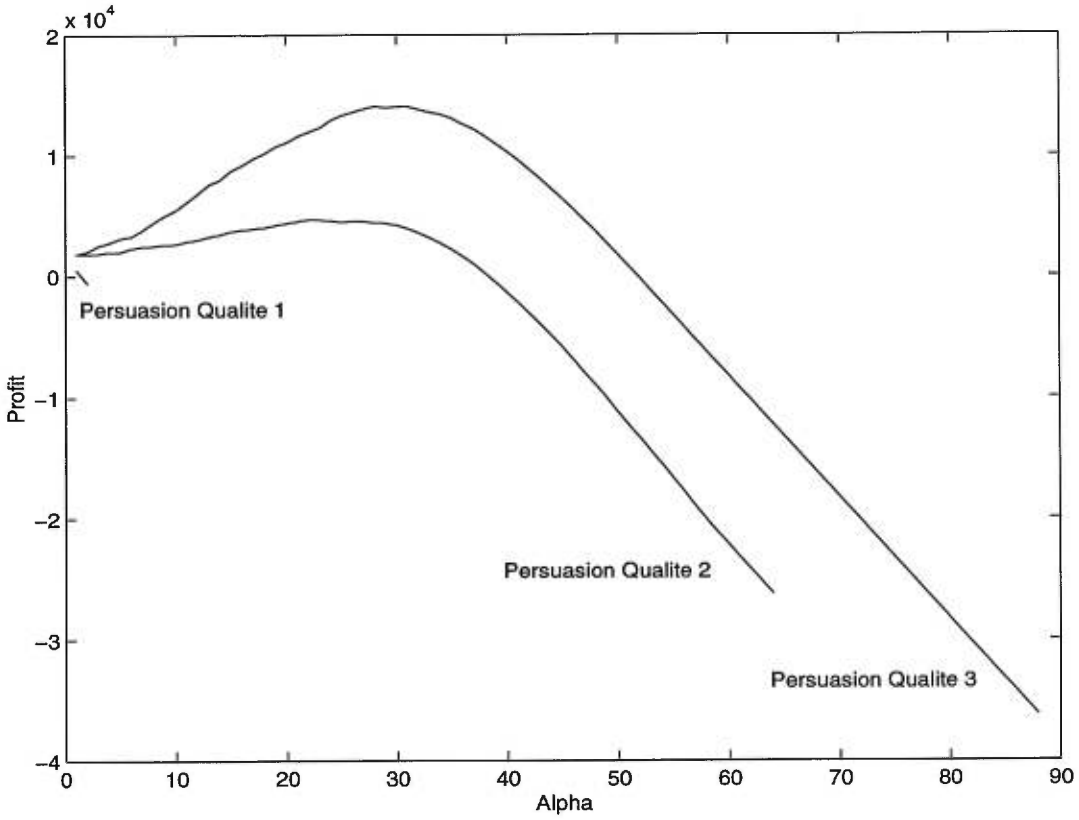


Figure 7.5. Résultats de la politique Best_Profit (méthode 1)

L'abscisse dans la figure 7.5 représente la variation de la variable *Alpha* (voir l'étape (iii) de la section 5-4-2-1 du chapitre 5). Cette variable représente la variation du *prix(i)* de la qualité i. L'ordonné représente le profit du système estimé par la persuasion d'une qualité donnée. La variation du profit est donnée en fonction de la variation de *Alpha*. Une augmentation de la valeur de *Alpha* permet une diminution du prix payé par l'usager pour obtenir la qualité Qi. Le résultat obtenu par chaque qualité de service est le suivant :

i) La courbe obtenue pour la qualité *Q1* est intitulée *'Persuasion Qualite 1'* :

Quand Alpha est nulle, le profit représente le profit du système sans appliquer la persuasion (*≈1563 UM*). Dans ce cas, 99.6% des usagers reçoivent la qualité Q1 puisque la différence entre l'appréciation des usagers pour cette qualité et le prix de Q1 (*2.4 UM*) est très grande par rapport aux autres qualités. L'augmentation de *Alpha* entraîne la diminution du prix du Q1 et par la suite, la diminution du profit du système. La courbe de *Alpha* de la qualité Q1 est très courte puisque le prix régulier de Q1 est petit (*2.4 UM*) et donc, *Alpha* atteint rapidement le prix régulier de Q1.

ii) La courbe obtenue pour la qualité *Q2* est intitulée *'Persuasion Qualite 2'* :

Quand Alpha est nulle, aucun usager n'a choisi la qualité Q2. Ceci est dû au fait que l'appréciation de chaque usager pour cette qualité est inférieure au prix régulier de cette qualité (*64.8 UM*). Donc, l'usager n'a aucun bénéfice à choisir la qualité Q2. La diminution du prix de Q2 par l'augmentation de *Alpha* permet d'augmenter le profit du système. Le profit maximal obtenu est égal à *5224 UM*. Ceci est dû au fait que les usagers transitent de la qualité Q1 vers Q2. Quand la majorité des usagers reçoivent la qualité Q2 (*Alpha ≈ 40*), le profit du système commence à diminuer à cause de la réduction du prix de Q2.

iii) La courbe obtenue pour la qualité *Q3* est intitulée *'Persuasion Qualite 3'* :

Quand *Alpha* est nulle, quatre usagers seulement parmi 1000 usagers reçoivent la qualité Q3 puisque le prix régulier de cette qualité est relativement cher (88.8 UM) par rapport au prix de la qualité Q1. L'augmentation de *Alpha* entraîne la diminution du prix à payer par l'usager pour recevoir Q3. Donc, plusieurs usagers transitent de la qualité Q1 vers la qualité Q3 puisque la différence entre les appréciations des usagers et les prix des qualités est plus grande pour la qualité Q3 (bénéfice net pour les usagers est plus grand). Le profit maximal obtenu en persuadant avec Q3 est égal à *14 826 UM*. Cette augmentation est de l'ordre de 800% du profit initial (sans persuasion *≈1563 UM*). Quand *Alpha* augmente (*≈ 50*), le profit du système commence à diminuer par rapport au profit initial à cause de la réduction du prix de Q3.

Analyse et discussion

Dans les cas suivants, nous allons modifier le prix régulier (présenté ci-dessus) de chaque qualité de service à la fois pour voir l'impact de cette modification sur le système.

(i) Modification du prix de Q1

a- Dans le cas où les prix des qualités sont (16.8; 64.8; 88.8) au lieu de (2.4; 64.8; 88.8) pour Q1, Q2 et Q3 respectivement, la répartition des usagers par rapport à ces qualités est égale à (77.9% ; 5.4%; 16.7%). L'augmentation du prix de la qualité Q1 par rapport à l'exemple 7-3-1 permet une augmentation du profit initial (*≈18 255 UM*). Ceci est dû au fait que certains usagers ont choisi les qualités Q2 et Q3 puisque le prix de la qualité Q1 a augmenté. La persuasion avec la qualité Q2 ou Q3 permet d'avoir une augmentation du profit du système car la majorité des usagers reçoivent la qualité Q1. Par exemple, le profit estimé obtenu en persuadant avec Q3 est égal à 25 972 UM. Cependant le profit net obtenu et qui représente la différence entre le profit maximal obtenu par la persuasion avec Q3 et le profit initial est plus grand dans l'exemple 7-3-1.

b- Dans le cas où les prix des qualités sont (36; 64.8; 88.8) au lieu de (2.4; 64.8; 88.8) pour Q1, Q2 et Q3 respectivement, la répartition des usagers par rapport à ces qualités est égale à (18.5% ; 24.6%; 56.9%). Le profit initial obtenu est égal à 42579 UM. À cause de l'augmentation du prix de la qualité Q1 par rapport à l'exemple (a), les usagers ont préféré plus Q2 et Q3. Étant donné que la majorité des usagers reçoivent la qualité Q3, la persuasion avec Q3 n'est pas bénéfique pour le système. Il est préférable que ces usagers paient le prix régulier de Q3 au lieu de payer un prix réduit.

(ii) Modification du prix de Q2

a- Si les prix des qualités sont (2.4; 4.8; 88.8) au lieu de (2.4; 64.8; 88.8), la répartition des usagers est égale à (0.1%; 99.9%; 0%). Le profit initial est égal à 2795 UM. La majorité des usagers ont choisi Q2 puisque le prix de Q2 est plus motivant que celui de Q1. La persuasion avec Q3 permet d'augmenter le profit du système (≈ *4078 UM*) puisque les usagers ne l'utilisent pas.

b- Si les prix des qualités deviennent (2.4; 26.4; 88.8) au lieu de (2.4; 64.8; 88.8), alors 29.3% reçoivent la qualité Q1 et 70.7% reçoivent la qualité Q2. Le profit initial est égal à 11 286 UM. La persuasion avec la qualité Q3 permet d'avoir un profit maximum (≈*14 984 UM*) puisque cette qualité n'est pas utilisée par les usagers. Nous remarquons l'augmentation du profit initial et du profit obtenu par persuasion avec Q3 par rapport au cas (a).

c- Si les prix des qualités deviennent (2.4; 36; 88.8) et donc le prix de Q2 a encore augmenté par rapport à (a) et (b), alors 58.4% des usagers choisissent la qualité Q1, 41.5% choisissent Q2 et seulement 0.1% des usagers préfèrent Q3. Le profit initial est égal à 9531 UM. La diminution du profit initial par rapport au cas (b) est dû au fait que la majorité des usagers reçoivent la qualité Q1 au lieu de Q2 dans (b). Il est possible dans cette situation de persuader avec Q2, le profit estimé obtenu est égal à 9880 UM. Alors que la persuasion avec la qualité Q3 permet d'avoir un profit plus grand (≈*15 461 UM*) puisque cette qualité est reçue par une minorité des usagers.

(iii) Modification du prix de Q3

Dans le cas où les prix des qualités sont (2.4; 64.8; 67.2) et donc une diminution du prix de Q3 par rapport à l'exemple de 7-3-1, alors 34.8% des usagers choisissent Q3 et 65.2% des usagers choisissent Q1. Ceci est dû à la diminution du prix de Q3. Le profit initial est égal à 14 528 UM. La persuasion avec la qualité Q3 permet d'obtenir une augmentation du profit (≈ *19 756 UM*).

À travers tous ces exemples, nous déduisons que :

(i) Le profit net obtenu et qui représente la différence entre le profit maximal obtenu par persuasion avec Q3 et le profit initial est plus grand dans l'exemple 7-3-1. Nous déduisons que le profit maximal est obtenu quand la majorité des usagers utilisent la qualité Q1 (profit initial est petit) et uniquement une minorité d'usagers reçoivent la qualité Q3. La persuasion avec la qualité Q3 donne un profit meilleur que celui obtenu par la persuasion avec Q1 ou Q2.

(ii) Si la majorité des usagers reçoivent une qualité Qi, alors la persuasion avec cette qualité Qi entraîne une diminution du profit du système. Il est plus avantageux pour le système que la majorité des usagers paient un prix régulier au lieu de payer un prix réduit.

L'augmentation du profit du système lors d'une opération de persuasion dépend de plusieurs facteurs. Parmi ces facteurs, les coûts de chaque qualité de service et la répartition des usagers selon ces qualités. Dans l'exemple suivant, les prix des Qds sont (24; 48; 72) et 9.6% des usagers reçoivent la qualité Q1, 28% reçoivent Q2 et 62.4% reçoivent Q3. Le profit initial est égal à 35 332 UM. La persuasion avec les différentes qualités entraîne une diminution du profit du système. Ceci est dû au fait que la majorité des usagers utilisent Q3 et donc, le profit initial est déjà grand.

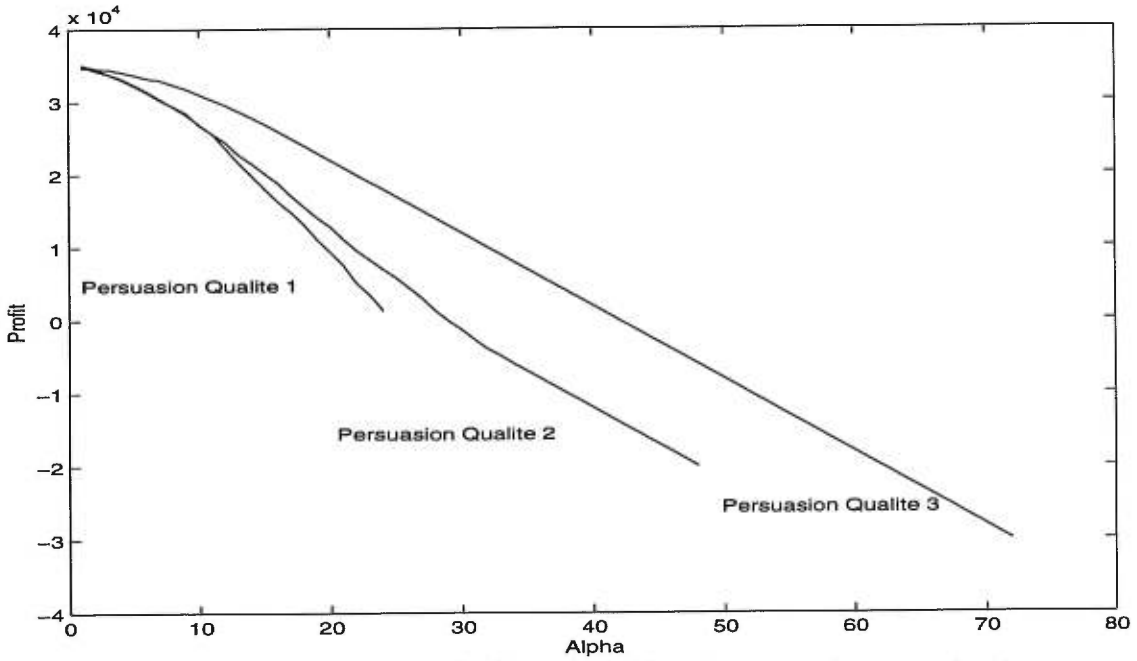


Figure 7.6. Exemple d'une opération de persuasion non réussie

Dans le cas réel, les usagers cherchent toujours à bénéficier d'une meilleure qualité avec un prix réduit. Ils cherchent à maximiser le bénéfice obtenu. L'existence de plusieurs compagnies qui offrent le même service et qui s'adresse à la même clientèle crée toujours une forte compétition entre ces compagnies. Celles-ci sont obligées de réduire les prix et de présenter souvent des rabais aux usagers pour garantir un certain pourcentage de la clientèle. Dans notre modèle, nous n'avons pas pris en considération la présence d'une compétition d'une autre compagnie de prestation de service.

b) Utilisation de la deuxième méthode

En appliquant la deuxième méthode (étape (ii) de 7-3-1) de la politique *Best_Profit* sur les usagers de l'application de téléconférence et en prenant en considération les mêmes données de 7-3-1, le profit obtenu par la persuasion de la qualité Q3 est presque similaire au profit obtenu avec l'application de la première méthode du *Best_Profit*. Ceci est dû au fait que la deuxième méthode de la politique *Best_Profit* considère que les usagers qui recevaient déjà la qualité de persuasion Q3, ne vont pas bénéficier de la réduction du prix. Or, uniquement 0.4% des usagers recevaient la qualité Q3 .

Donc, nous considérons l'exemple suivant où les prix des qualités sont (2.4; 64.8; 79.2). La diminution du prix de la qualité Q3 va permettre à certains usagers de choisir la qualité Q3. Ainsi, 89.8% des usagers utilisent Q1 et 10.2% reçoivent Q3. La figure suivante montre les résultats obtenus en appliquant une réduction équitable du prix de Q3 pour tous les usagers (méthode 1 du *Best_Qos*).

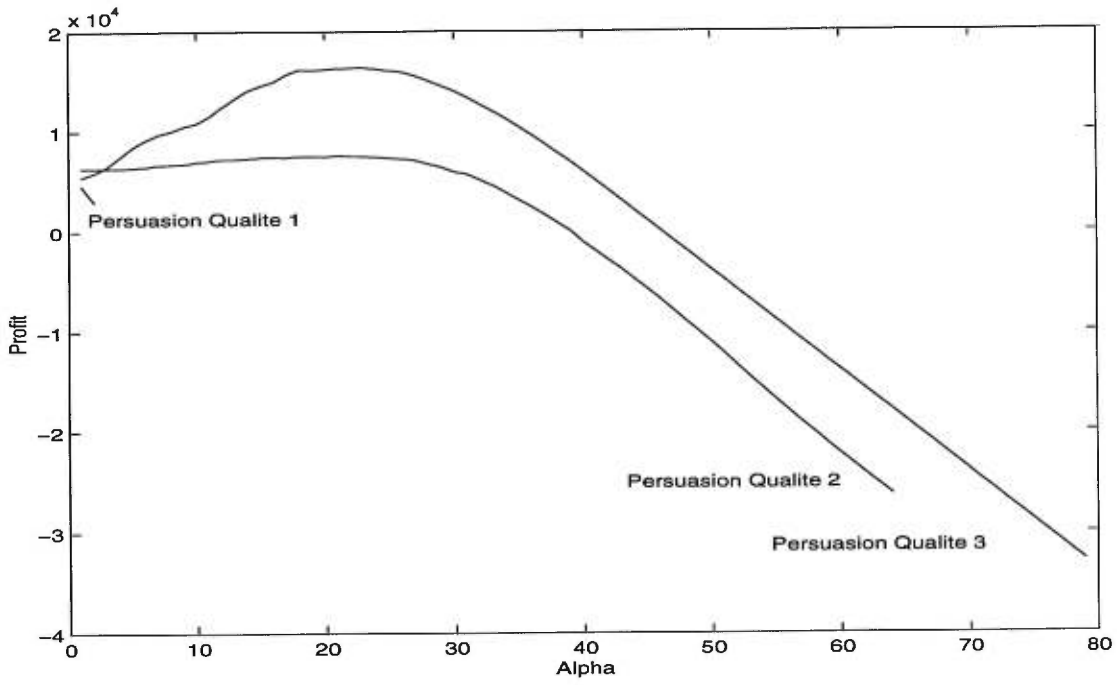


Figure 7.7. Exemple de la politique Best_Profit (méthode 1)

La figure suivante montre les résultats obtenus en appliquant une réduction non équitable du prix de Q3 pour les usagers (méthode 2 du *Best_Qos*).

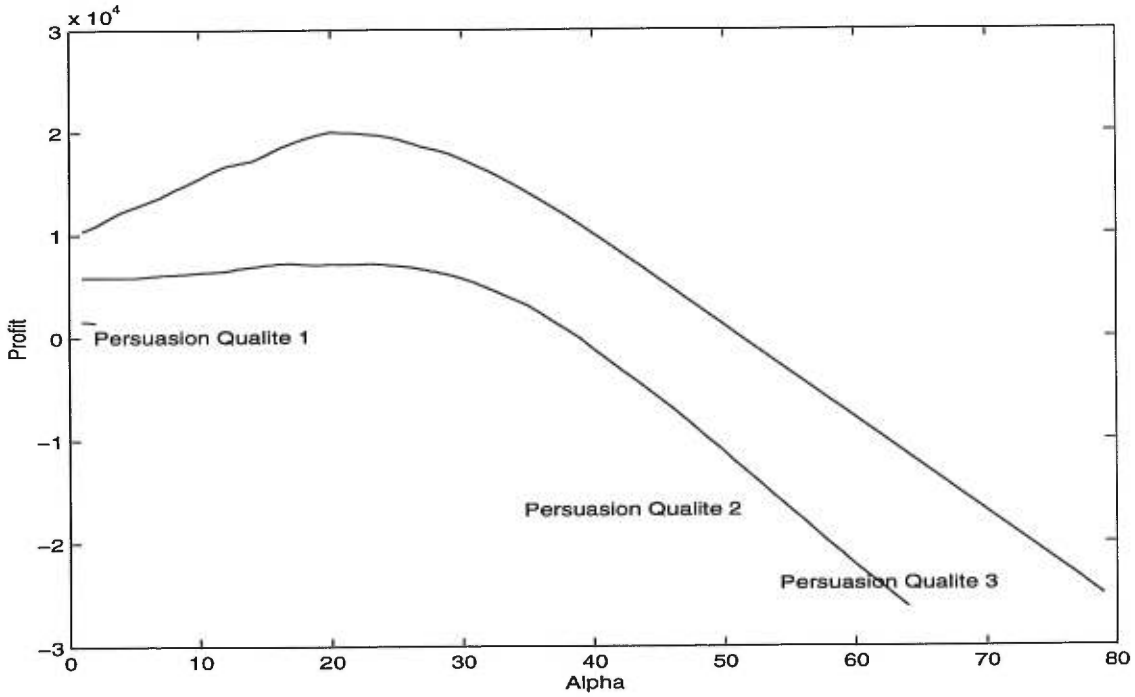


Figure 7.8. Résultats de la politique Best_Profit (méthode 2)

Le profit initial est égal à 5935 UM. Étant donné que la majorité des usagers reçoivent la qualité Q1, il est fort probable que la persuasion permettra d'augmenter le profit du système. En effet, la persuasion avec la qualité Q3, en utilisant la méthode 1 du *Best_Profit*, permet d'avoir un profit égal à 16 670 UM (voir figure 7.7). Alors que l'utilisation de la méthode 2 du *Best_Profit* permet d'avoir un profit égal à 19 750 UM (voir figure 7.8). Celui-ci est plus grand que le profit obtenu par la méthode 1 et le profit net obtenu est égal à 13 815 UM.

Cette méthode a comme avantage de faire bénéficier le système de l'augmentation du profit au lieu de partager les bénéfices de la persuasion d'une manière équitable entre les différents usagers.

En conclusion, la politique de persuasion *Best_Profit* permet d'aboutir à :

(i) Une augmentation du profit du système dans le cas où la majorité d'usagers recevaient la qualité la plus faible et uniquement une minorité d'usagers recevaient la meilleure qualité. Dans ce cas, le profit initial n'est pas très grand puisque le prix correspondant à la qualité la plus faible est normalement le moins cher. La persuasion avec la meilleure qualité permet de diminuer le prix de cette qualité. Par la suite, plusieurs usagers vont transiter de la qualité la plus faible vers la meilleure. Le partage des ressources réservées pour la réception de la meilleure qualité permettra une grande augmentation du profit du système.

(ii) Une diminution du profit du système, si la majorité des usagers recevaient la meilleure qualité. Dans ce cas, le profit initial du système est déjà grand. Il est difficile de réaliser une augmentation du profit du système par la persuasion des qualités moins bonnes. En plus, la persuasion avec la meilleure qualité n'est pas rentable. Il est plus bénéfique pour le système que la majorité des usagers paient le prix régulier de cette qualité au lieu de payer un prix réduit. Le prix régulier est plus grand que le prix réduit et donc, le profit du système sera plus grand.

7-3-2 Exemple de l'utilisation de la politique Best_Qos

Dans cette politique, la persuasion est réalisée pour chaque qualité de service et consiste à choisir la qualité qui permet d'avoir un profit supérieur ou égal au profit initial. Si cette qualité est trouvée, tous les usagers qui reçoivent une qualité moins bonne vont transiter vers cette qualité de persuasion sans payer plus. L'application de la politique *Best_Qos* sur les prix de l'exemple 7-3-1 entraîne une diminution du profit du système. Cette politique se caractérise par le fait que le revenu du système avant persuasion est le même que celui d'après persuasion. L'augmentation du profit du système nécessite donc une diminution des coûts après la persuasion puisque le revenu ne change pas. Ceci est difficile à réaliser.

Dans l'exemple suivant, nous considérons que les prix des qualités de service sont (57.6; 76.8; 86.8) au lieu de (2.4; 64.8; 88.8) de l'exemple 7-3-1. La majorité des usagers choisissent la qualité Q3 (99% d'usagers), alors que seulement 0.2% et 0.8% d'usagers choisissent Q1 et Q2 respectivement. Les autres informations de l'exemple 7-3-1 sont pris en compte sans aucune modification. Le profit initial est égal à 50 246 UM. Le profit par persuasion avec Q2 permet d'avoir 50 244 UM et celui de Q3 est égal à 50 258 UM. Nous réalisons la persuasion avec Q3 puisqu'elle garantie un profit similaire au profit initial et une libération des ressources des qualités Q1 et Q2.

Pour appliquer la politique *Best_Qos*, il suffit que le profit du système après persuasion augmente. Même si cette augmentation est faible (ou nulle) par rapport à la politique *Best_Profit*, cette politique permet de garantir une libération certaine des ressources pour toutes les qualités moins bonnes que la qualité de persuasion. Cette libération est certaine car nous avons supposé dans la section 5-4-2-2 du chapitre 5 que tous les usagers qui reçoivent une qualité moins bonne que la qualité de persuasion vont transiter vers cette qualité (pas de contrainte matérielle).

Cette politique est avantageuse dans le cas, par exemple, d'une application de vidéo-sur-demande. Nous supposons que plusieurs usagers vont assister à un film donné à un moment précis. Après avoir réalisé la connexion de tous les usagers, l'agent TM attend une certaine durée pour s'assurer de la stabilité du système. Normalement, tous les participants se connectent au début du film. Nous réalisons la persuasion avec les mêmes données présentées ci-dessus. La persuasion avec la qualité Q3 permet aux usagers qui reçoivent les qualités Q1 et Q2 de transiter vers la meilleure qualité Q3 sans payer plus et donc, ils sont satisfaits de cette

persuasion. Elle permet également une libération des ressources réservées pour les qualités Q1 et Q2.

En conclusion, la politique *Best_Qos* peut être appliquée pour libérer des ressources dans un réseau surchargé. Alors que la politique *Best_Profit* permet une augmentation du profit du système grâce au partage des ressources. L'utilisation de l'une de ces deux politiques *Best_Profit* ou *Best_Qos* dépend des objectifs du système.

7-4 Erreurs statistiques sur les résultats de la politique Best_Profit

Nous considérons l'exemple de télé-éducation présenté ci-dessus (section 7-3-1). Les paramètres utilisés ne subissent aucune modification. Dans l'exemple suivant, nous allons utiliser la politique *Best_Profit* avec diminution du prix de la qualité de persuasion d'une manière équitable entre tous les usagers (méthode 1). Nous allons calculer le profit initial des usagers. Nous calculons également le prix de la qualité de persuasion Q3, le profit estimé obtenu par persuasion avec Q3 et le profit effectivement réalisé après persuasion des usagers. La différence entre le profit estimé et celui réalisé (erreur de persuasion) est aussi calculée ainsi que le profit additionnel net obtenu grâce à la persuasion. Nous nous intéressons particulièrement aux erreurs statistiques. La figure suivante illustre la relation entre les différents calculs réalisés et les résultats obtenus.

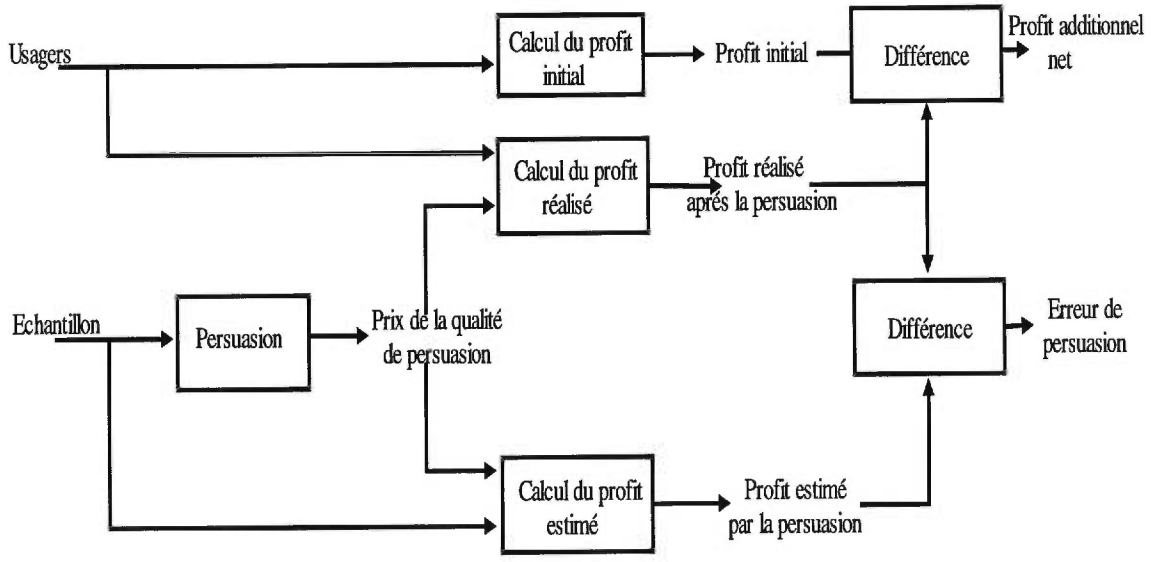


Figure 7.9. Schéma global de différents calculs réalisés

Nous avons opté pour deux méthodes de simulation. D'abord, nous considérons que le système des usagers est variable. Nous allons réaliser la même opération de persuasion 1000 fois afin d'aboutir à une bonne approximation. Pour chaque opération réalisée, nous calculons le profit initial des usagers puisqu'ils diffèrent d'une opération à une autre ainsi que toutes les autres données. La deuxième méthode consiste a choisir un seul système d'usagers (système fixe) et de faire varier l'échantillon plusieurs fois.

7-4-1 Simulation avec système d'usagers variable

Dans ce type de simulation, nous allons choisir plusieurs systèmes d'usagers différents. Pour chaque système d'usagers, nous disposons d'un échantillon. Le but de répéter cette opération plusieurs fois est de voir dans combien de cas, le profit est bien estimé. Étant donné que les appréciations de l'échantillon diffère de celles des usagers, nous obtenons un profit estimé qui diffère du profit réalisé. Si le profit réalisé est supérieur ou égal au profit estimé et ce dernier est supérieur au profit initial, l'opération de persuasion est réussie. Dans le cas contraire, si l'erreur de persuasion est grande, nous déduisons que l'estimation du profit n'était pas bonne. Donc, nous ne pouvons pas garantir une augmentation du profit du système après la persuasion. Selon la figure7.9, nous allons varier les usagers et l'échantillon plusieurs fois. Les différents résultats obtenus par la simulation sont les suivants :

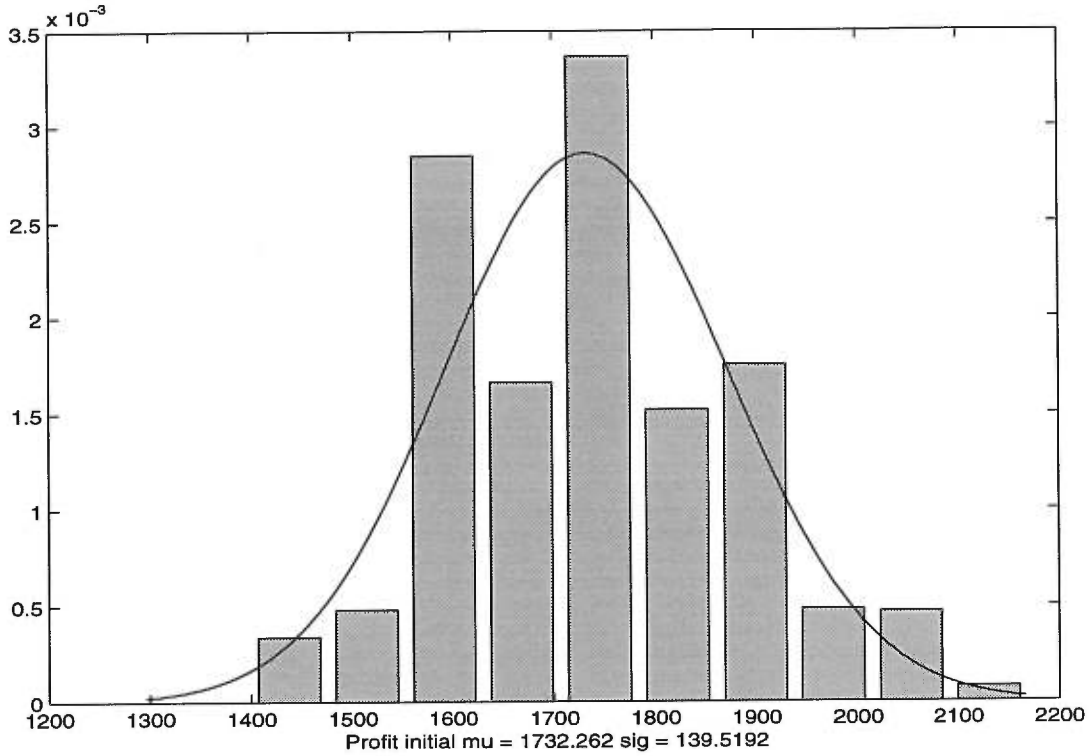


Figure 7.10. La distribution du profit initial

Profit initial

Selon l'histogramme obtenu (voir figure 7.10), la moyenne est de 1732.2 UM et l'écart type est de 139.5 UM. L'axe des abscisses représente les différents intervalles du profit initial obtenus, alors que l'axe des ordonnées représente la probabilité d'avoir un profit initial donné. La surface d'un rectangle représente la probabilité d'avoir un profit initial appartenant à l'intervalle correspondant. Ainsi, la somme de toutes les surfaces est égale à 1. Nous remarquons que dans la majorité des cas, le profit initial est très proche de la moyenne (l'écart type est petit). Le profit initial diffère d'un cas à un autre puisque les appréciations des usagers sont choisies d'une manière aléatoire. La majorité d'usagers choisissent la qualité Q1 et uniquement une minorité préfèrent recevoir la qualité Q3. La variation entre les pourcentages du nombre d'usagers qui reçoivent une qualité donnée entraîne cette variation du profit initial.

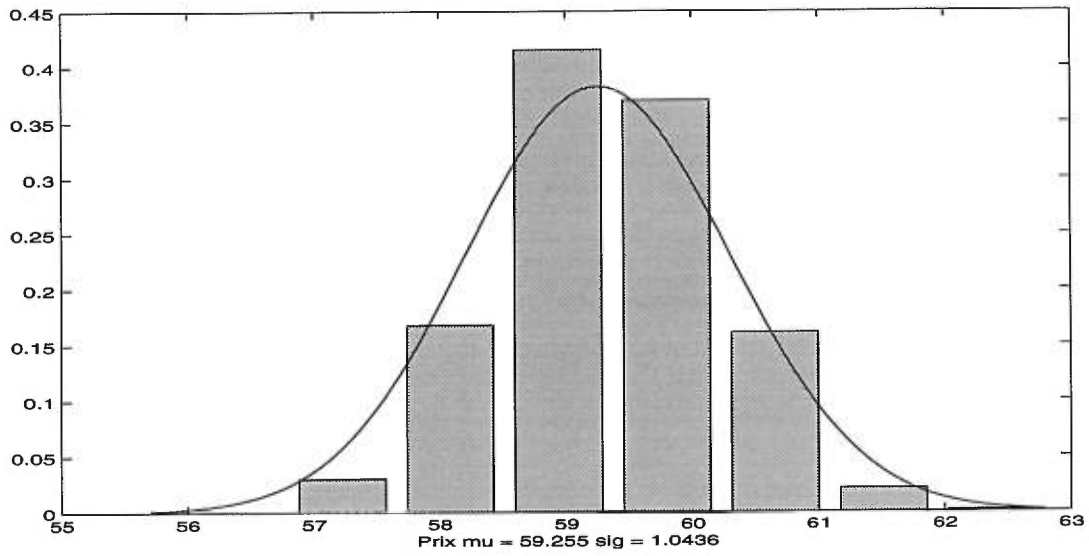


Figure 7.11. La distribution du prix de la qualité de persuasion

Prix de la qualité de persuasion

Selon l'histogramme obtenu (voir figure 7.11), la moyenne est de 59.2 UM et l'écart type est de 1 UM. L'axe des abscisses représente les intervalles du prix de la qualité de persuasion, alors que l'axe des ordonnées représente la probabilité de proposer un prix donné aux usagers. La surface d'un rectangle représente la probabilité de suggérer aux usagers un prix donné appartenant à l'intervalle correspondant. Ainsi, la somme de toutes les surfaces des rectangles est égale à 1. Les différents prix de la qualité de persuasion Q3 proposés aux usagers appartiennent à cet intervalle [56.8, 62.8]. Au fur et à mesure que la valeur de Alpha augmente, le prix de la qualité Q3 diminue. Le prix régulier de cette qualité est de 88.8 UM et Alpha est de valeur nulle au début de chaque opération de persuasion. Alpha est incrémenté successivement par la valeur 1 et le prix de Q3 devient le prix régulier moins la valeur de Alpha. Ceci explique pourquoi le prix de Q3 appartient à cet intervalle. En plus, le fait d'avoir un échantillon grand (1000 usagers) permet d'avoir un écart type petit et donc, une grande concentration du prix autour de la moyenne. Dans le cas d'un échantillon de 100 usagers ou 10, nous allons avoir comme écart type 2.2 et 5.4 respectivement. Les courbes dans ces cas sont plus larges et plus étendues. Donc, plus la taille de l'échantillon est petite, plus l'intervalle des prix de Q3 proposés aux usagers est plus grand. Par conséquent l'estimation du profit ne sera pas bonne.

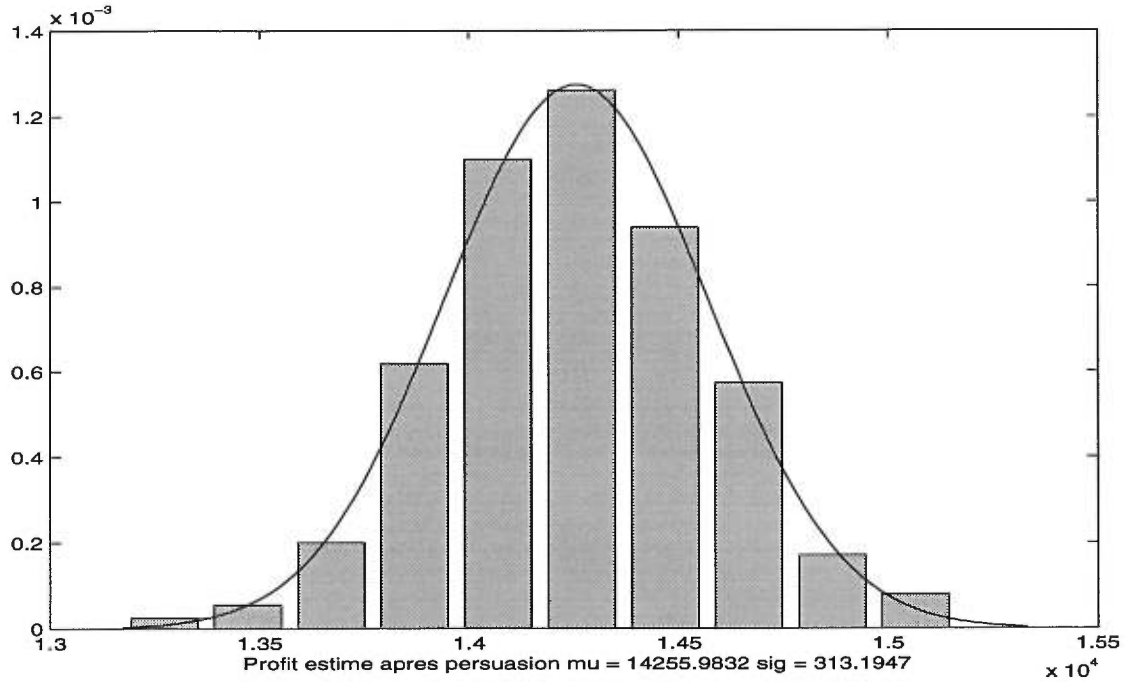


Figure 7.12. La distribution du profit estimé

Profit estimé après persuasion

La figure 7.12 montre que l'histogramme obtenu suit une loi normale. En effet, il y a une concordance entre les données obtenues 'histogramme' et la loi normale associée 'courbe continue'. La moyenne est égale à 14255.9 UM et l'écart type est égal à 313.1 UM. En fait, nous remarquons qu'il y a une grande concentration au tour de la moyenne. Le fait d'avoir un intervalle réduit des prix de la qualité de persuasion Q3 permet de réduire également l'intervalle de variation du profit estimé. Si la taille d'échantillon est égale à 100 ou 10, les courbes du profit estimé obtenues sont plus larges. Ceci est tout à fait normal puisque l'écart type devient plus grand. Selon la formule suivante :

Var(X) * 1000 = Var(Y) * 100, où X et Y représentent le profit estimé quand l'échantillon est égale à 1000 et 100 usagers respectivement et Var représente la variance.

Donc, Var (Y) = Var(X) * 1000 /100

Étant donné que $Var(X) = \sigma_x^2$ et $Var(Y) = \sigma_y^2$

Nous déduisons que : $\sigma_y = \sigma_x * (10)^{1/2} \approx 990.4$ UM

De la même manière, nous calculons $\sigma_z = \sigma_x * 10 \approx 3131.9$ UM, où Z représente le profit estimé pour un échantillon de 10 usagers. En réalisant des simulations sur des échantillons de 10 et 100 usagers, nous trouvons que l'écart type est égal à 3070.6 et 987.9 respectivement.

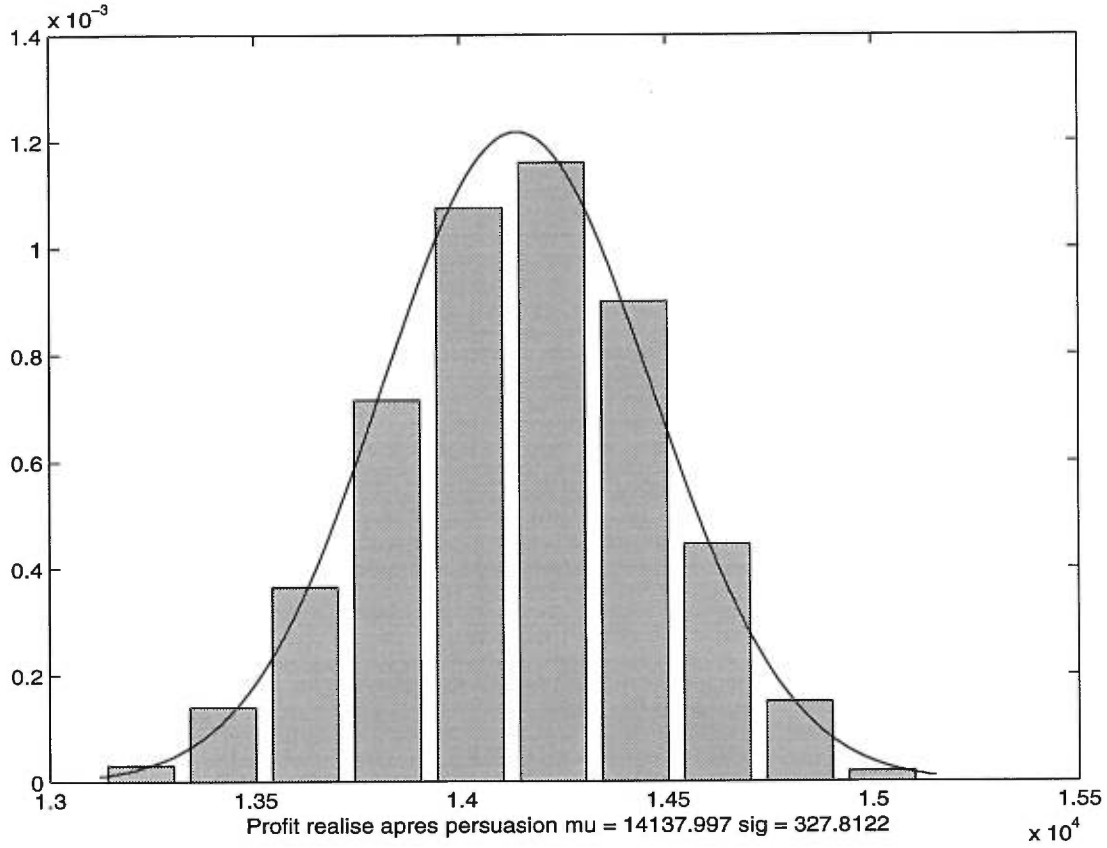


Figure 7.13. La distribution du Profit réalisé après persuasion

Profit réalisé après persuasion

La figure 7.13 montre que l'histogramme suit une loi normale. La moyenne est de 14137.9 UM et l'écart type est de 327.8 UM. La concentration des données autour de la moyenne est dû au fait que les prix de la qualité de persuasion proposés aux usagers réels du système appartiennent à un intervalle réduit. L'écart type du profit réalisé est presque égal à l'écart type du profit estimé puisque les usagers fictifs (échantillon) et ceux du système ont la même taille (1000 usagers). Dans le cas où l'échantillon est d'une petite taille (10 par exemple), l'écart type du profit réalisé devient plus grand et la courbe plus étendue. Dans ce cas, la variation du prix de Q3 proposé aux usagers est plus grande et donc, le profit réalisé varie plus.

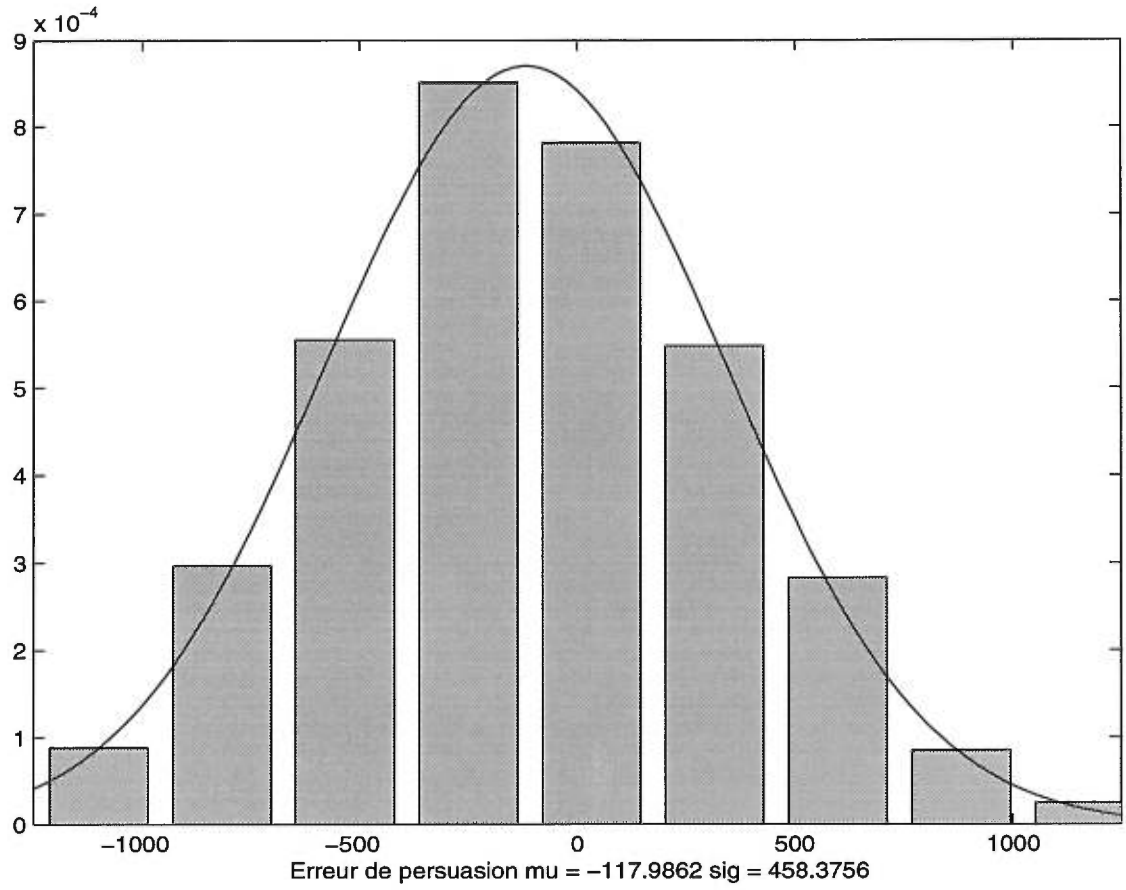


Figure 7.14. La distribution de l'erreur de persuasion

Erreur de persuasion

L'histogramme de la figure 7.14 représente la différence entre le profit réalisé et le profit estimé. La moyenne est égale à –117.9 UM et l'écart type est de 458.3 UM. Étant donné que le profit estimé et le profit réalisé sont indépendants puisque chaque système d'usagers est différent de l'échantillon, la moyenne de cette erreur est égale à la différence des moyennes du profit réalisé et du profit estimé (14137.9 – 14255.9) et l'écart type de cette erreur est calculé selon les formules suivantes :

Var(Erreur) = Var(Profit estimé) + Var(Profit réalisé) = -117.9 UM

Écart type(Erreur) = (Écart type(Profit estimé)^2 + Écart type(Profit réalisé)^2)$^{1/2}$ = 458.3 UM.

Selon la courbe de distribution de l'erreur, nous déduisons que dans 95% des cas, l'erreur de persuasion est concentrée autour de la moyenne dans un rayon de deux fois l'écart type de l'erreur [-1034.7 ; 798.7]. Dans le meilleur cas, le système gagne plus que 10% du

profit qu'il a estimé. Dans le pire cas, le système perd 12% de ce qu'il a estimé. Ceci signifie que le système réalise ce qu'il estime.

Dans les cas où la taille de l'échantillon est plus petite, la moyenne de l'erreur et l'écart type deviennent plus grandes. Si par exemple, la taille de l'échantillon est égale à 10 usagers, la moyenne est de –3368.6 UM et l'écart type est égal à 3539.4. Dans ce cas, les prix de Q3 proposés aux usagers varient beaucoup, ceci entraîne une grande variation du profit estimé et du profit réalisé. La figure suivante montre la distribution de l'erreur de persuasion dans le cas d'un échantillon de 10 usagers.

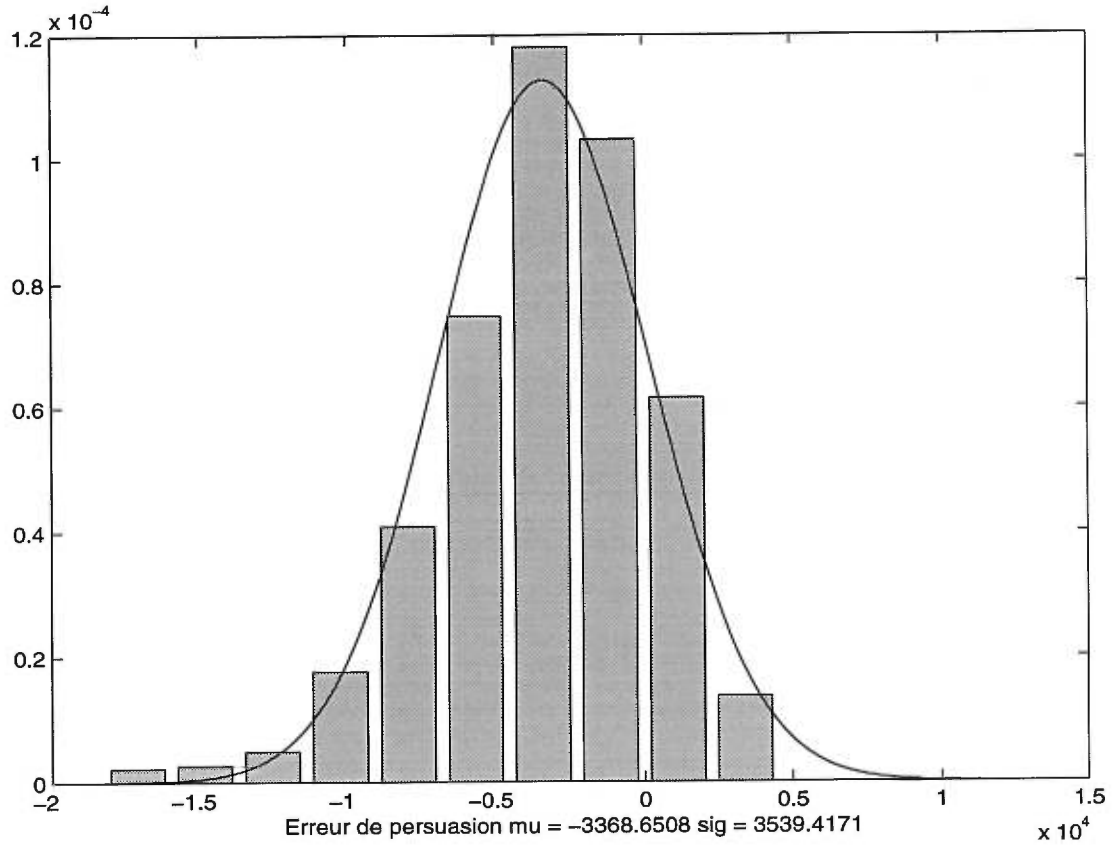


Figure 7.15. L'erreur de persuasion en utilisant un échantillon réduit

La grande différence entre le profit réalisé et le profit estimé est due à la mauvaise estimation du profit estimé. Selon ce cas, nous déduisons que si la taille de l'échantillon est inférieure à celle du système des usagers réels, l'estimation du profit n'est pas bonne et le profit réalisé n'est pas très proche du profit estimé.

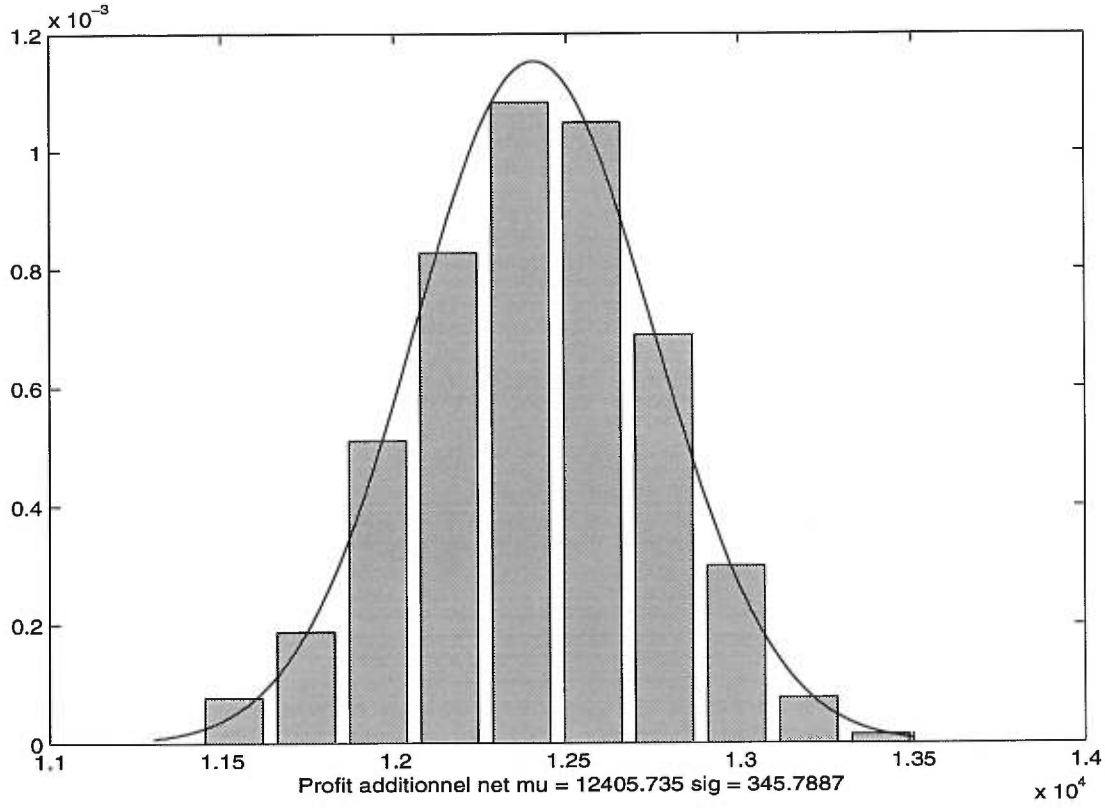


Figure 7.16. La distribution du profit additionnel net

Profit additionnel net

L'histogramme correspondant au profit additionnel net (voir figure 7.16) suit une loi normale. Ce profit représente la différence entre le profit réalisé après une persuasion et le profit initial. Cet histogramme est de moyenne de 12405.7 UM et d'un écart type de 345.7 UM. Selon l'histogramme, le système réalise 700% du profit initial en moyenne. Dans le pire cas, le système réalise uniquement 660% du profit initial. Dans le meilleur cas, le profit additionnel net représente 780% du profit initial. Il est clair que dans tous les cas, le système gagne avec la persuasion. Cette opération est toujours rentable pour le système. Ceci montre que le système gagne énormément par l'opération de persuasion. Dans le cas où l'échantillon est de 100 ou 10 usagers, la moyenne du profit additionnel net est égale à 12251.9 et 11337.9 respectivement et l'écart type est égal à 478.2 et 1618. L'augmentation de la moyenne du profit est inversement proportionnelle à la taille de l'échantillon. En fait, dans un échantillon de grande taille, le profit est bien estimé et proche du profit réalisé et donc, le profit additionnel net est plus grand.

7-4-2 Simulation avec système d'usagers fixe

Dans l'exemple suivant, nous allons considérer un seul système d'usagers fixe (1000 usagers) et nous allons varier l'échantillon plusieurs fois (1000 fois) (voir figure 7.9). Le but de cette simulation est de voir l'effet de l'échantillon sur un système d'usagers donné.

Dans cet exemple, le profit initial est égal à 1916 UM. Les différents prix de la qualité de persuasion Q3 présentés aux usagers appartiennent à cet intervalle [56.8, 62.8]. Le fait de changer l'échantillon entraîne une variation du profit estimé. La figure suivante montre la distribution du profit estimé.

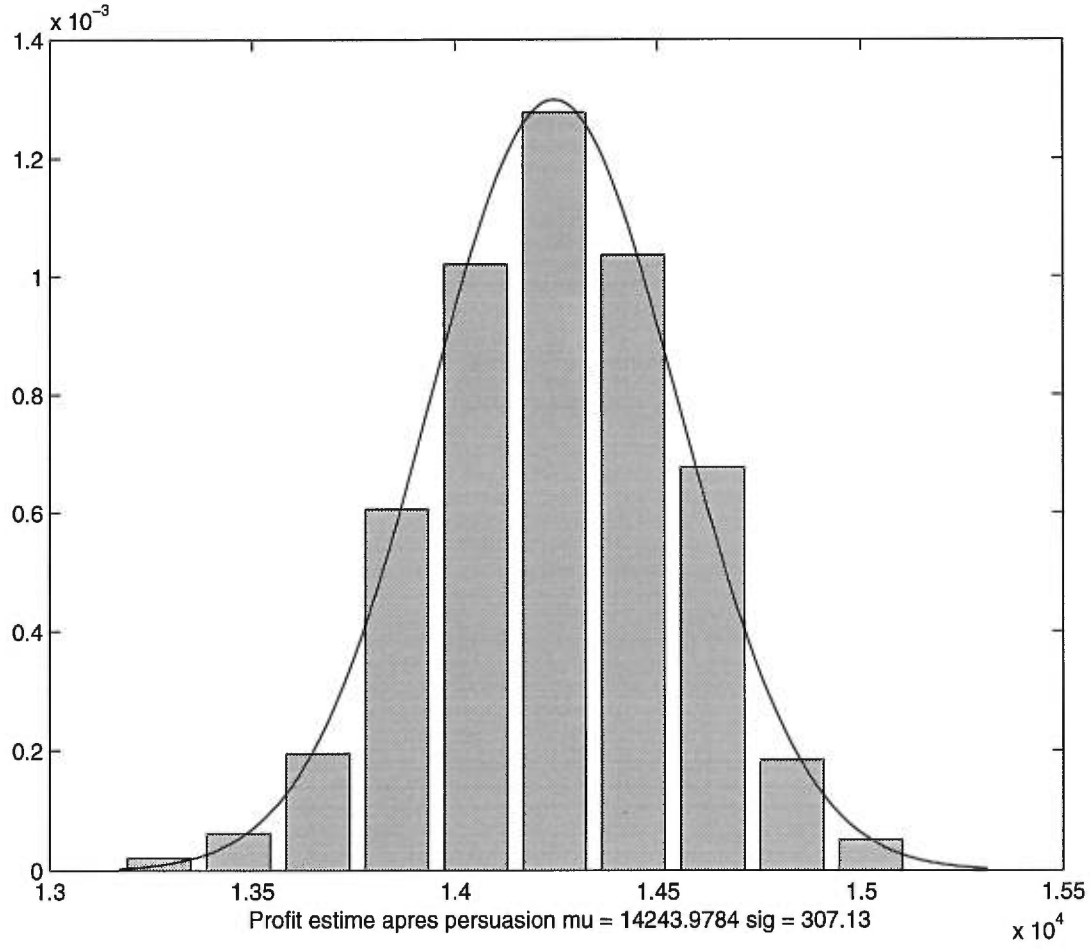


Figure 7.17. La distribution du profit estimé dans un système fixe

Cet histogramme montre que le profit estimé suit une loi normale. La moyenne est égale à 14 243.9 UM et l'écart type est 307.1 UM. Nous remarquons une grande concentration autour de la moyenne puisque l'écart type est petit et l'échantillon est grand.

La figure suivante montre la distribution du profit réalisé selon le prix de Q3. Nous disposons d'un seul système et donc, un seul profit est réalisé lorsqu'un seul prix de Q3 est proposé aux usagers du système.

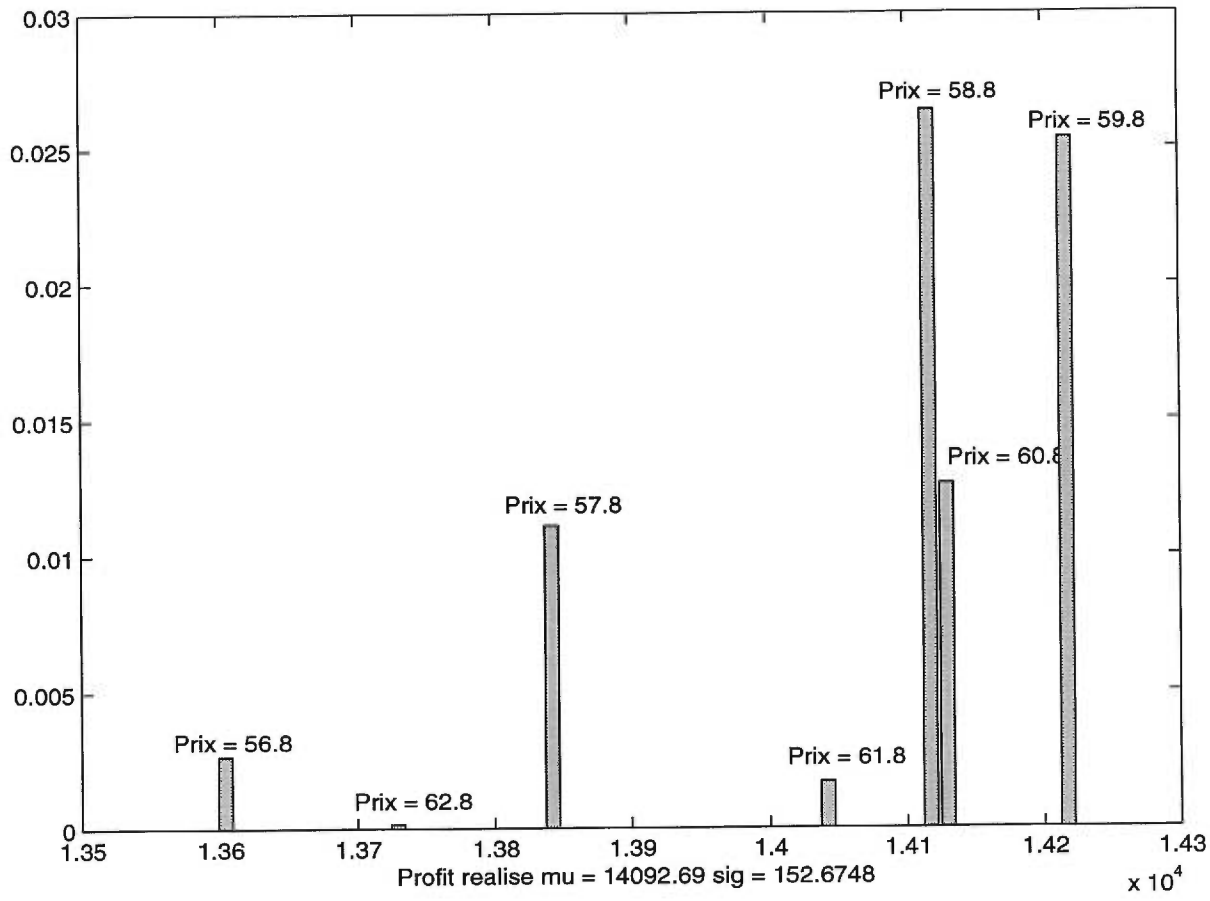


Figure 7.18. La distribution du profit réalisé dans un système fixe

Selon cette figure, nous remarquons que le profit réalisé atteint son maximum quand le prix de la qualité de persuasion Q3 proposé aux usagers est autour de la moyenne des prix (59.2UM). Au fur et à mesure que le prix s'éloigne de cette moyenne, le profit réalisé diminue progressivement. La variation du prix de la Q3 autour de la moyenne entraîne une légère diminution du profit réalisé. Cette diminution du profit réalisé est négligeable.

L'histogramme suivant (voir figure 7.19) montre que l'erreur de persuasion suit une loi normale. La moyenne est égale à –151.2 UM et l'écart type est égal à 311.4 UM. Ces données sont proches de celles de l'erreur de persuasion dans l'exemple précédent (système d'usagers variable).

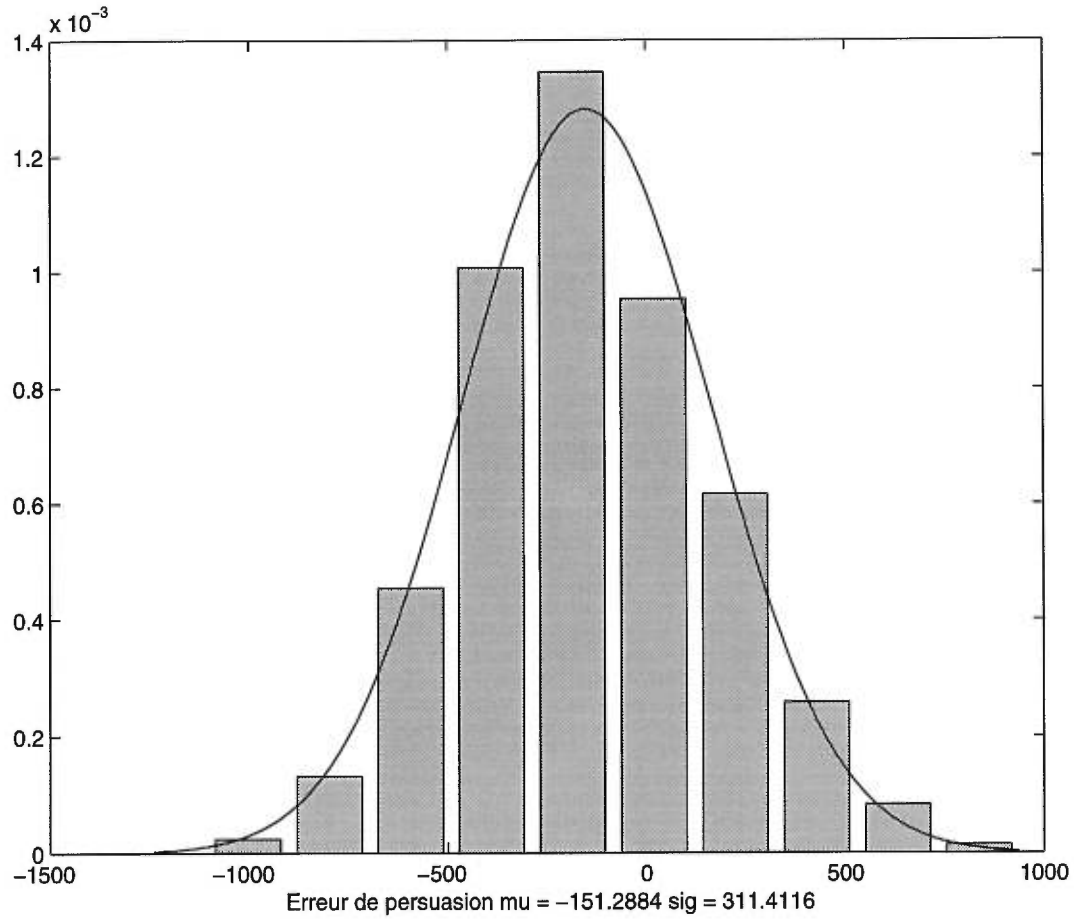


Figure 7.19. La distribution de l'erreur de persuasion dans un système fixe

Dans le cas où l'échantillon est de taille plus réduite, 100 usagers par exemple, nous obtenons une grande variation du prix de Q3. La courbe associée à la distribution du profit estimé est plus large et le profit réalisé n'est pas proche du profit estimé. La moyenne de l'erreur de persuasion est plus grande (-413.4 UM) et l'écart type est plus grand (999 UM). La figure suivante montre la distribution de l'erreur de persuasion dans le cas d'un échantillon de 100 usagers.

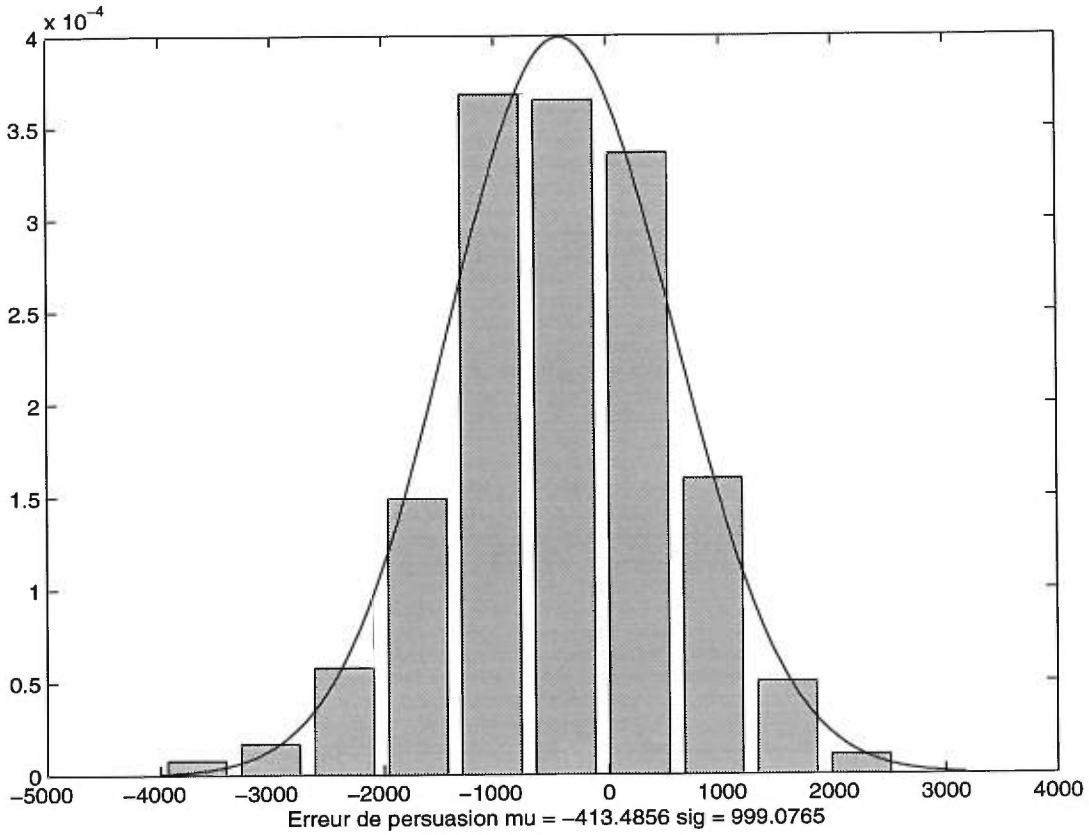


Figure 7.20. L'erreur de persuasion dans un système fixe avec un échantillon réduit

Dans le cas précédent d'un système fixe de mille usagers, nous allons fixer la valeur du prix de la qualité Q3 pour voir la variation du profit estimé. Étant donné que les appréciations des usagers des échantillons diffèrent, le profit estimé obtenu est différent lorsque nous proposons à cet échantillon un prix donné. La variation de ce profit pour chaque prix suit une loi normale. Le profit réalisé pour chaque prix de Q3 est fixe. L'erreur de persuasion dans ce cas est la différence entre le profit réalisé et la moyenne du profit estimé correspondante. Nous obtenons les résultats suivants :

Prix de Q3 (UM)	Moyenne du Profit estimé (UM)	Profit réalisé (UM)	Erreur de persuasion (UM)
56.8	13 973.3	13 598	-375.3
57.8	14 137.5	13 836	-301
58.8	14 251.2	14 112	-139.2
59.8	14 291.1	14 223	-68.1
60.8	14 280.1	14 130	-150.1
61.8	14 264.3	14 045	-219.3
62.8	14 355	13 733	-622

Table 7.1. La variation de l'erreur selon le prix de la qualité de persuasion

Selon la table 7.1, nous déduisons que lorsque le prix de Q3 est très proche de la moyenne des prix proposés (59.2 UM), l'erreur de persuasion est très proche de la valeur nulle. Donc, le profit estimé représente bien le profit réalisé et le système assure une certaine garantie sur l'augmentation du profit du système. Quand le prix de Q3 s'éloigne de la moyenne, l'erreur de persuasion subit une légère augmentation qui est négligeable. Malgré cette augmentation, le profit réalisé est très grand par rapport au profit initial.

Selon les différents résultats obtenus de l'exemple considéré, le profit n'est pas bien estimé quand la taille de l'échantillon est inférieure à la taille du système des usagers réels. Dans ce cas l'estimation du profit n'est pas bonne et le profit réalisé diffère du profit estimé. Le profit additionnel net obtenu n'est pas aussi grand. Ceci est dû à la grande variation des prix de la qualité de persuasion proposés aux usagers, ce qui entraîne une variation considérable du profit estimé et du profit réalisé. En plus, le fait d'avoir un échantillon et un système d'usagers de grande taille permet d'avoir une grande concentration du prix de la qualité de persuasion proposé aux usagers autour de la moyenne et donc, la variation du profit réalisé est négligeable et la persuasion est toujours rentable pour le système.

En conclusion, la persuasion permet la satisfaction des usagers puisqu'ils paient un prix réduit inférieur au prix régulier pour recevoir une qualité donnée. Elle permet aussi l'optimisation des ressources (libération ou partage) et donc, l'augmentation du nombre d'usagers servis par cette application ou par d'autres applications MM. La persuasion permet également une grande augmentation du profit du système.

Conclusion

Pendant les dernières années, plusieurs applications multimédias ont été développées. Ces applications nécessitent une grande capacité de transfert, une conservation des contraintes temporelles entre les informations, une diffusion multipoint des données et une garantie de service. La qualité de service est devenue actuellement très importante dans les applications multimédias. Le bon déroulement de ces applications nécessite une gestion efficace de la qualité de service.

Le protocole de gestion coopérative de la qualité de service dans les applications multimédias, développé à l'université de Montréal, permet de gérer les différentes qualités de service offertes aux usagers d'une manière distribuée. Ce protocole était écrit dans un langage naturel ce qui entraîne une confusion et une ambiguïté. Le but du présent travail est de spécifier formellement ce protocole. Nous avons utilisé le langage de description formelle SDL pour aboutir à une spécification claire, précise et non ambiguë. Nous avons également amélioré le protocole puisqu'il souffrait de plusieurs anomalies. Ensuite, nous l'avons simulé et validé.

Étant donné que la notion du coût est importante dans les applications multimédias, nous avons modélisé le comportement de l'usager pour choisir une qualité donnée parmi l'ensemble des qualités disponibles. Nous avons développé des stratégies de persuasion. Ces stratégies consistent à persuader des usagers à transiter vers une qualité de service particulière. La persuasion permet de libérer des ressources, d'augmenter le profit du système et de mieux satisfaire les usagers. Nous avons développé également des stratégies pour calculer les prix des qualités de service après avoir réalisé une persuasion et pour exécuter d'autres opérations de persuasion.

Nous avons présenté également les résultats de la simulation du protocole de gestion coopérative de la qualité de service en utilisant l'outil SDT. Les résultats de la simulation des politiques de persuasion ont été obtenus par l'utilisation de l'outil MATLAB. Ces résultats ont prouvé le bon fonctionnement du protocole ainsi que l'utilité et l'importance de la persuasion.

La persuasion est très importante et elle permet d'assurer plusieurs avantages. C'est un nouveau domaine et nécessite plusieurs recherches. Nous suggérons le développement d'autres politiques de persuasion au niveau Tree_Manager et au niveau Source et de les comparer.

Nous suggérons également d'améliorer le protocole de gestion coopérative de la qualité de service afin qu'il réalise d'autres fonctionnalités de gestion telles que : la spécification des paramètres de la Qds, le mappage, le contrôle d'admission, la réservation des ressources, le contrôle de la Qds, la facturation, etc. Ainsi, ce protocole peut devenir complet. La mobilité des agents est aussi un point important à traiter afin de fournir une distribution flexible des agents à travers le réseau.

Bibliographie

[Alf-98] Marco Alfano, *Design and implementation of a cooperative multimedia environment with Qos control*, Computer Communications 21, Pages 350-361, 1998.

[Aur-98] Cristina Aurrecoechea, Andrew T. Campbell et Linda Hauw, *A Survey of Qos Architectures,* ACM/Springer Verlag Multimedia Systems, Vol. 6 No. 3, Pages 138-151, Mai 1998.

[Ban-96] Anindo Banerjea, Domenico Ferrari, Bruce A. Math, Mark Moran, Dinesh C. Verma et Hui Zhang, *The Tenet Real-Time Protocol Suite : Design, Implementation, and Experiences*, IEEE/ACM Transactions on Networking, Vol. 4 No. 1., Février 1996.

[Bel-89] Ferenc Belina et Dieter Hogrefe, *The CCITT-Specificattion and Description Language SDL,* Computer Networks and ISDN Systems, Pages 311-341, 1989.

[Bel-91] Ferenc Belina, Dieter Hogrefe et Amardeo Sarma, *SDL with Applications from Protocol Specification,* Édition Prentice Hall, 1991.

[Bes-94] L. Besse, L. Dairaine, L. Fedaoui, W. Tawbi et k. Thai, *Towards an Architecture for Distributed Multimedia Application Support,* Proc. *International Conference on Multimedia Computing and Systems,* Boston, Mai 1994.

[Boc-97] Gregor V. Bochmann et Abdelhakim Hafid, *Some Principles for Quality of service Management*, Distributed Systems Engineering Journal 4, Pages 16-27, 1997.

[Bra-94] Braden R., D. Clark et S. Shenker, *Integrated Services in the Internet Architecture : an Overview*, RFC-1633, 1994.

[Cam-94] Andrew Campbell, Geoff Coulson et David Hutchison, *A Quality of Service Architecture*, ACM SIGCOMM Computer Communication Review, Vol. 24 No.2, Pages 6-27, Avril 1994.

[Cam-97] Andrew T. Campbell et Geoff Coulson, *Qos Adaptive Transports : Delivering Scalable Media to the Desktop*, IEEE Network, Vol. 12. No.2., Pages 18-27, Mars/Avril 1997.

[Cla-87] D. Clark, M. Lambert et L. Zhang, *NETBLT : A High Throughput Transport Protocol*, SigComm Workshop, Pages 335-359, 1987.

[Dab-96] Walid Dabbous et Jean Bolot, *Applications Multimédia sur l'Internet*, Colloque Francophone d'Ingénierie des protocoles (CFIP'96), Rabat, Maroc, Octobre 1996.

[Dia-97] Michel Diaz et Philippe Owezarski, *From Multimedia models to multimedia transport protocols*, Computer Networks and ISDN Systems, Vol. 29 N. 7, Aout 1997.

[Fae-94] Ove Faergemand, Anders Olsen, *Introduction to SDL-92*, Computer Networks and ISDN Systems 26, Pages 1143-1167, 1994.

[Fer-92] Domenico Ferrari, Anindo Banerjea et Hui Zhang, *Network Support For Multimedia*, Rapport technique TR-92-072, International Computer Science Institute, Berkeley, CA, Octobre 1992.

[Fis-97] Stefan Fischer, Abdelhakim Hafid, Gregor V. Bochmann et Hermann de Meer, *Cooperative Qos Management for Multimedia Applications*, International Conference on Multimedia Computing and Systems (ICMCS'97), Ottawa, Canada, Pages 303-310, Juin 1997.

[Ful-98] Errin W. Fulp, Maximilian Ott, Daniel Reininger et Douglas S. Reeves, *Paying for Qos : An Optimal Distributed Algorithm for Pricing Network Resources*, International Workshop on Quality of Service, Napa, CA, May 1998.

[Haf-96a] Abdelhakim Hafid, Gregor V. Bochmann et Brigitte Kerhervé, *A Quality of Service Negociation Procedure for Distributed Multimedia Presentational Applications*, High Performance of Distributed Processing, Syracuse, USA, 1996.

[Haf-96b] Abdelhakim Hafid, *Gestion de la Qualité de service dans les applications Multimedia,* Thèse de Doctorat, L'Université de Montréal, Montréal, Canada, Août 1996.

[Haf-97a] Abdelhakim Hafid et Gregor V. Bochmann, *A General Framework For Quality of Service Management*, Multimedia Tools and Applications, 1997.

[Haf-97b] Abdelhakim Hafid et Stefan Fisher, *A Multi-agent Architecture for Cooperative Quality of service Management*, International Conference on Management of Multimedia Networks ans Services (MMNS'97), Montréal, Canada, juillet 1997.

[Haf-98] Abdelhakim Hafid, Gregor V. Bochmann et Rachida Dssouli, *Distributed Multimedia Applications and Quality of Service,* Electronic Journal on Network and Distributed Processing, No. 6, Pages 1-50, Février 1998.

[Hua-97] Jean-François Huard et Aurel A. Lazar, *On End-To-End Qos Mapping*, IFIP, New York, NY, Mai 1997.

[Hut-95] David Hutchison, Geoff Coulson, Andrew Campbell et Gordon S. Blair, *Quality of Service Management in Distributed Systems*, Network and Distributed Systems Management, Chapitre 11, Édition Moris Sloman, Mai 1995.

[ISO-84] ISO 7498, *Basic Reference Model for Open Systems Interconnection*, 1984.

[ISO-95] ISO, *Quality of Service Framework*, International Standards Organization, UK, 1995.

[ITU-89] CCITT 1.350, *General Aspects of Quality of Service and Network Performance in Digital Networks*, International Telecommunication Union, Genève, 1989.

[Jen-96] N. Jennings et M. Wooldridge, *Softwarw Agents*, IEEE Review, Pages 17-20, 1996.

[ODP-95] ITU-T Recommendation X.903, *Reference Model for Open Distributed Processing. Part 3 : Prespective Model,* ITU-ISO, Genève, 1995.

[Pla-95] Thomas Plagemann, Knut A. Saethre, et Vera Goebel, *Application Requirements and Qos Negociation in Multimedia Systems*, Second Workshop on Protocols for Multimedia Systems, Slazburg, Australia, Octobre 1995.

[Rac-91] RACE IBC CFS D150, *General Aspects of Quality of Service and System Performance in IBC*, RIC, 1991.

[Rup-98] Björn Rupp, Richard Edell, Harish Chand et Pravin Varaiya, INDEX : A Platform for Determining how People Value the Quality of their Internet Access, International Workshop on Quality of Service, Napa, CA, May 1998.

[Sal-96] Yahya Y. Al-Salqan et Carl K. Chang, *Temporal Relations and Synchronization Agents,* IEEE Multimedia No2, Pages 30-39, 1996.

[Sar-96] A. Sarma, *Introduction to SDL-92,* Computer Networks and ISDN Systems 28, Pages 1603-1615, 1996.

[Tha-95] Kim-Loan Thai, Véronique Vèque et Simon Znaty, Architecture des réseaux haut débit, édition Hermes, 1995.

[Ven-97] Nalini Venkatasubramanian, Klara Nahstedt, *An Integrated Metric for Video Qos,* Proc. ACM Multimedia, 1997.

[Vog-95] Andreas Vogel, Brigitte Kerhervé, Gregor V. Bochmann et Jan Gecsei, *Distributed Multimedia and Qos : A Survey*, IEEE Multimedia, Vol. 2, No. 2, 1995.

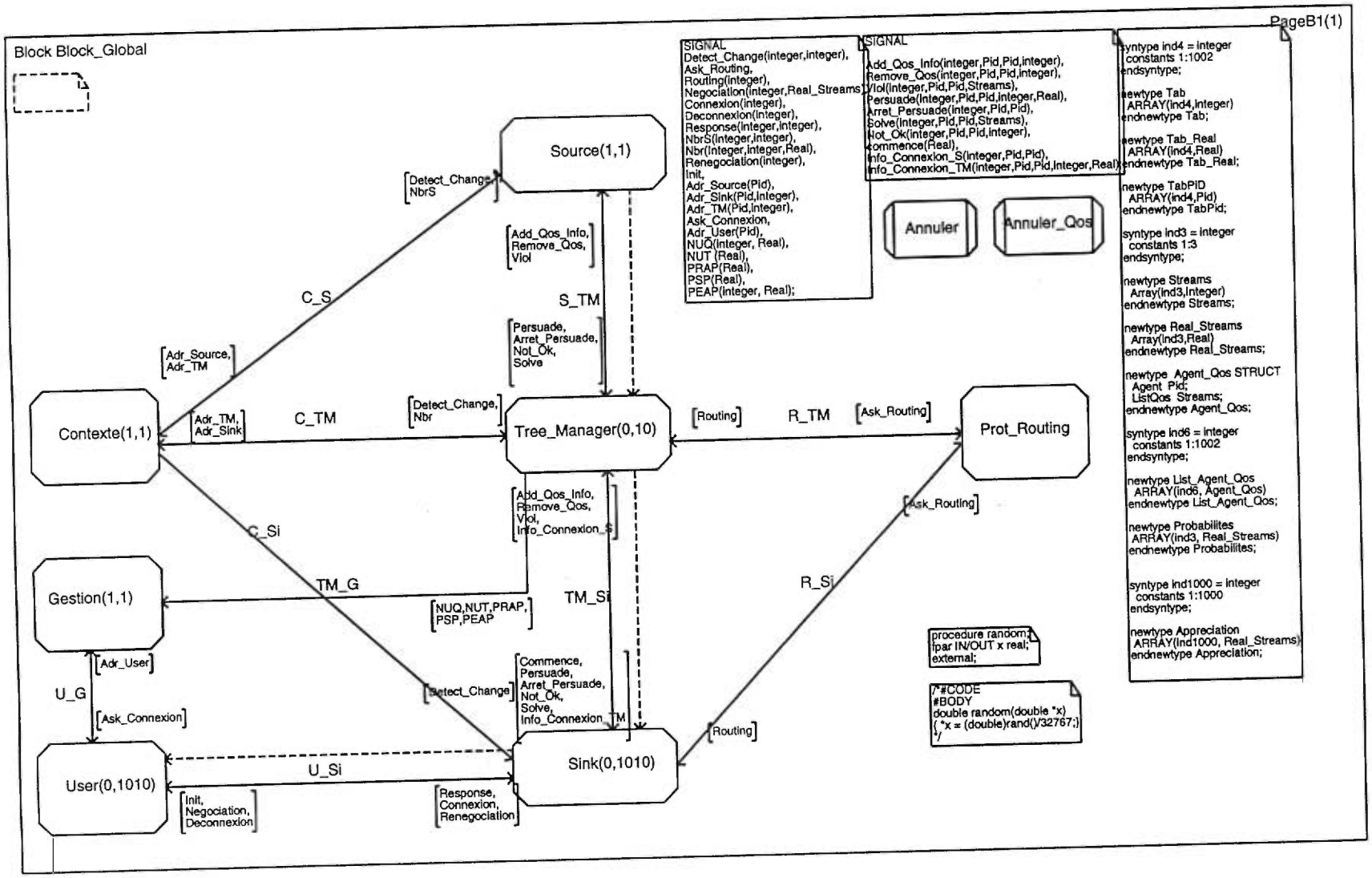
[Woo-94] M. Wooldridge et N. Jennings, *Agent Theories Architectures and Languages : A Survey*, Proc. of ECAI-Workshop, Pays bas, 1994.

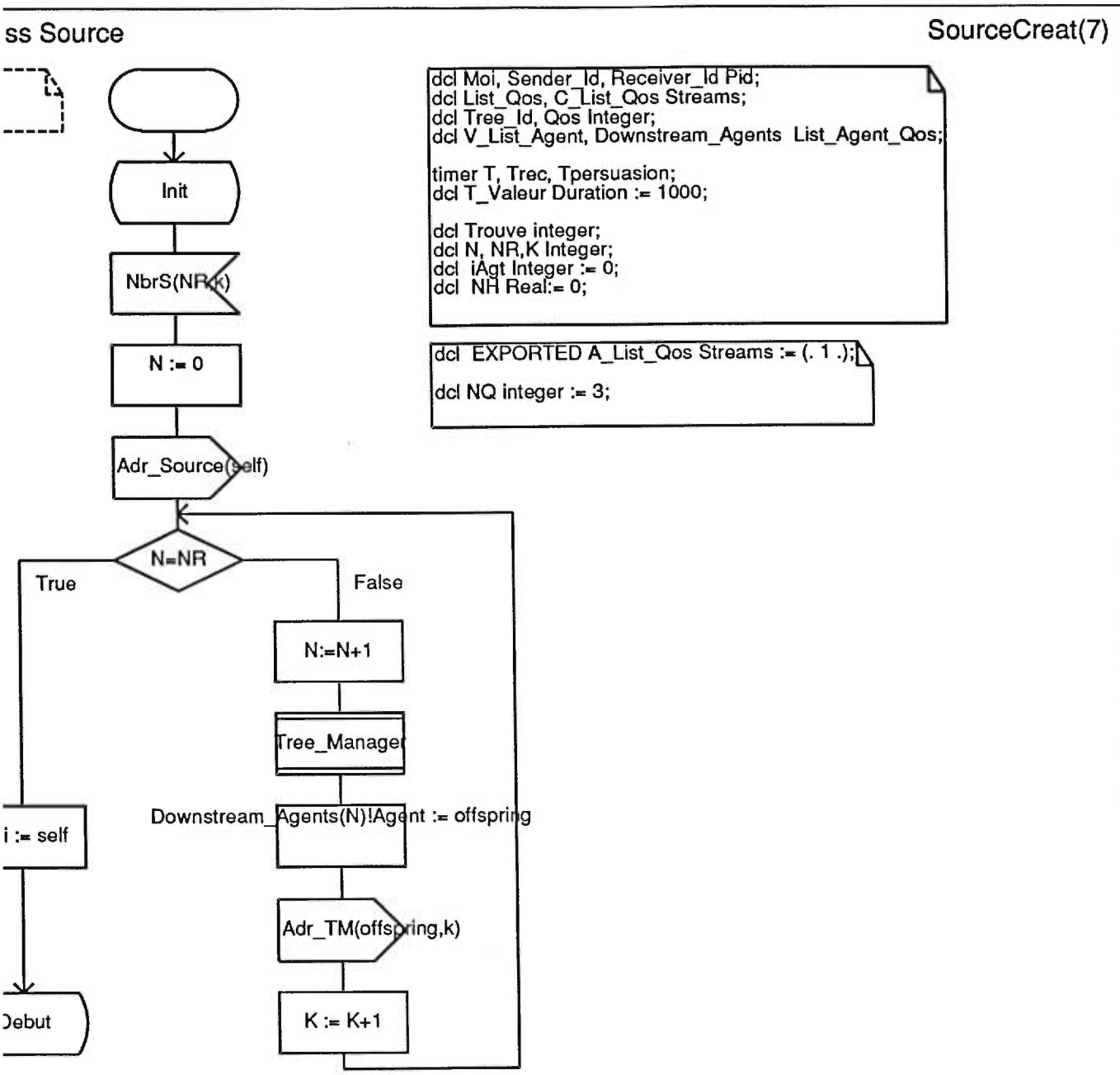
[Wro-97] J. Wroclawski, *The Use of RSVP with IETF Integrated Services*, RFC-2210, 1997.

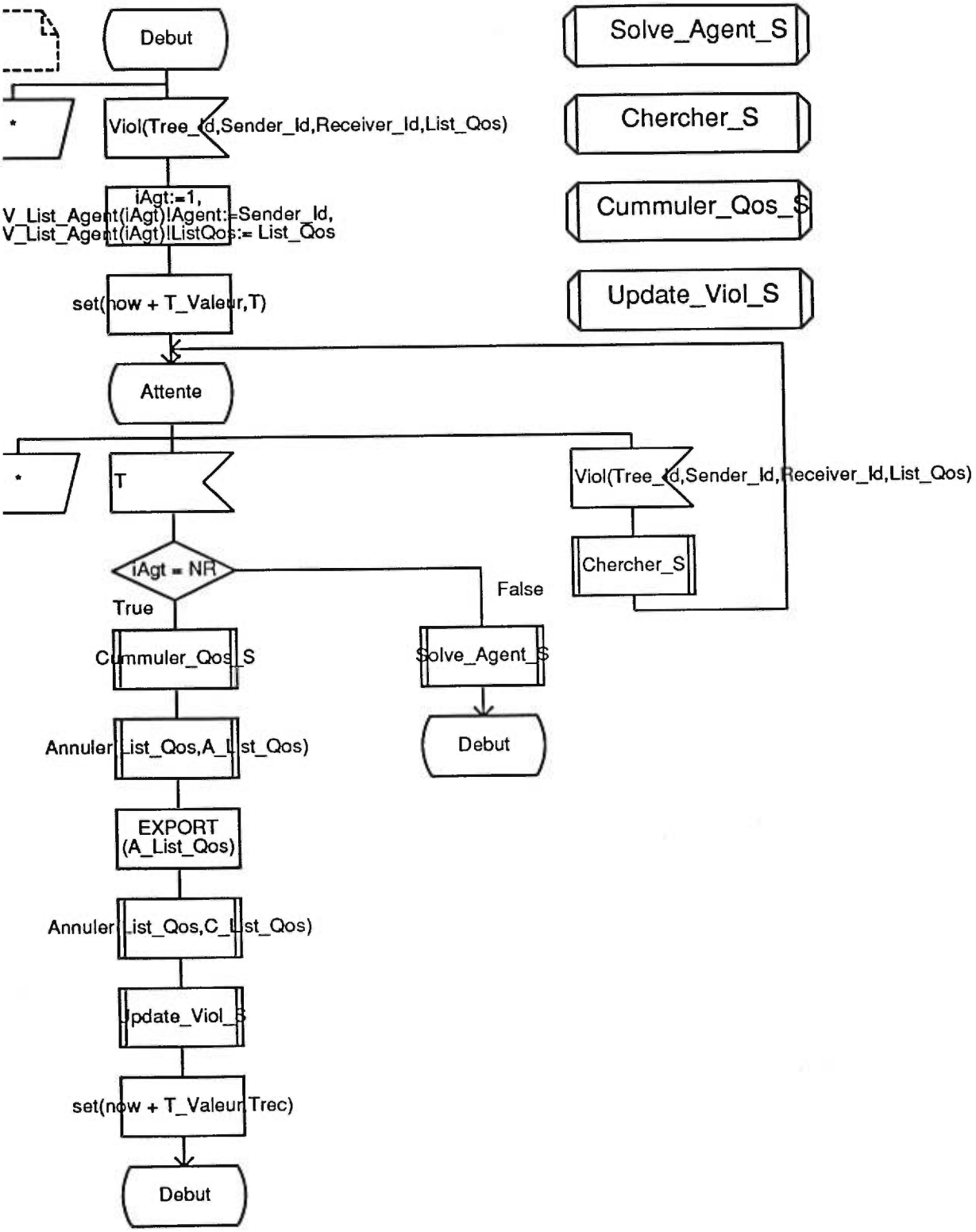
[Zha-93] Lixia Zhang, Steve Deering, Deborah Estrin, Scott Shenker et Daniel Zappala, *RSVP : A New Resource ReSerVation Protocol*, IEEE Network, Septembre 1993.

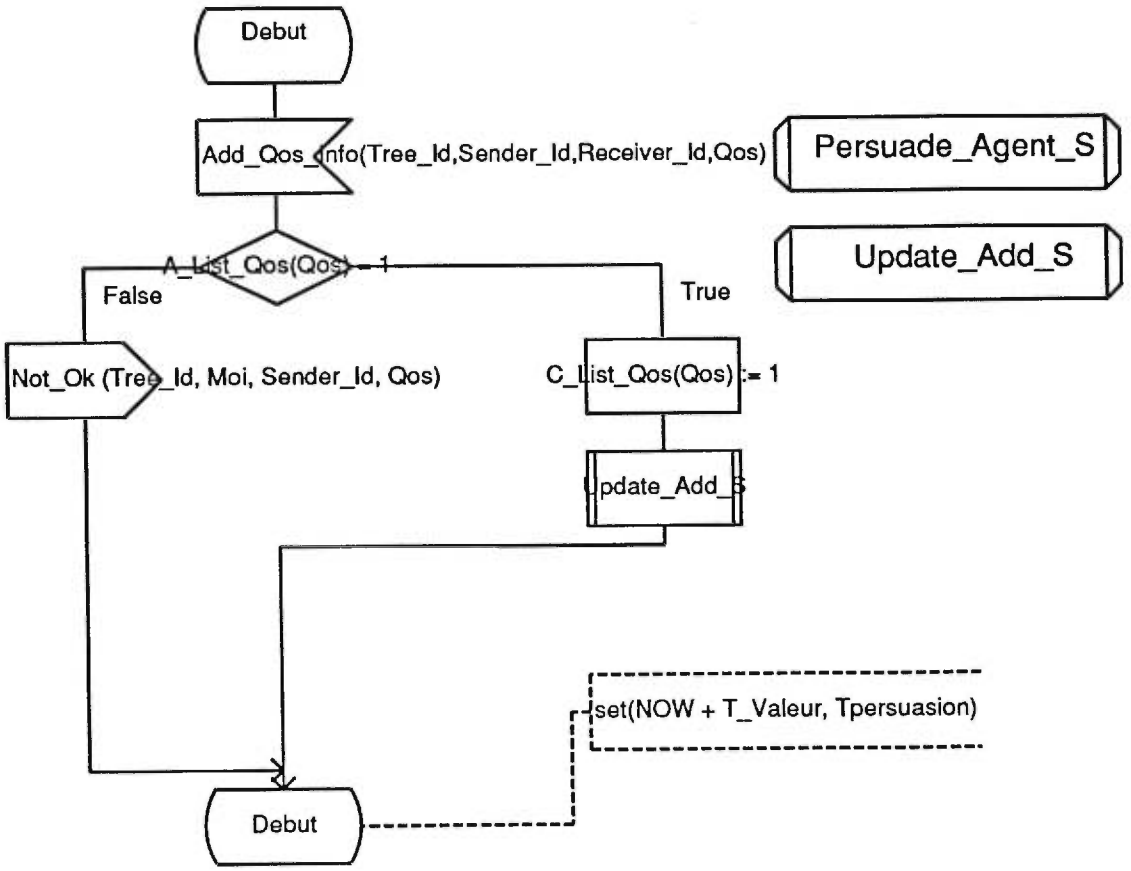
Annexe A

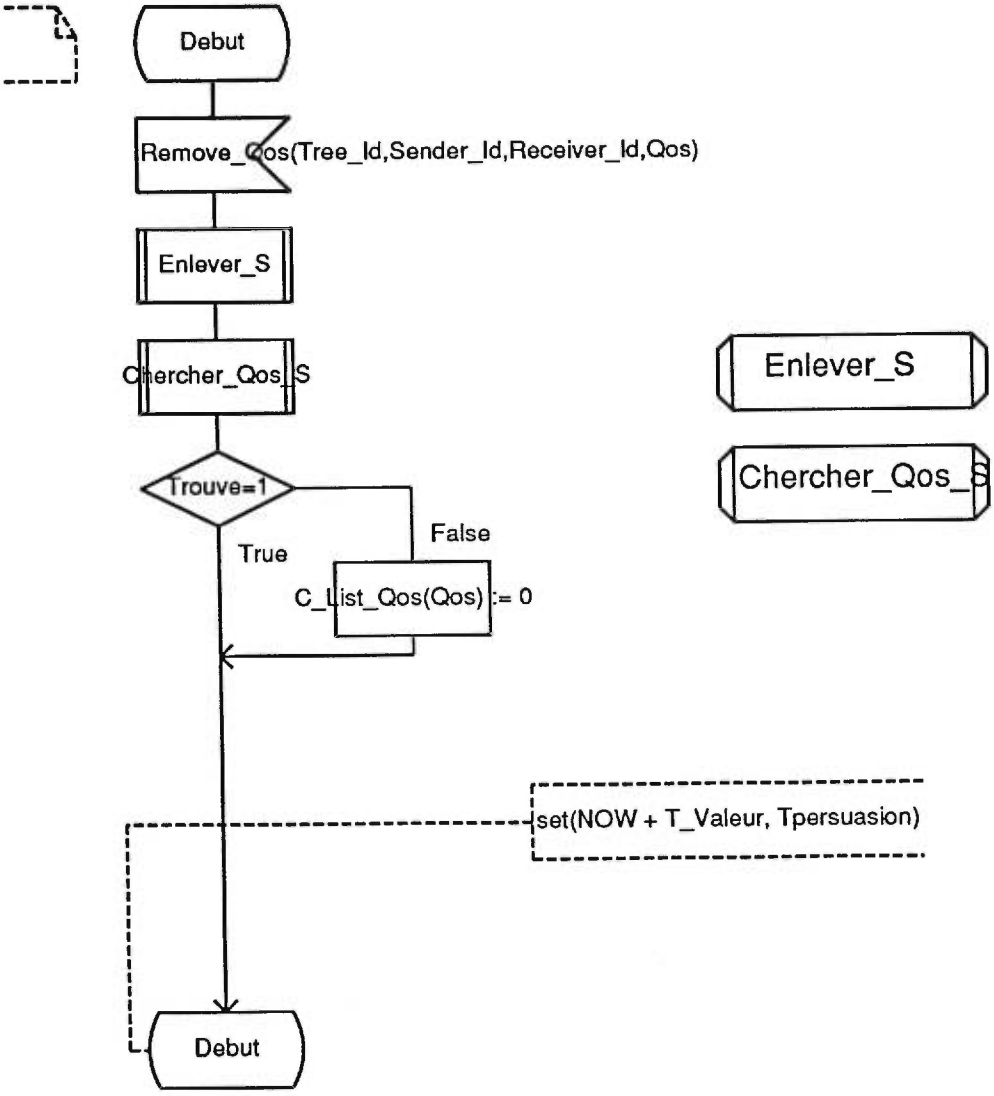
Spécification en SDL du protocole de gestion coopérative de la qualité de service dans les applications multimédias

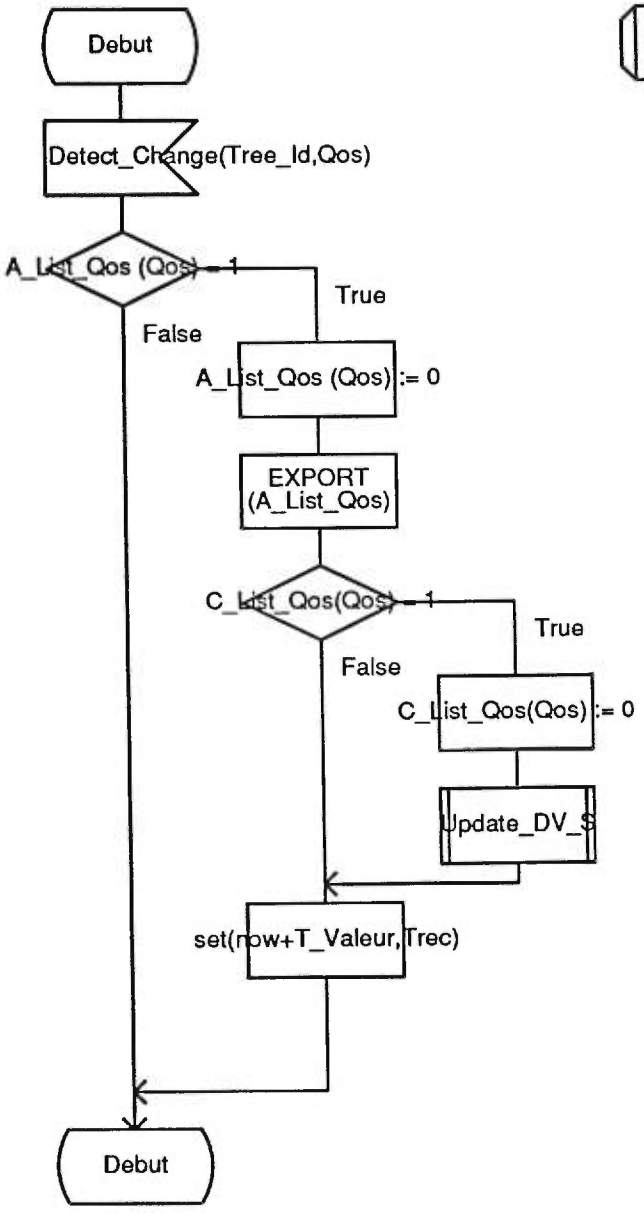

Block Block_Global
PageB1(1)
Source(1,1)
Contexte(1,1)
Tree_Manager(0,10)
Prot_Routing
Gestion(1,1)
User(0,1010)
Sink(0,1010)
Annuler
Annuler_Qos
Detect_Change, NbrS
Add_Qos_Info, Remove_Qos, Viol
C_S
S_TM
Persuade, Arret_Persuade, Not_Ok, Solve
Adr_Source, Adr_TM
Adr_TM, Adr_Sink
C_TM
Detect_Change, Nbr
Routing
R_TM
Ask_Routing
Add_Qos_Info, Remove_Qos, Viol, Info_Connexion_S
C_Si
TM_G
TM_Si
R_Si
NUQ,NUT,PRAP, PSP,PEAP
Adr_User
U_G
Ask_Connexion
Detect_Change
Commence, Persuade, Arret_Persuade, Not_Ok, Solve, Info_Connexion_TM
U_Si
Init, Negociation, Deconnexion
Response, Connexion, Renegociation
SIGNAL
Detect_Change(integer,integer),
Ask_Routing,
Routing(integer),
Negociation(integer,Real_Streams),
Connexion(integer),
Deconnexion(Integer),
Response(Integer,integer),
NbrS(Integer,integer),
Nbr(Integer,integer,Real),
Renegociation(integer),
Init,
Adr_Source(Pid),
Adr_Sink(Pid,Integer),
Adr_TM(Pid,Integer),
Ask_Connexion,
Adr_User(Pid),
NUQ(integer, Real),
NUT (Real),
PRAP(Real),
PSP(Real),
PEAP(integer, Real);
SIGNAL
Add_Qos_Info(integer,Pid,Pid,integer),
Remove_Qos(integer,Pid,Pid,integer),
Viol(integer,Pid,Pid,Streams),
Persuade(Integer,Pid,Pid,Integer,Real),
Arret_Persuade(integer,Pid,Pid),
Solve(integer,Pid,Pid,Streams),
Not_Ok(integer,Pid,Pid,Integer),
Commence(Real),
Info_Connexion_S(Integer,Pid,Pid),
Info_Connexion_TM(integer,Pid,Pid,Integer,Real);
syntype ind4 = integer
constants 1:1002
endsyntype;
newtype Tab
ARRAY(ind4,Integer)
endnewtype Tab;
newtype Tab_Real
ARRAY(ind4,Real)
endnewtype Tab_Real;
newtype TabPID
ARRAY(ind4,Pid)
endnewtype TabPid;
syntype ind3 = integer
constants 1:3
endsyntype;
newtype Streams
Array(ind3,Integer)
endnewtype Streams;
newtype Real_Streams
Array(ind3,Real)
endnewtype Real_Streams;
newtype Agent_Qos STRUCT
Agent Pid;
ListQos Streams;
endnewtype Agent_Qos;
syntype ind6 = integer
constants 1:1002
endsyntype;
newtype List_Agent_Qos
ARRAY(ind6, Agent_Qos)
endnewtype List_Agent_Qos;
newtype Probabilites
ARRAY(ind3, Real_Streams)
endnewtype Probabilites;
syntype ind1000 = integer
constants 1:1000
endsyntype;
newtype Appreciation
ARRAY(ind1000, Real_Streams)
endnewtype Appreciation;
procedure random;
fpar IN/OUT x real;
external;
/*#CODE
#BODY
double random(double *x)
{ *x = (double)rand()/32767;}
*/

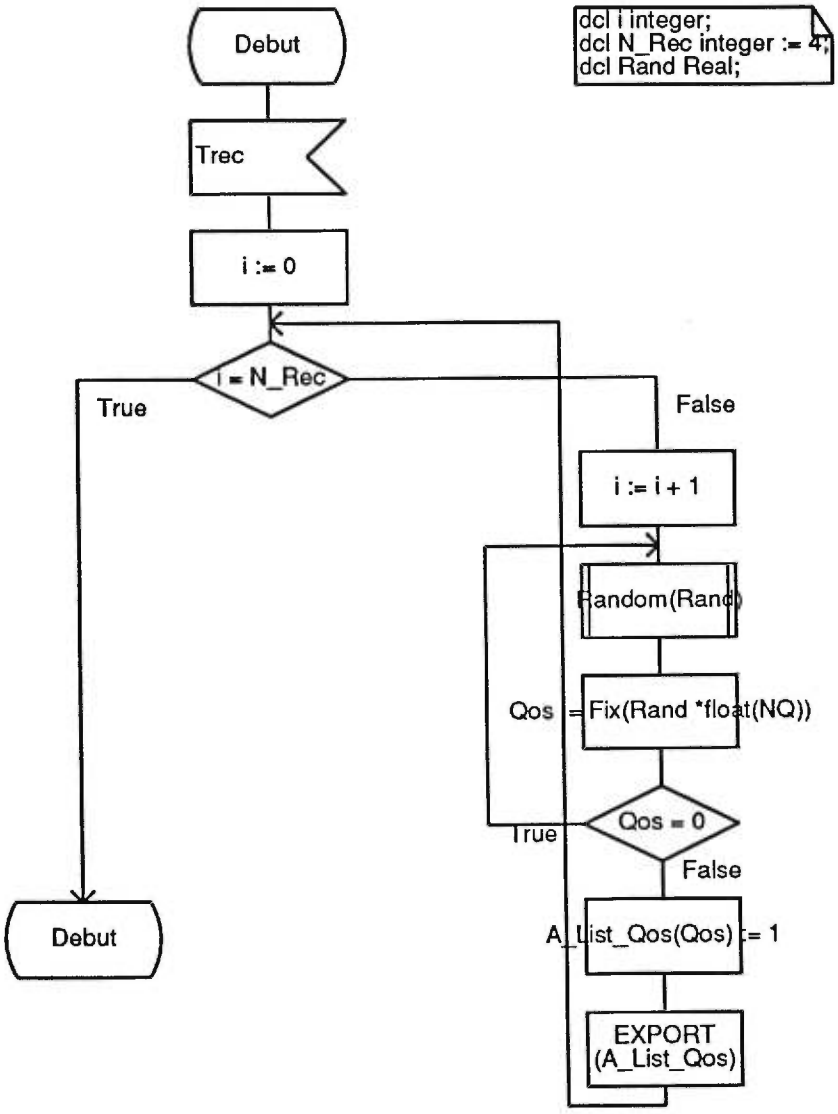

ss Source
SourceCreat(7)
dcl Moi, Sender_Id, Receiver_Id Pid;
dcl List_Qos, C_List_Qos Streams;
dcl Tree_Id, Qos Integer;
dcl V_List_Agent, Downstream_Agents List_Agent_Qos;
timer T, Trec, Tpersuasion;
dcl T_Valeur Duration := 1000;
dcl Trouve integer;
dcl N, NR,K Integer;
dcl iAgt Integer := 0;
dcl NR Real:= 0;
dcl EXPORTED A_List_Qos Streams := (. 1 .);
dcl NQ integer := 3;
Init
NbrS(NR,k)
N := 0
Adr_Source(self)
N=NR
True
False
N:=N+1
Tree_Manager
Downstream_Agents(N)!Agent := offspring
i := self
Adr_TM(offspring,k)
Debut
K := K+1


Debut
Viol(Tree_Id,Sender_Id,Receiver_Id,List_Qos)
iAgt:=1,
V_List_Agent(iAgt)!Agent:=Sender_Id,
V_List_Agent(iAgt)!ListQos:= List_Qos
set(now + T_Valeur,T)
Attente
T
iAgt = NR
True
False
Cummuler_Qos_S
Annuler_List_Qos,A_List_Qos)
EXPORT
(A_List_Qos)
Annuler_List_Qos,C_List_Qos)
Update_Viol_S
set(now + T_Valeur,Trec)
Debut
Solve_Agent_S
Debut
Viol(Tree_Id,Sender_Id,Receiver_Id,List_Qos)
Chercher_S
Solve_Agent_S
Chercher_S
Cummuler_Qos_S
Update_Viol_S


Debut
Add_Qos_Info(Tree_Id,Sender_Id,Receiver_Id,Qos)
Persuade_Agent_S
Update_Add_S
A_List_Qos(Qos) = 1
False
True
Not_Ok (Tree_Id, Moi, Sender_Id, Qos)
C_List_Qos(Qos) := 1
Update_Add_S
set(NOW + T_Valeur, Tpersuasion)
Debut

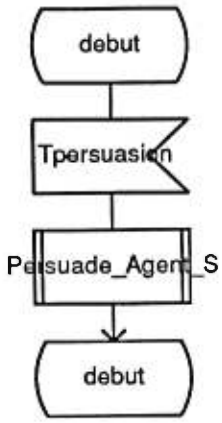

Debut
Remove_Qos(Tree_Id,Sender_Id,Receiver_Id,Qos)
Enlever_S
Chercher_Qos_S
Trouve=1
True
False
C_List_Qos(Qos) := 0
set(NOW + T_Valeur, Tpersuasion)
Debut
Enlever_S
Chercher_Qos_S

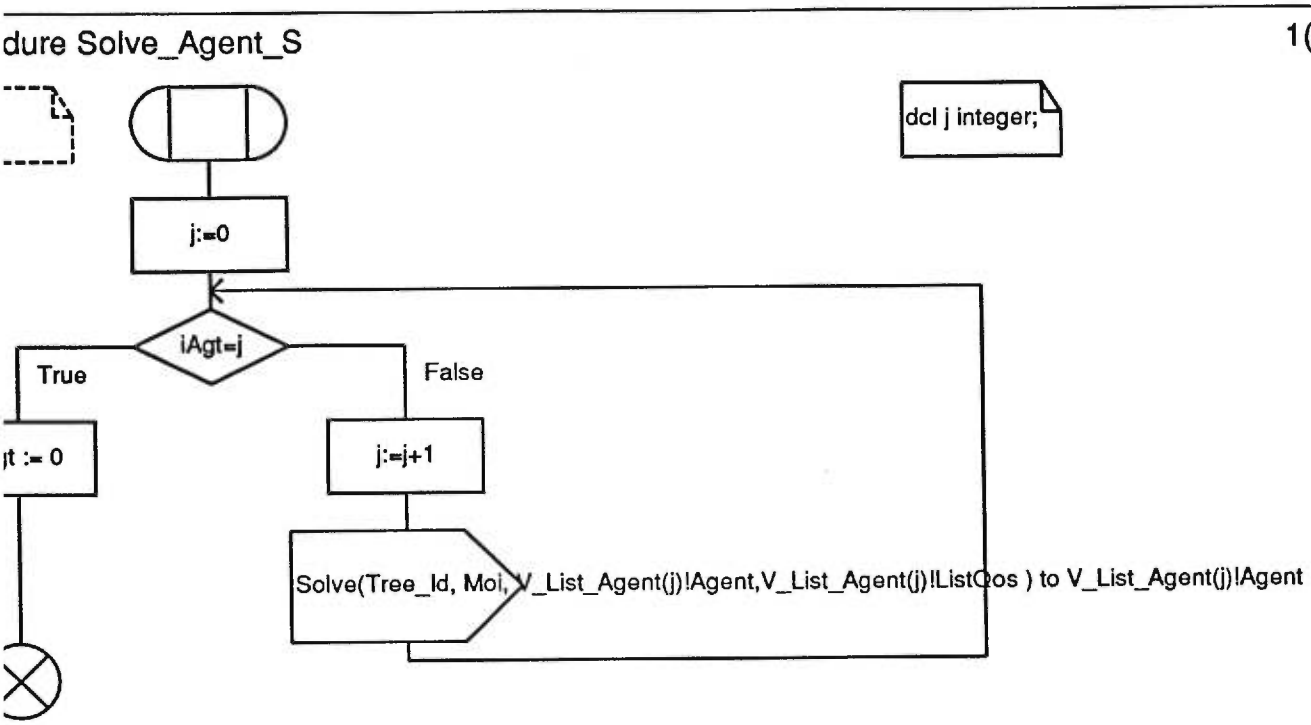

Debut
Update_DV_S
Detect_Change(Tree_Id,Qos)
A_List_Qos (Qos) = 1
True
False
A_List_Qos (Qos) := 0
EXPORT
(A_List_Qos)
C_List_Qos(Qos) = 1
True
False
C_List_Qos(Qos) := 0
Update_DV_S
set(now+T_Valeur,Trec)
Debut

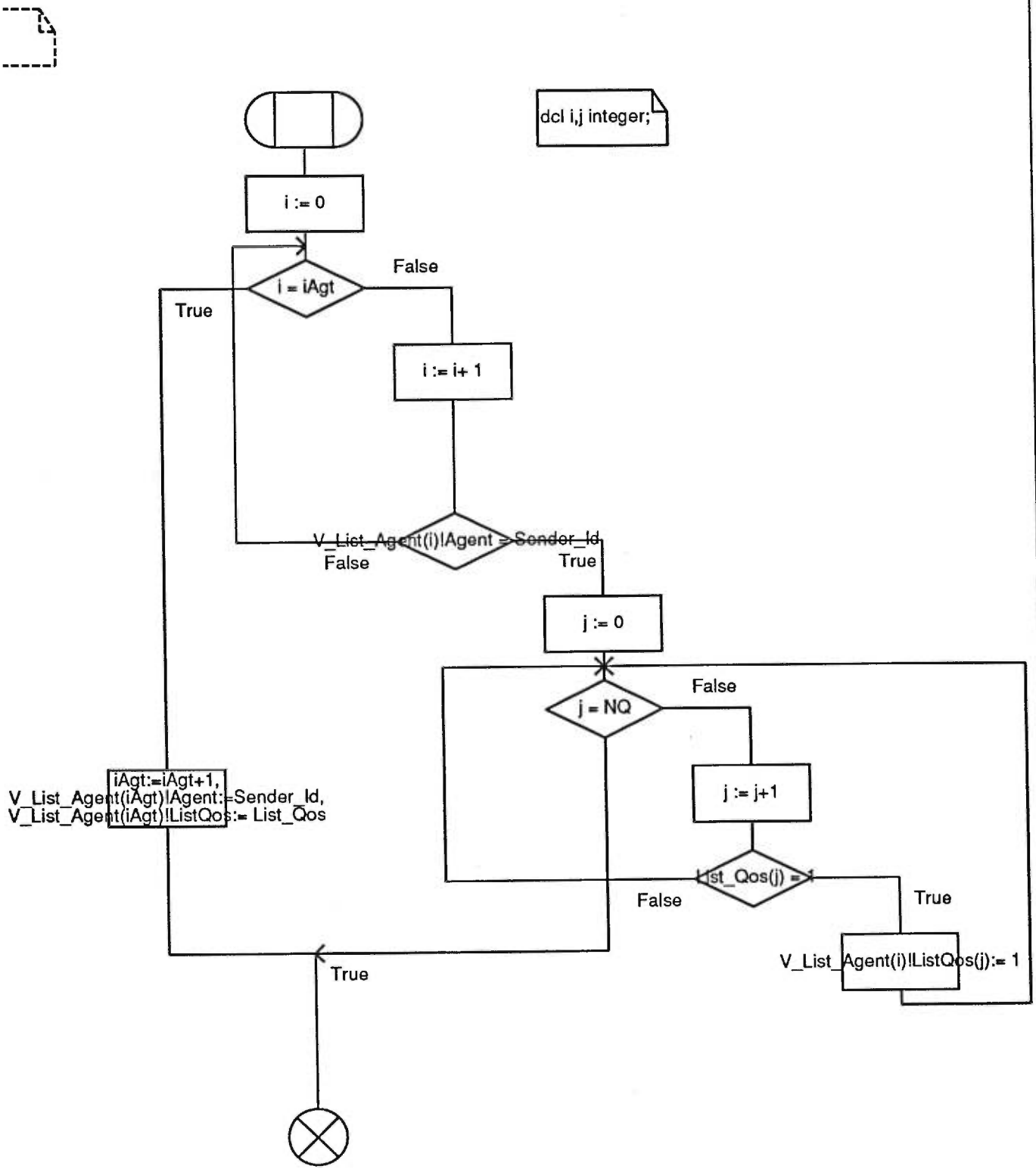

Debut
dcl i integer;
dcl N_Rec integer := 4;
dcl Rand Real;
Trec
i := 0
i = N_Rec
True
False
i := i + 1
Random(Rand)
Qos = Fix(Rand *float(NQ))
Qos = 0
True
False
Debut
A_List_Qos(Qos) := 1
EXPORT
(A_List_Qos)

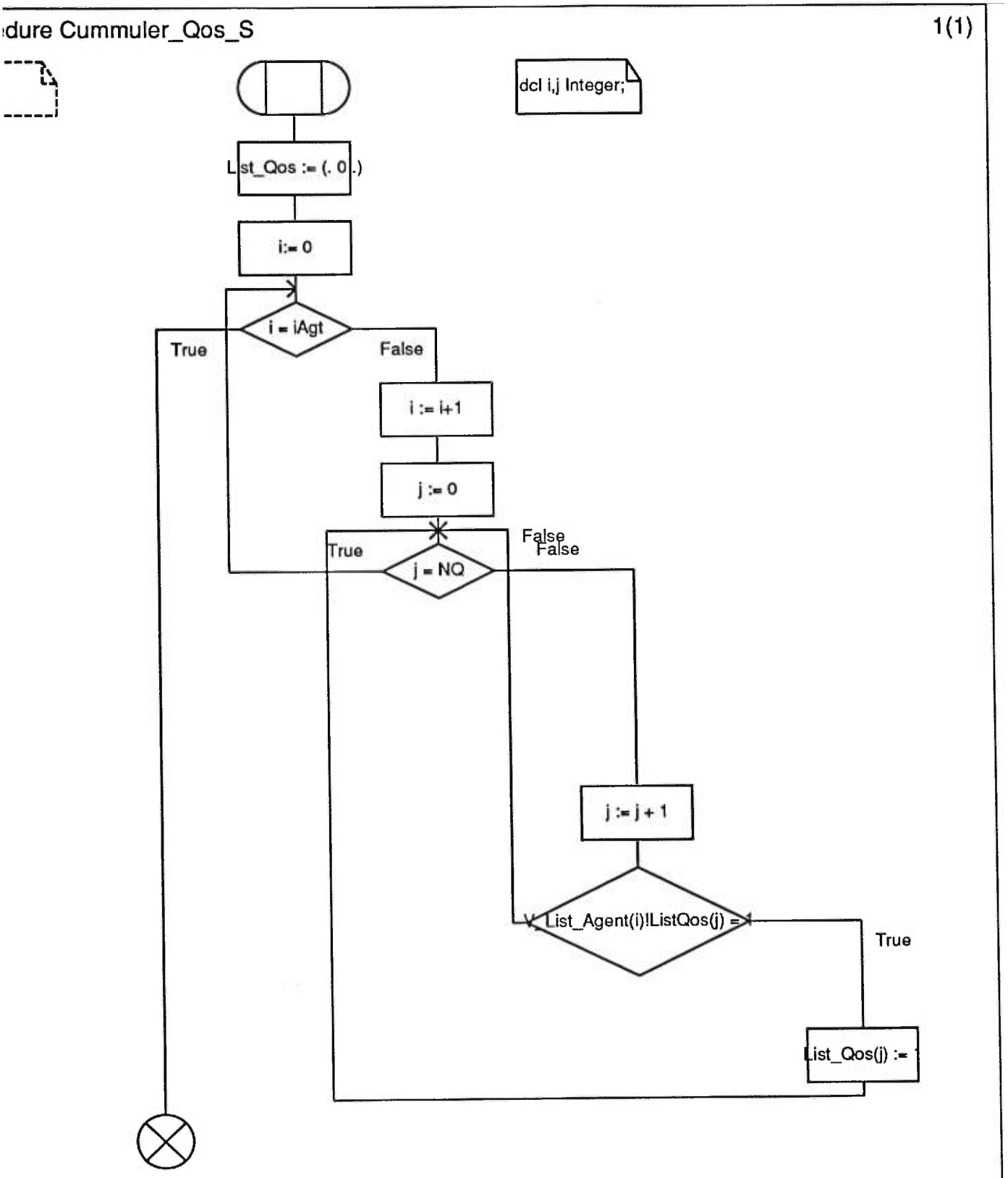
ss Source

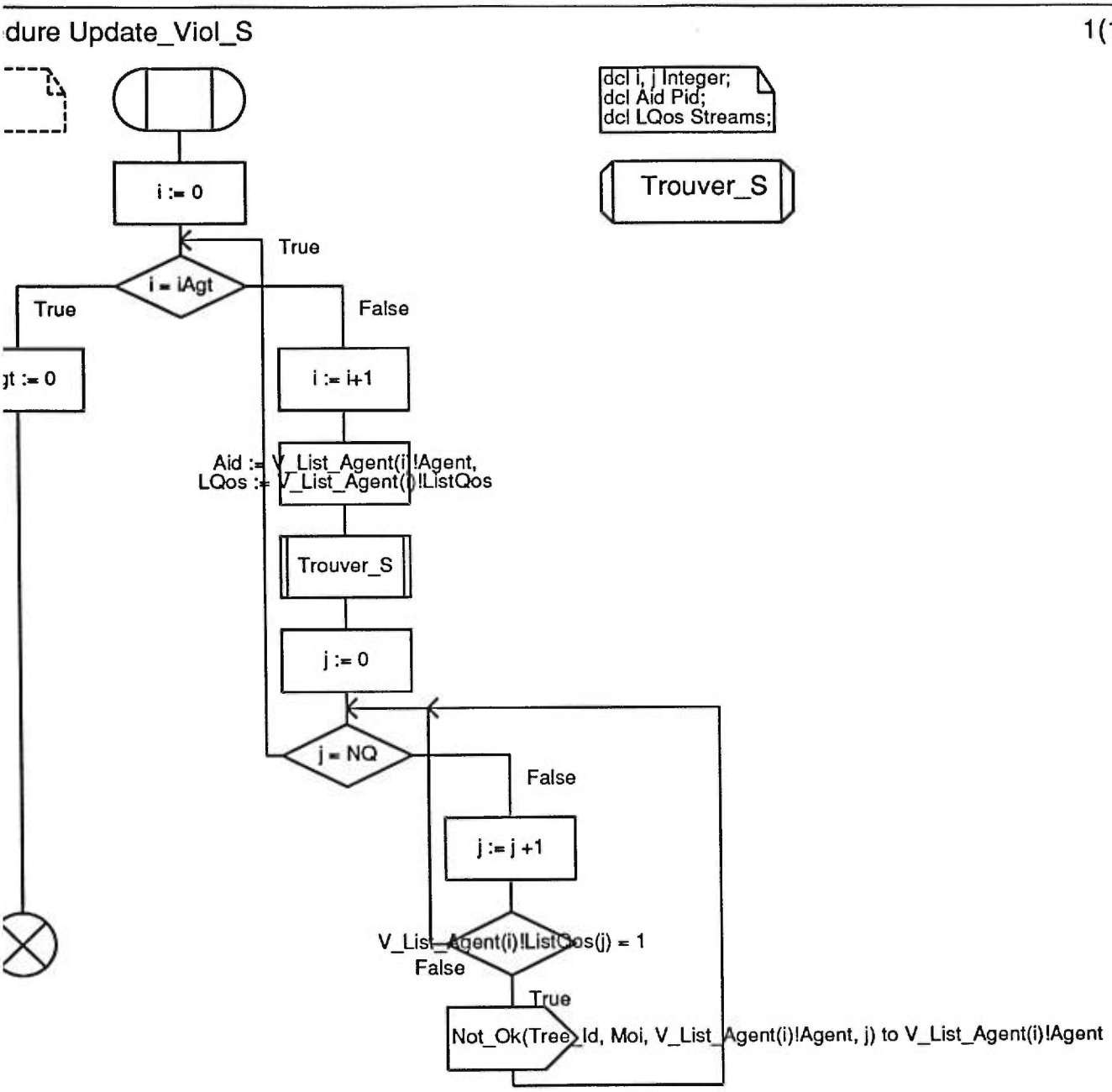
Source_Pers(7)

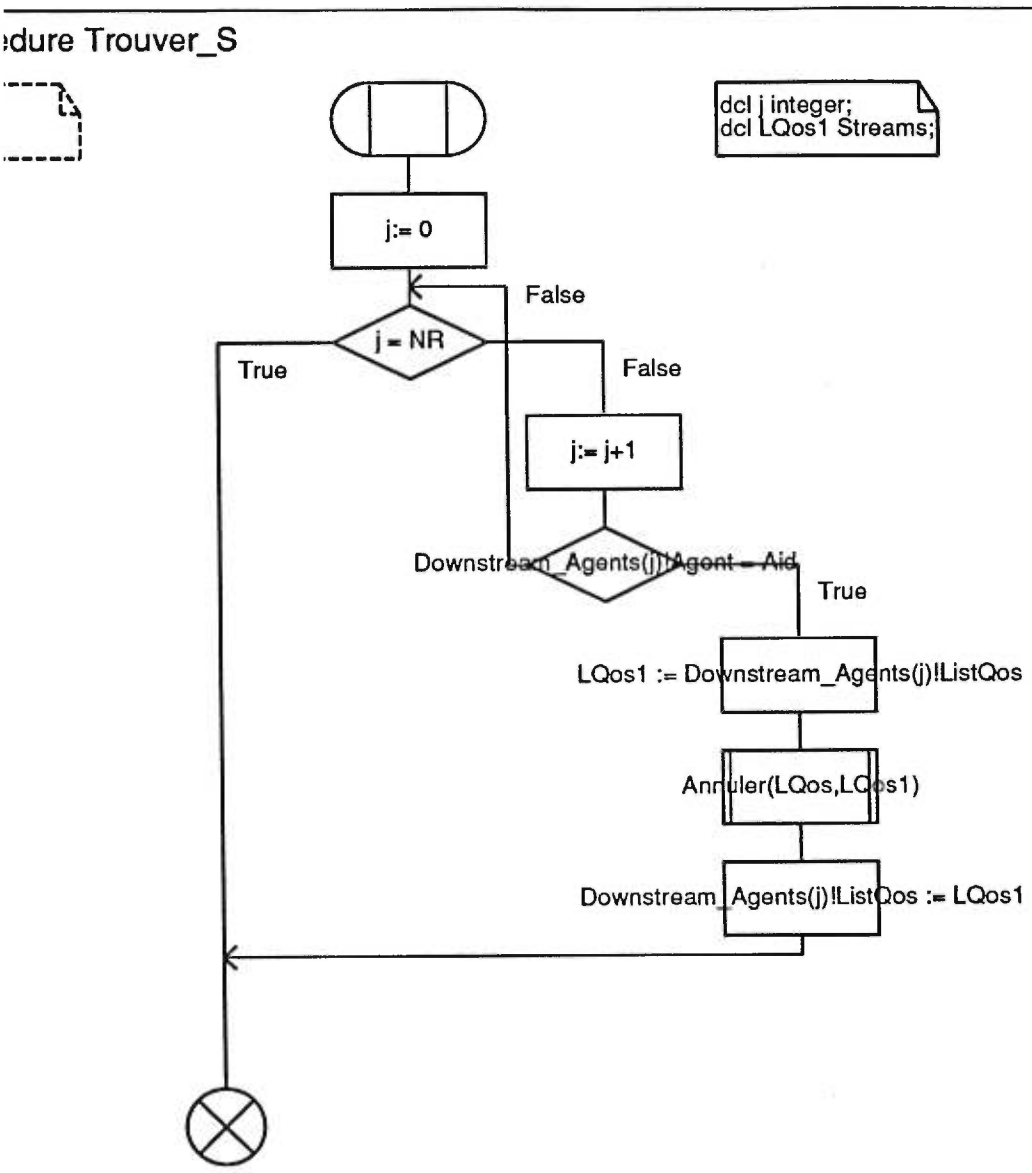



dure Solve_Agent_S
1(1)
dcl j integer;
j:=0
iAgt=j
True
False
jt := 0
j:=j+1
Solve(Tree_Id, Moi, V_List_Agent(j)!Agent,V_List_Agent(j)!ListQos) to V_List_Agent(j)!Agent

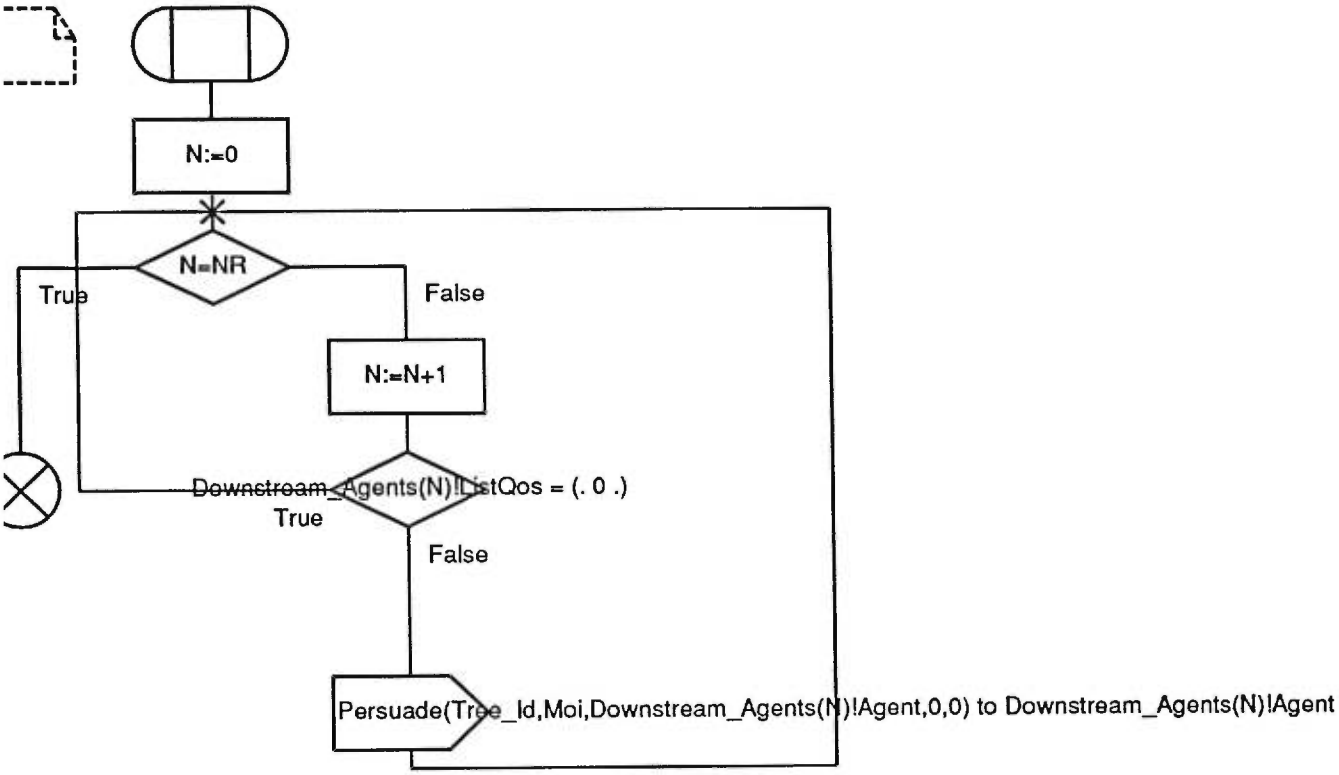

dcl i,j integer;
i := 0
i = iAgt
False
True
i := i+ 1
V_List_Agent(i)!Agent = Sender_Id
False
True
j := 0
j = NQ
False
j := j+1
List_Qos(j) = 1
False
True
V_List_Agent(i)!ListQos(j):= 1
iAgt:=iAgt+1,
V_List_Agent(iAgt)!Agent:=Sender_Id,
V_List_Agent(iAgt)!ListQos:= List_Qos
True


dure Cummuler_Qos_S
1(1)
dcl i,j Integer;
List_Qos := (. 0 .)
i:= 0
i = iAgt
True
False
i := i+1
j := 0
True
False
False
j = NQ
j := j + 1
List_Agent(i)!ListQos(j) =
True
List_Qos(j) :=


dure Update_Viol_S
1(1)
dcl i, j Integer;
dcl Aid Pid;
dcl LQos Streams;
Trouver_S
i := 0
True
i = iAgt
True
False
gt := 0
i := i+1
Aid := V_List_Agent(i)!Agent,
LQos := V_List_Agent()!ListQos
Trouver_S
j := 0
j = NQ
False
j := j +1
V_List_Agent(i)!ListQos(j) = 1
False
True
Not_Ok(Tree_Id, Moi, V_List_Agent(i)!Agent, j) to V_List_Agent(i)!Agent

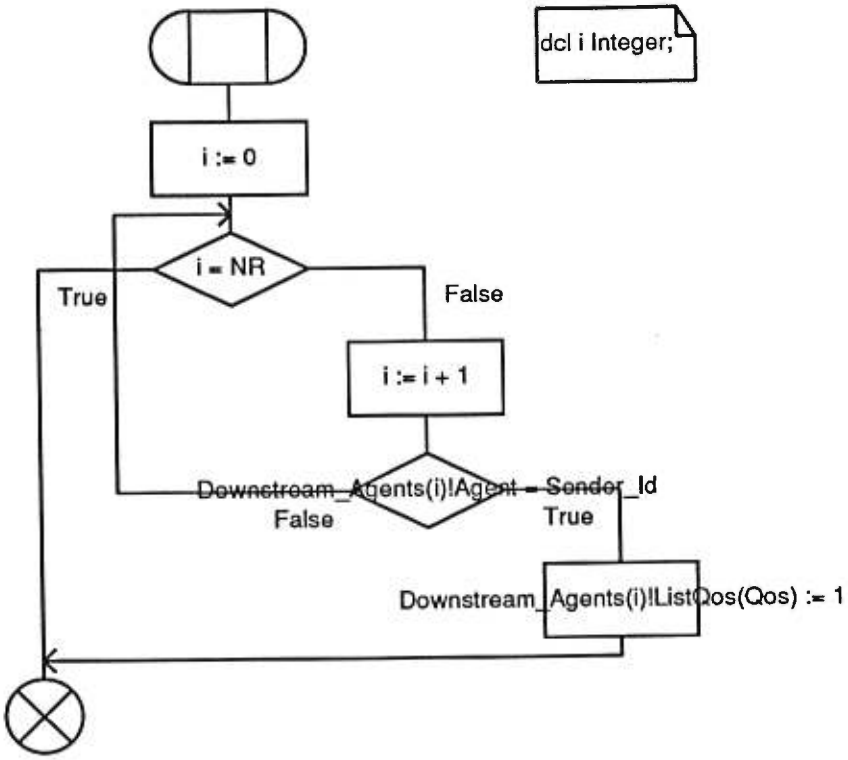

dure Trouver_S
1(1)
dcl j integer;
dcl LQos1 Streams;
j:= 0
False
j = NR
True
False
j:= j+1
Downstream_Agents(j)!Agent = Aid
True
LQos1 := Downstream_Agents(j)!ListQos
Annuler(LQos,LQos1)
Downstream_Agents(j)!ListQos := LQos1

dure Persuade_Agent_S 1(1)



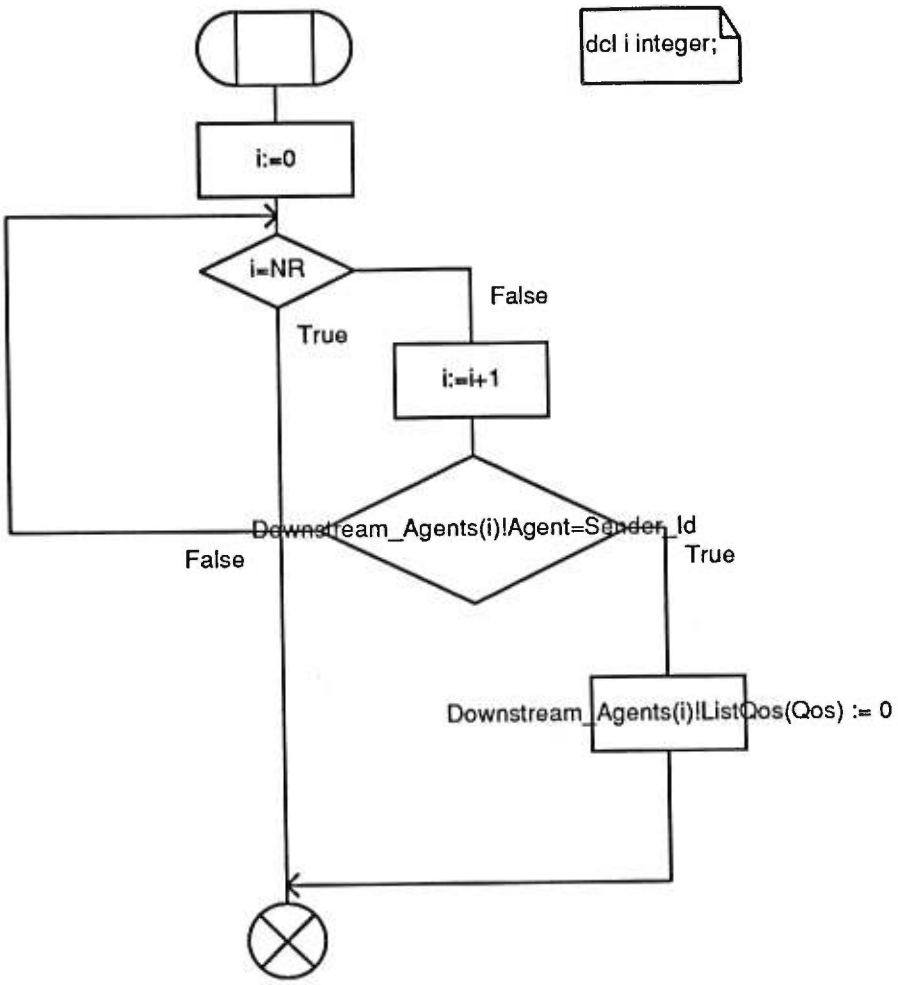
dure Update_Add_S

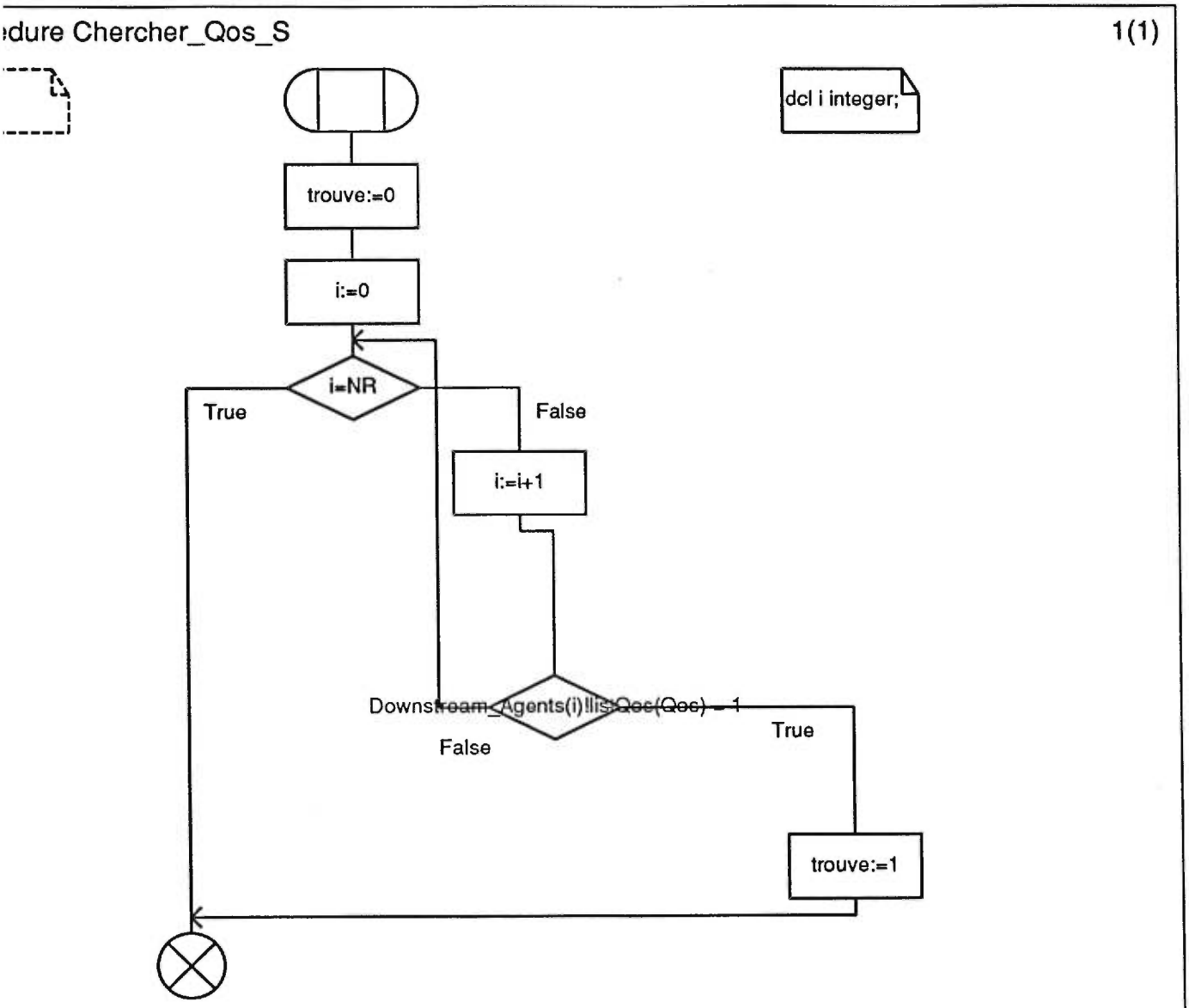
1(1)

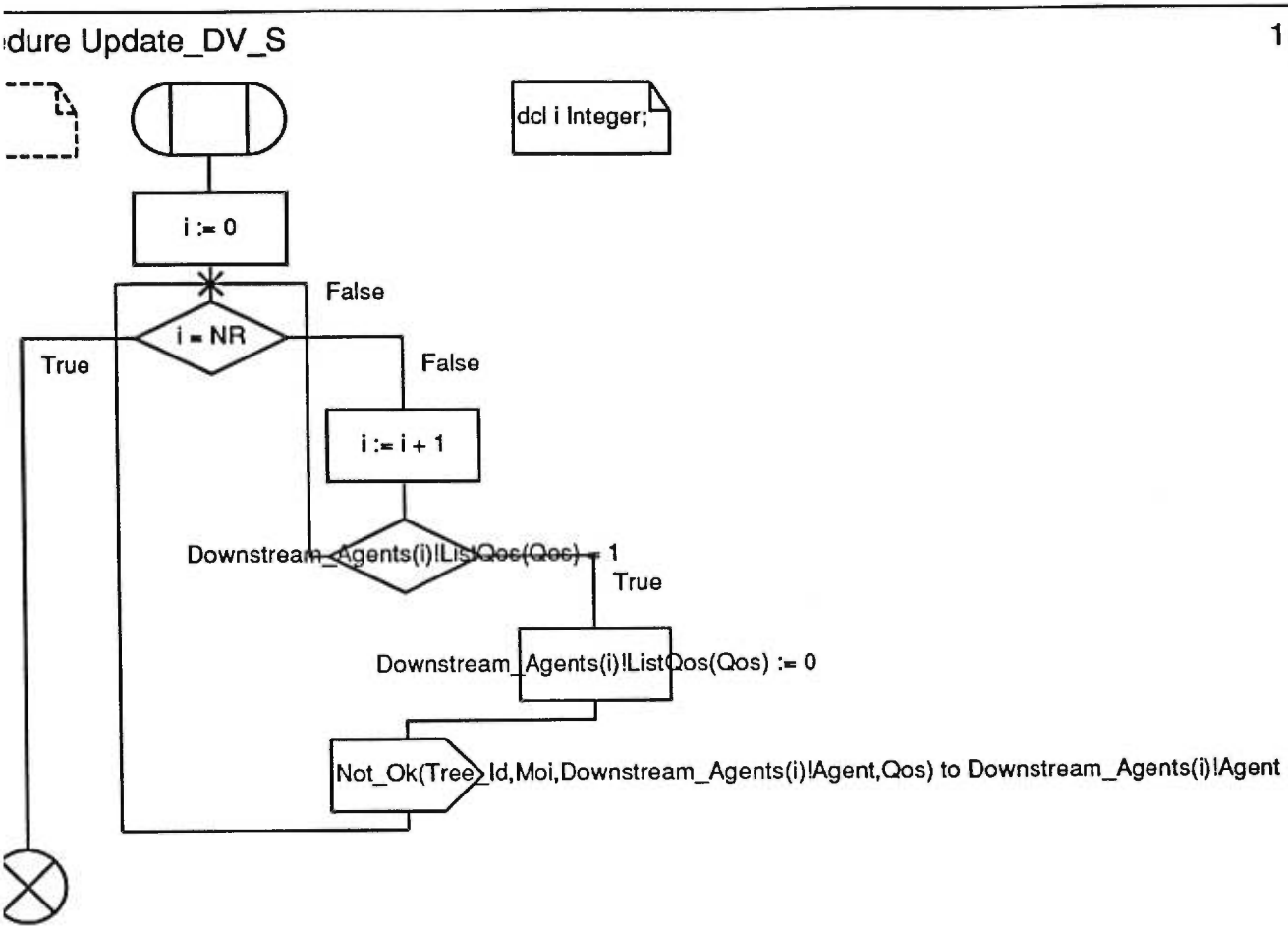


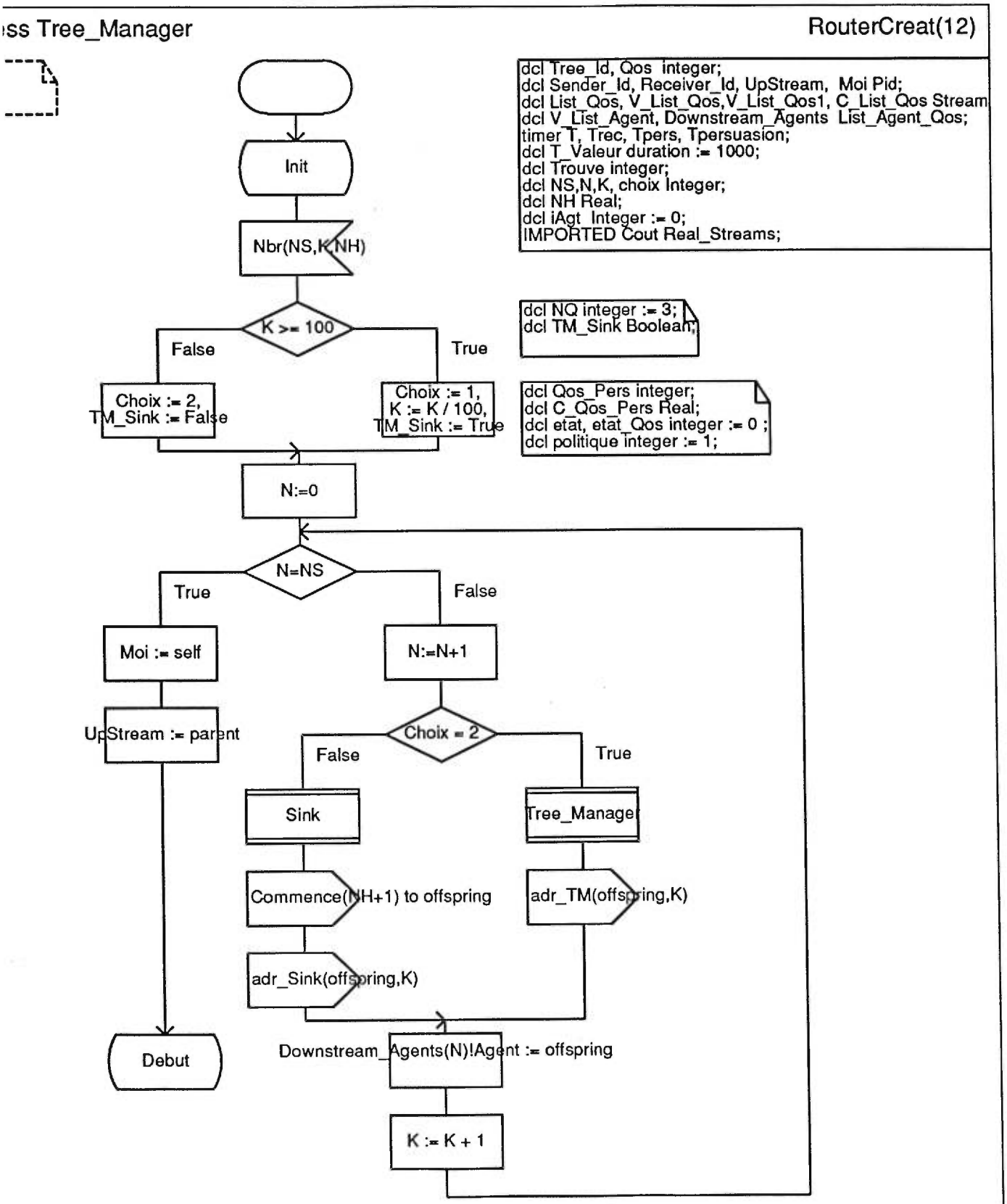
dure Enlever_S

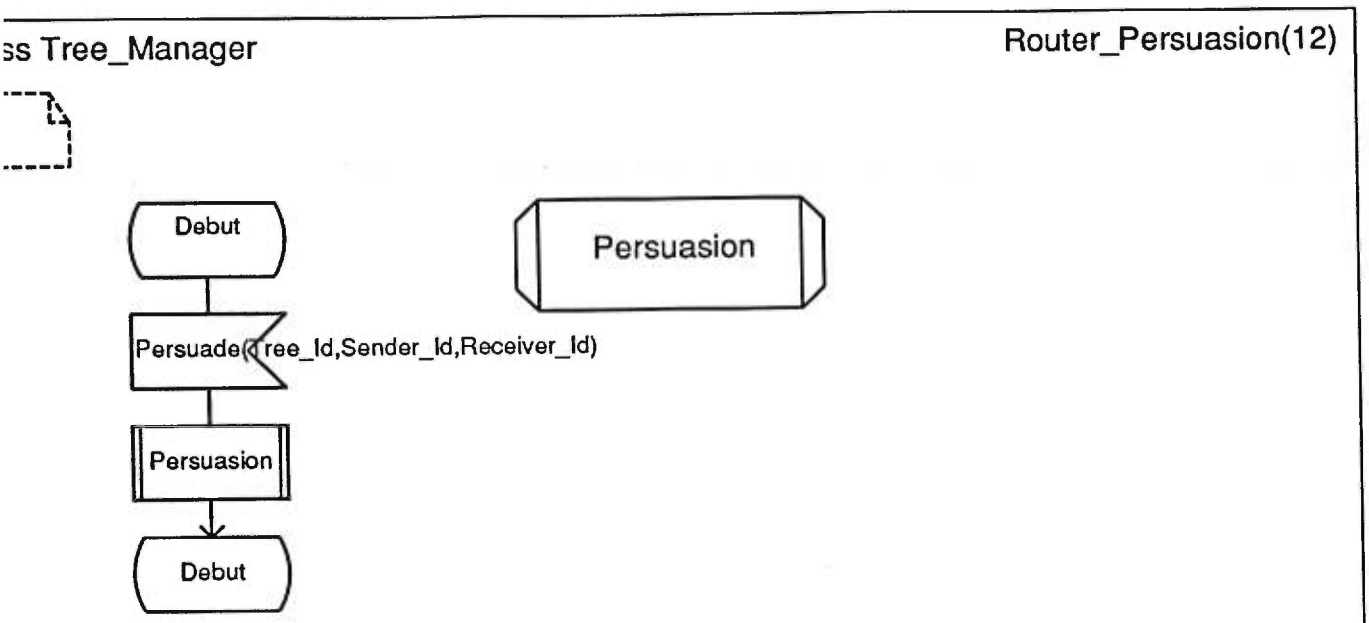
1(1)




dure Chercher_Qos_S
1(1)
dcl i integer;
trouve:=0
i:=0
i=NR
True
False
i:=i+1
Downstream≤Agents(i)!isQos(Qos) = 1
True
False
trouve:=1

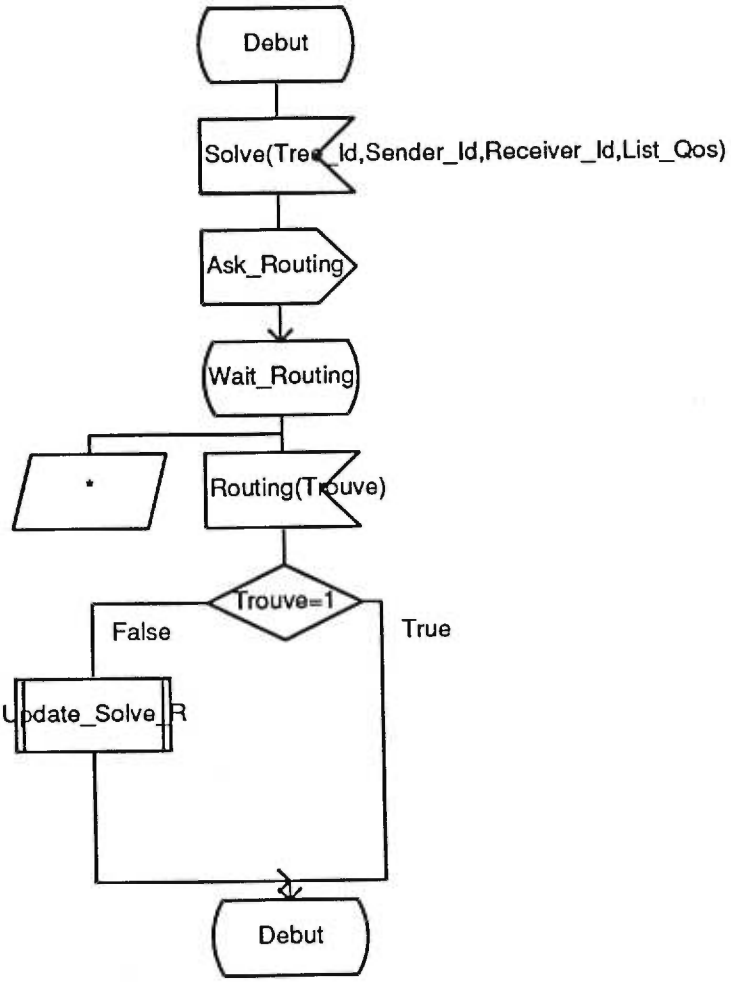

dure Update_DV_S
1(1)
dcl i Integer;
i := 0
False
i = NR
True
False
i := i + 1
Downstream_Agents(i)!ListQos(Qos) = 1
True
Downstream_Agents(i)!ListQos(Qos) := 0
Not_Ok(Tree_Id,Moi,Downstream_Agents(i)!Agent,Qos) to Downstream_Agents(i)!Agent

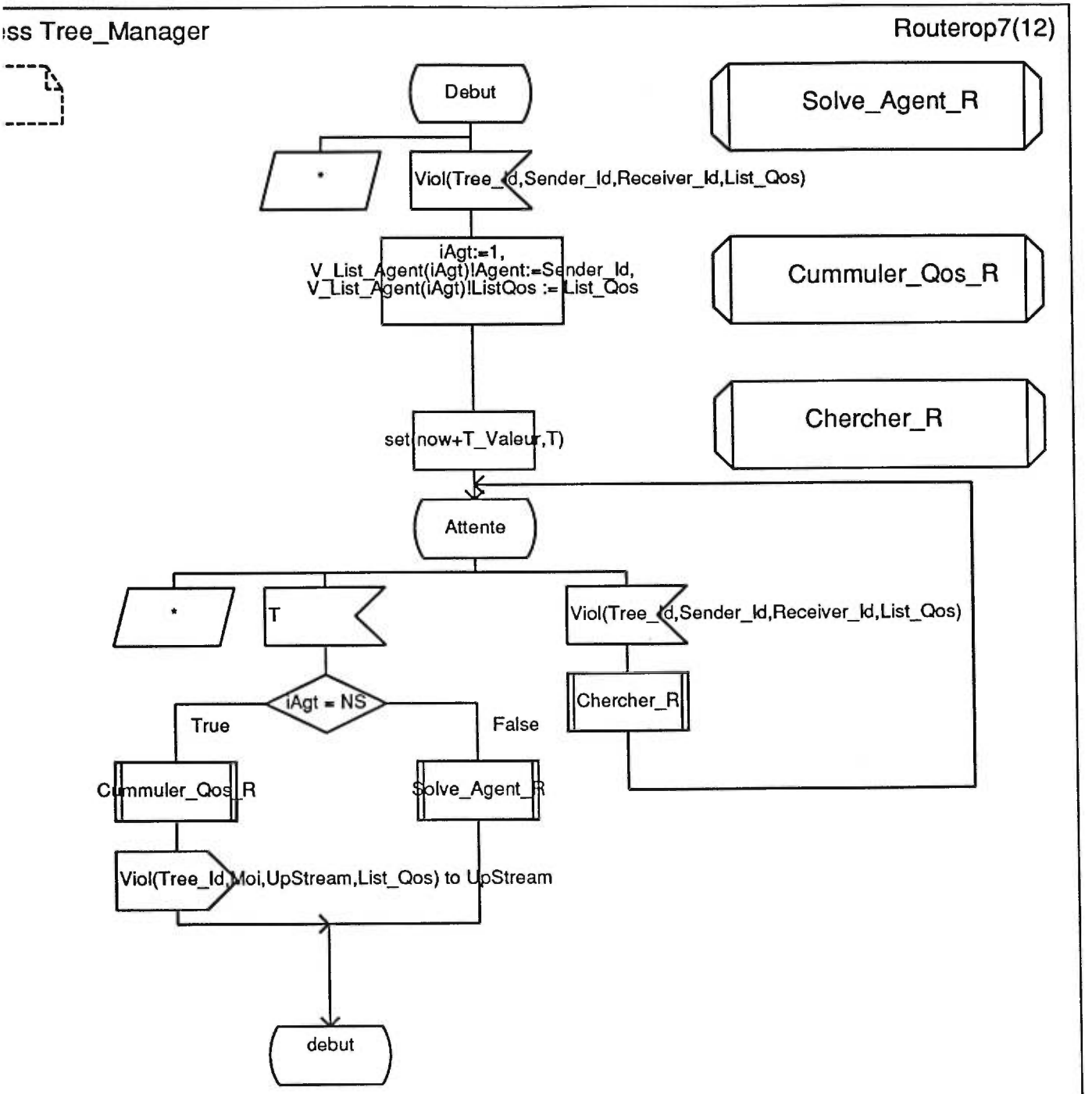

ss Tree_Manager
RouterCreat(12)
dcl Tree_Id, Qos integer;
dcl Sender_Id, Receiver_Id, UpStream, Moi Pid;
dcl List_Qos, V_List_Qos,V_List_Qos1, C_List_Qos Stream
dcl V_List_Agent, Downstream_Agents List_Agent_Qos;
timer T, Trec, Tpers, Tpersuasion;
dcl T_Valeur duration := 1000;
dcl Trouve integer;
dcl NS,N,K, choix Integer;
dcl NH Real;
dcl iAgt Integer := 0;
IMPORTED Cout Real_Streams;
Init
Nbr(NS,K,NH)
K >= 100
False
True
dcl NQ integer := 3;
dcl TM_Sink Boolean;
Choix := 2,
TM_Sink := False
Choix := 1,
K := K / 100,
TM_Sink := True
dcl Qos_Pers integer;
dcl C_Qos_Pers Real;
dcl etat, etat_Qos integer := 0 ;
dcl politique integer := 1;
N:=0
N=NS
True
False
Moi := self
N:=N+1
UpStream := parent
Choix = 2
False
True
Sink
Tree_Manager
Commence(NH+1) to offspring
adr_TM(offspring,K)
adr_Sink(offspring,K)
Debut
Downstream_Agents(N)!Agent := offspring
K := K + 1

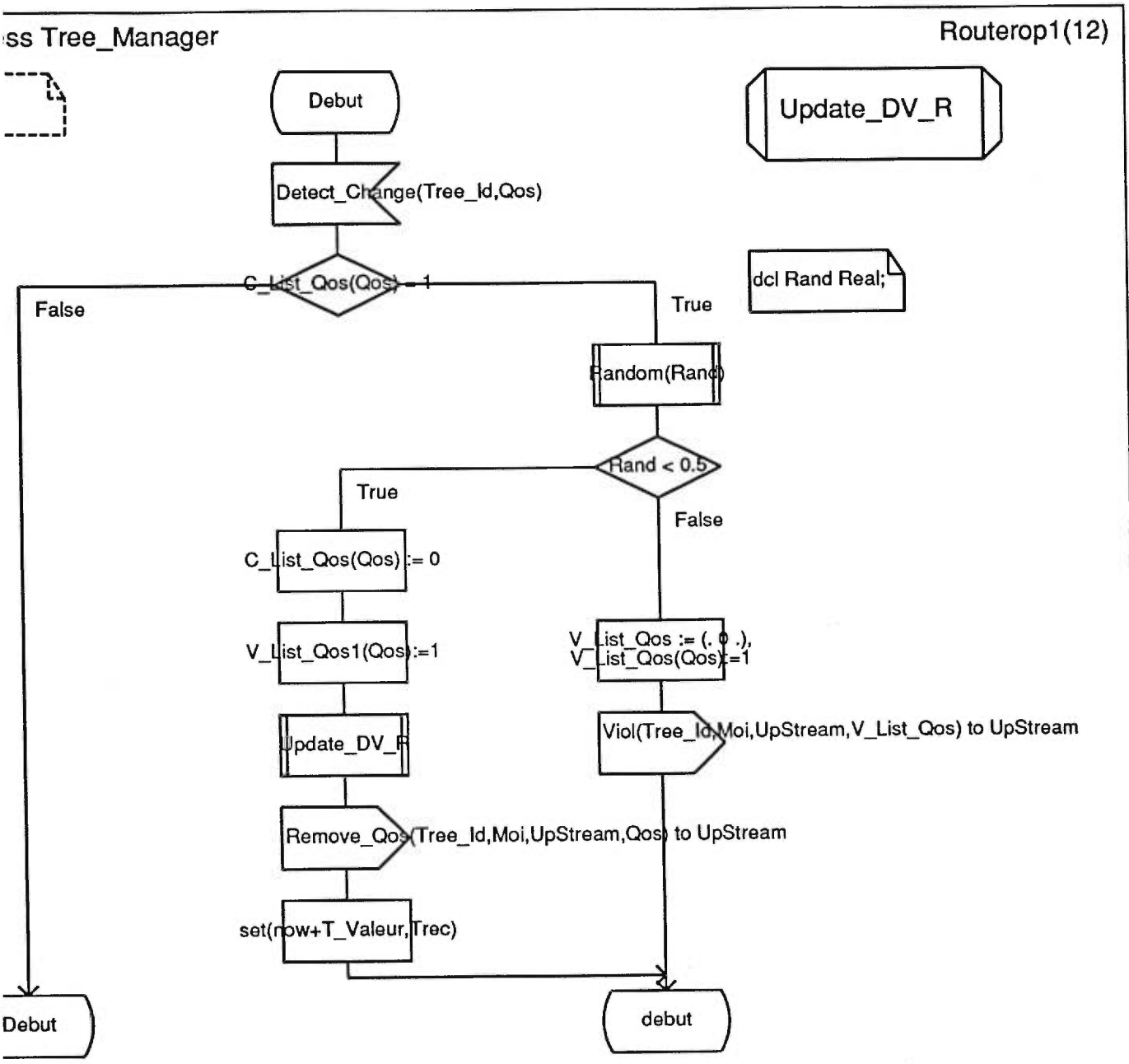

ss Tree_Manager
Router_Persuasion(12)
Debut
Persuasion
Persuade(Tree_Id,Sender_Id,Receiver_Id)
Persuasion
Debut

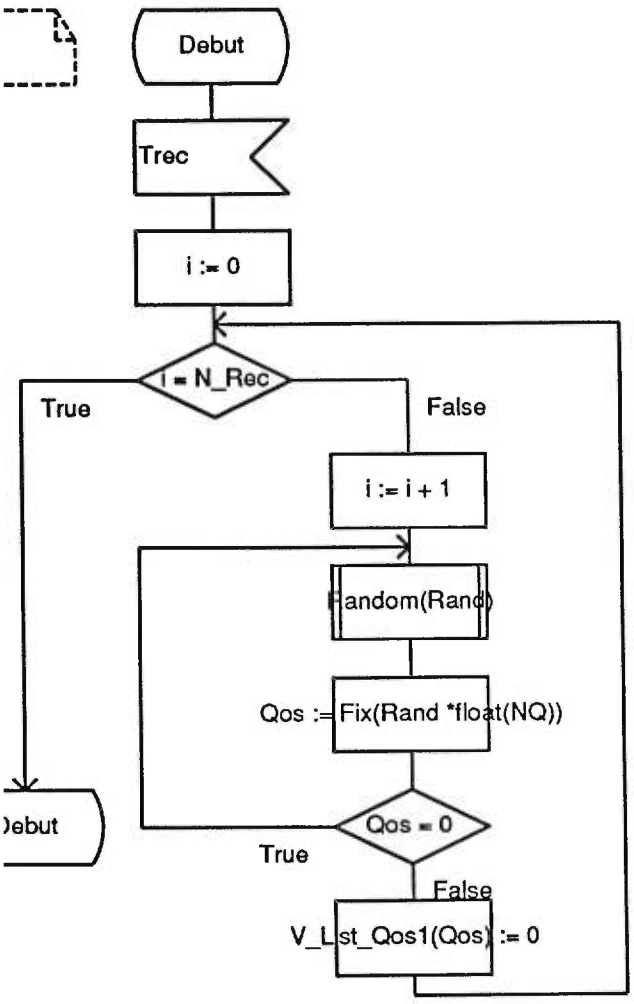
ss Tree_Manager

Routerop8(12)

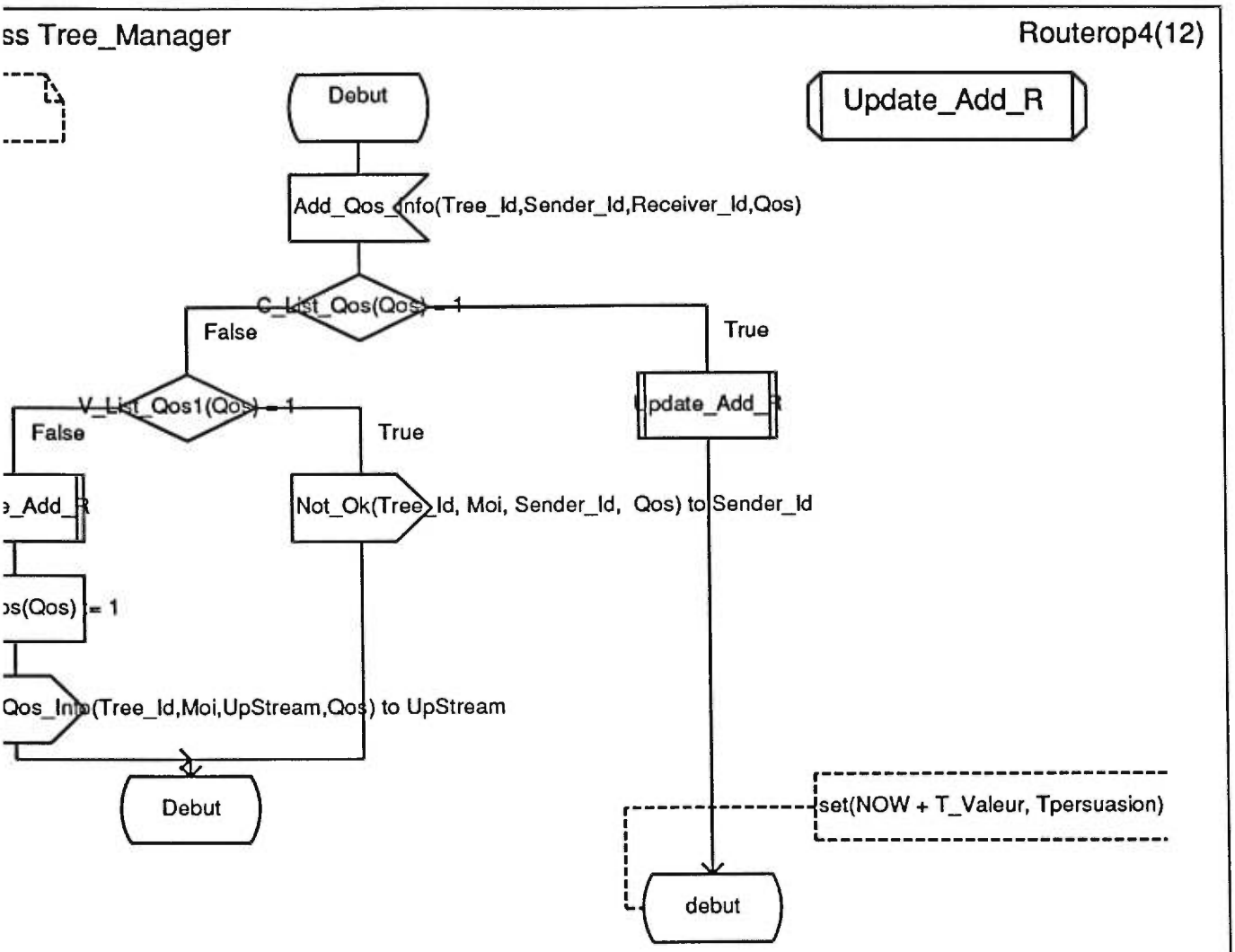


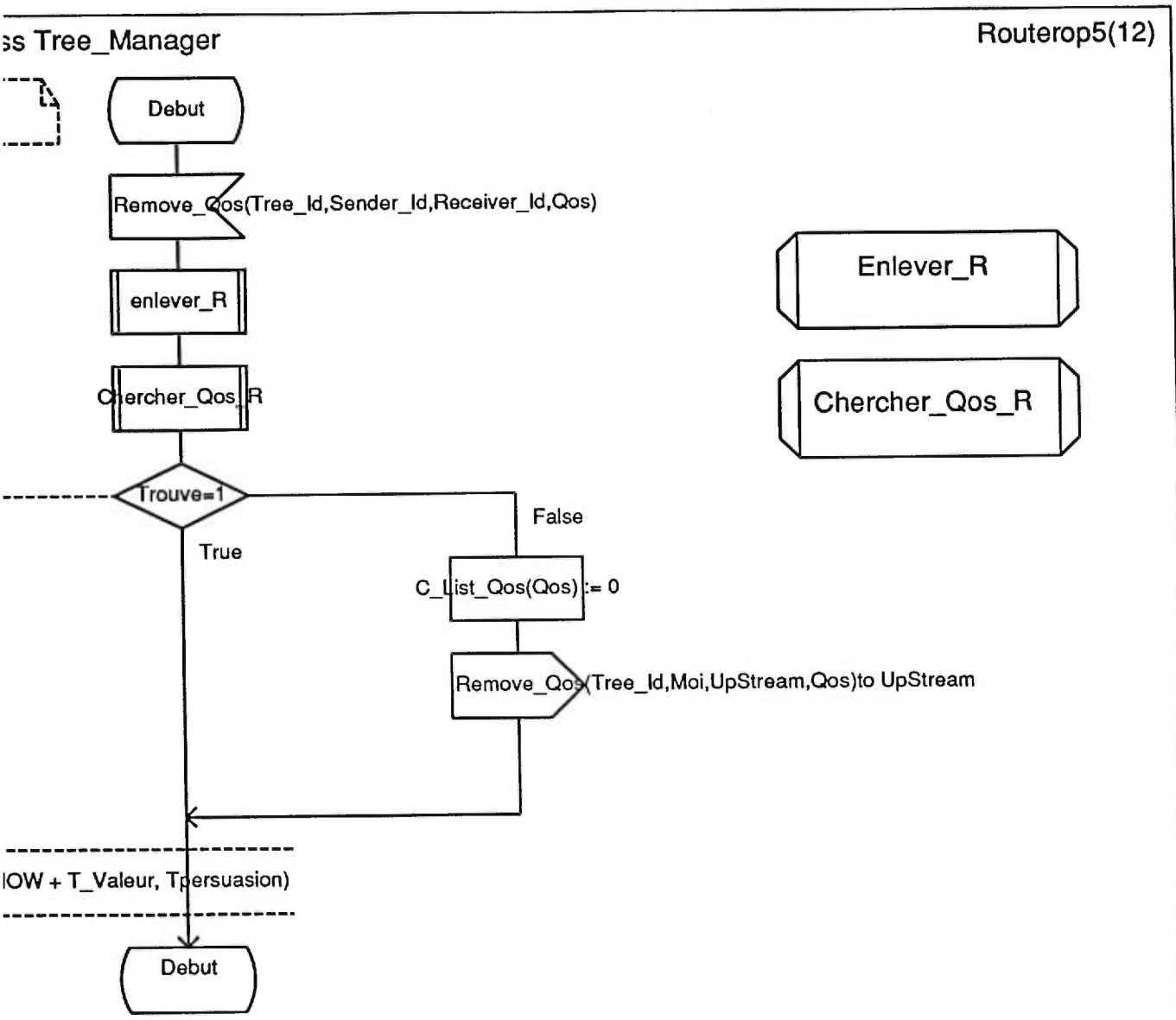

ess Tree_Manager
Routerop7(12)
Debut
Solve_Agent_R
*
Viol(Tree_Id,Sender_Id,Receiver_Id,List_Qos)
iAgt:=1,
V_List_Agent(iAgt)!Agent:=Sender_Id,
V_List_Agent(iAgt)!ListQos := List_Qos
Cummuler_Qos_R
Chercher_R
set(now+T_Valeur,T)
Attente
*
T
Viol(Tree_Id,Sender_Id,Receiver_Id,List_Qos)
Chercher_R
iAgt = NS
True
False
Cummuler_Qos_R
Solve_Agent_R
Viol(Tree_Id,Moi,UpStream,List_Qos) to UpStream
debut

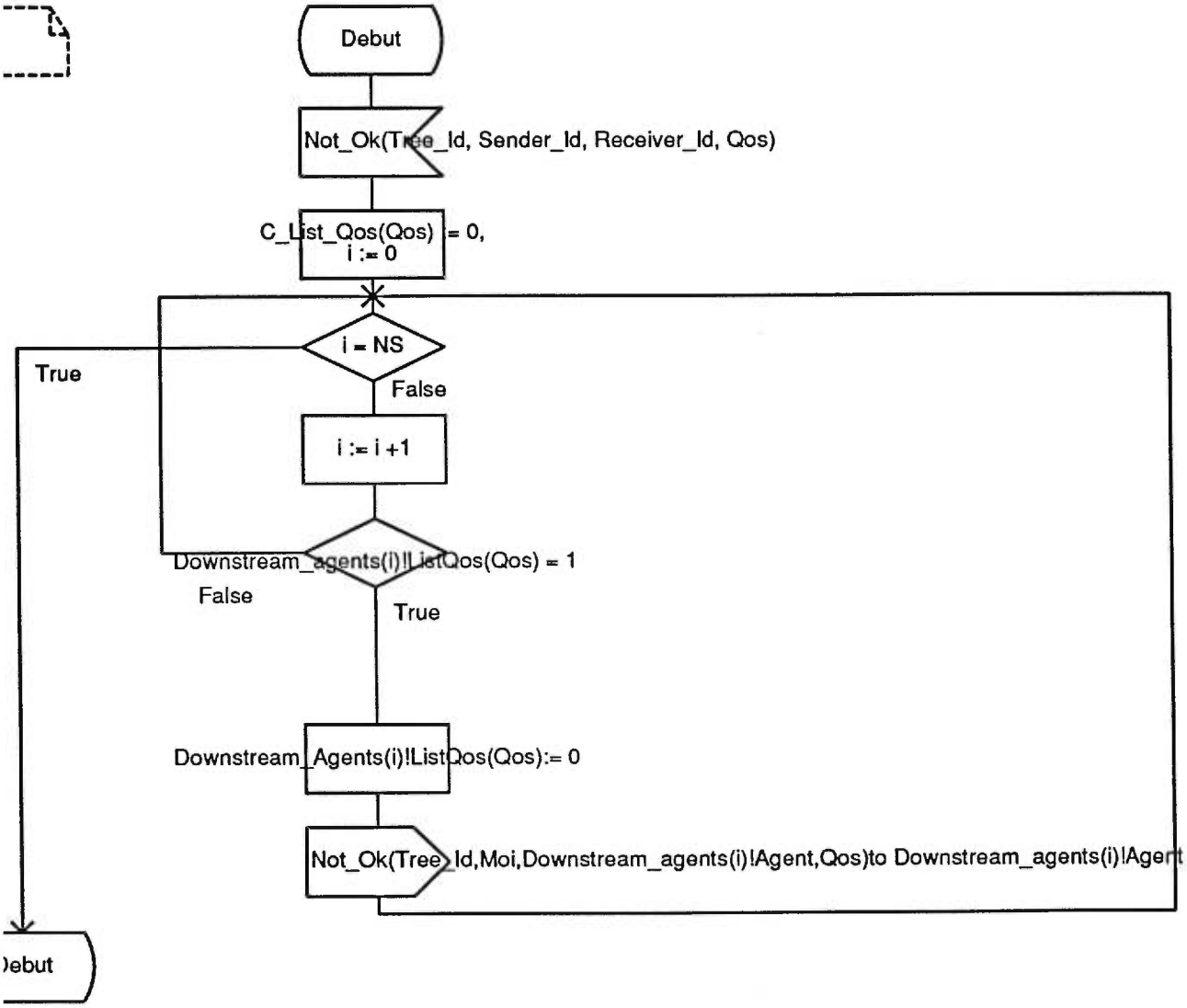

ss Tree_Manager
Routerop1(12)
Debut
Update_DV_R
Detect_Change(Tree_Id,Qos)
C_List_Qos(Qos) = 1
dcl Rand Real;
False
True
Random(Rand)
Rand < 0.5
True
False
C_List_Qos(Qos) := 0
V_List_Qos1(Qos):=1
V_List_Qos := (. 0 .),
V_List_Qos(Qos):=1
Update_DV_R
Viol(Tree_Id,Moi,UpStream,V_List_Qos) to UpStream
Remove_Qos(Tree_Id,Moi,UpStream,Qos) to UpStream
set(now+T_Valeur,Trec)
Debut
debut

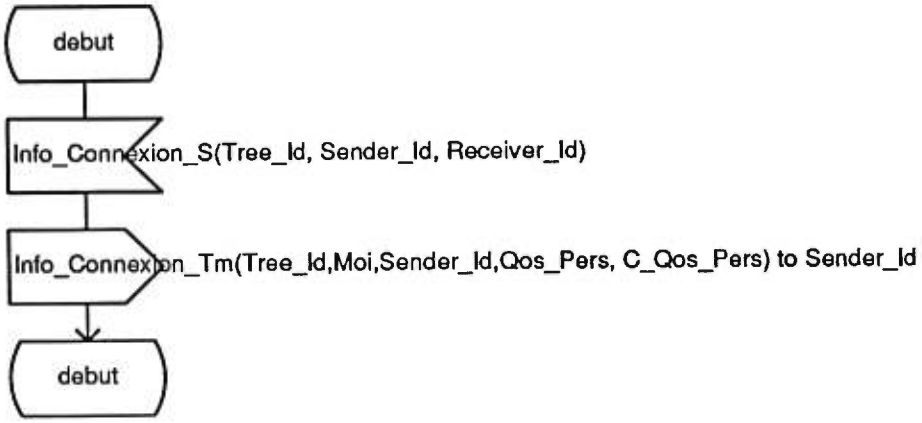


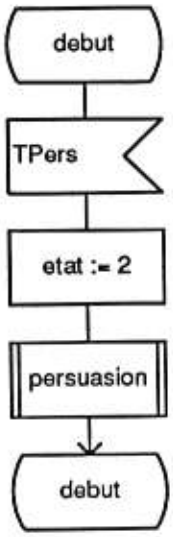

```
dcl i integer;
dcl N_Rec integer := 4;
```

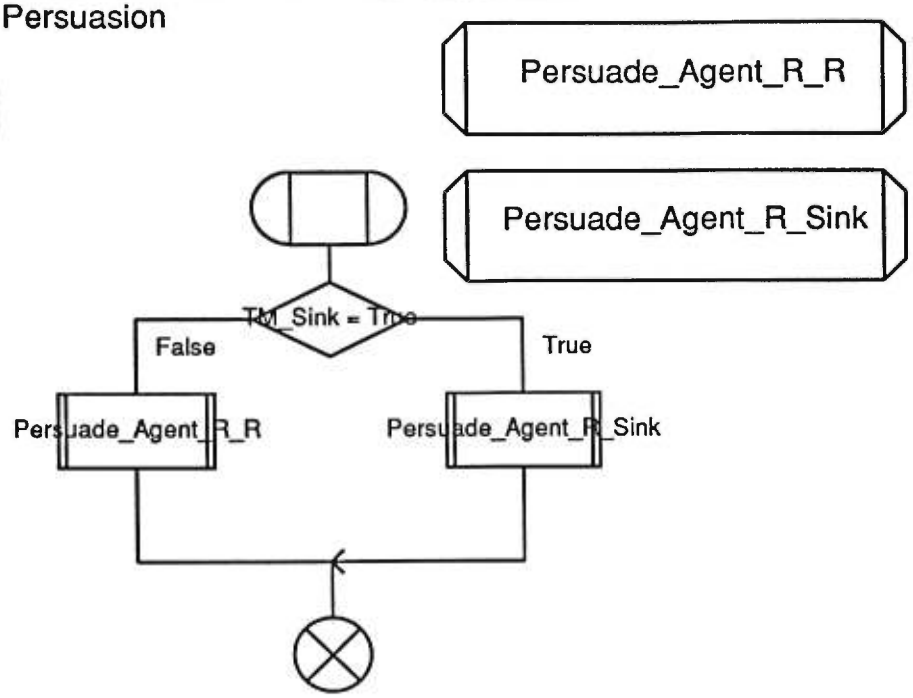

ss Tree_Manager
Routerop4(12)
Debut
Update_Add_R
Add_Qos_Info(Tree_Id,Sender_Id,Receiver_Id,Qos)
C_List_Qos(Qos) = 1
False
True
V_List_Qos1(Qos) = 1
False
True
Not_Ok(Tree_Id, Moi, Sender_Id, Qos) to Sender_Id
Qos_Info(Tree_Id,Moi,UpStream,Qos) to UpStream
Debut
set(NOW + T_Valeur, Tpersuasion)
debut

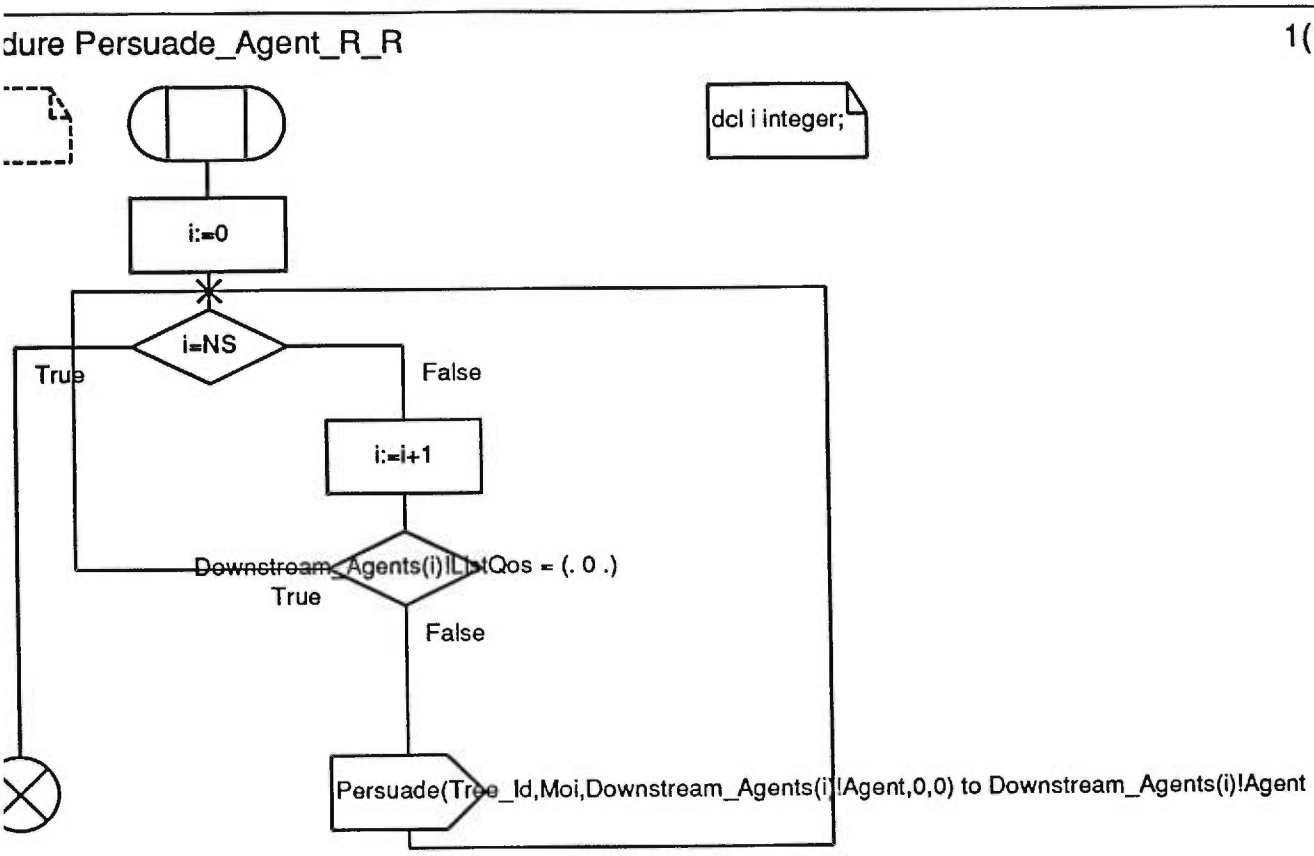

ss Tree_Manager
Routerop5(12)
Debut
Remove_Qos(Tree_Id,Sender_Id,Receiver_Id,Qos)
enlever_R
Chercher_Qos_R
Trouve=1
True
False
C_List_Qos(Qos) := 0
Remove_Qos(Tree_Id,Moi,UpStream,Qos)to UpStream
IOW + T_Valeur, Tpersuasion)
Debut
Enlever_R
Chercher_Qos_R

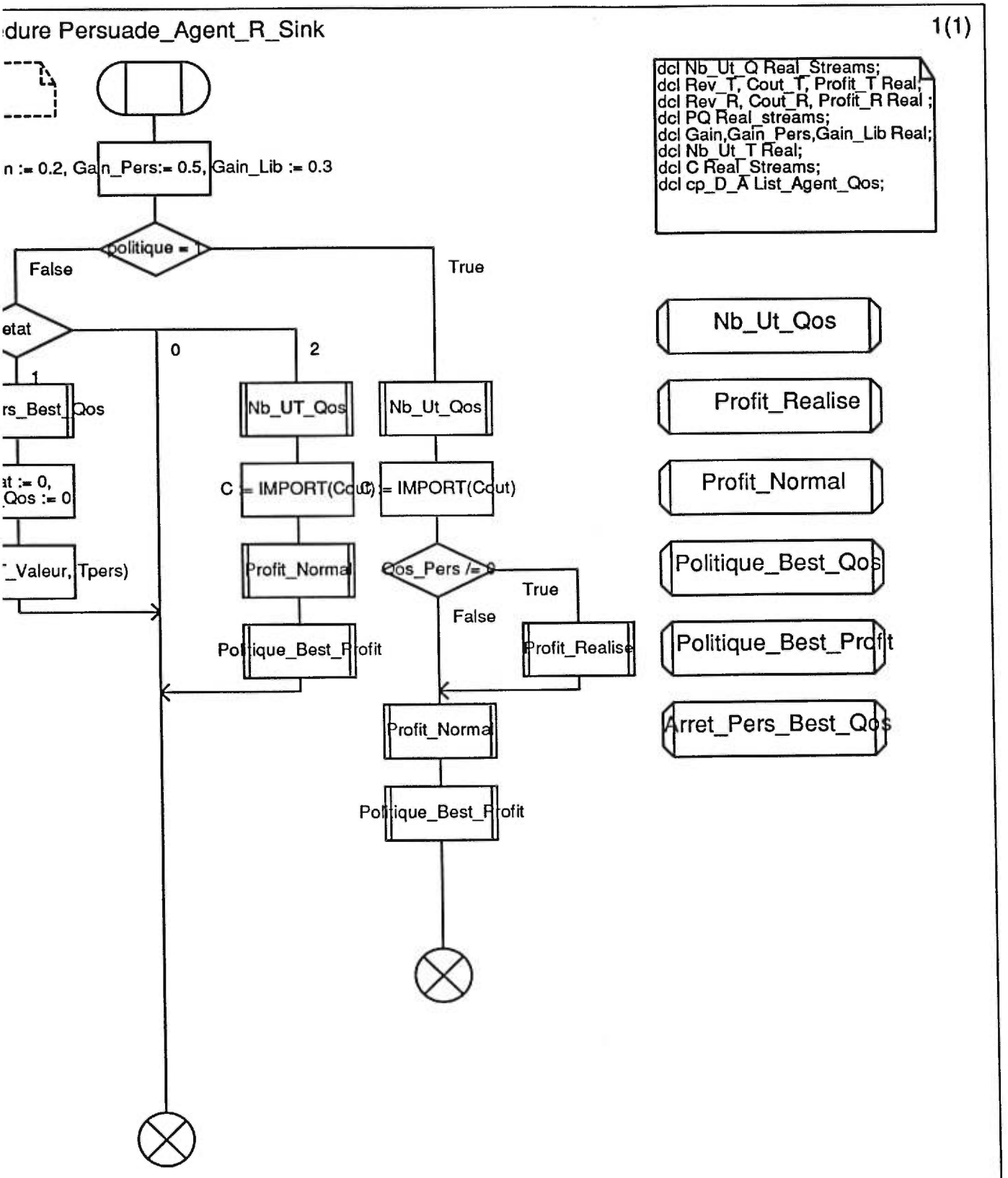

Debut
Not_Ok(Tree_Id, Sender_Id, Receiver_Id, Qos)
C_List_Qos(Qos) = 0,
i := 0
i = NS
True
False
i := i +1
Downstream_agents(i)!ListQos(Qos) = 1
False
True
Downstream_Agents(i)!ListQos(Qos):= 0
Not_Ok(Tree_Id,Moi,Downstream_agents(i)!Agent,Qos)to Downstream_agents(i)!Agent
)ebut

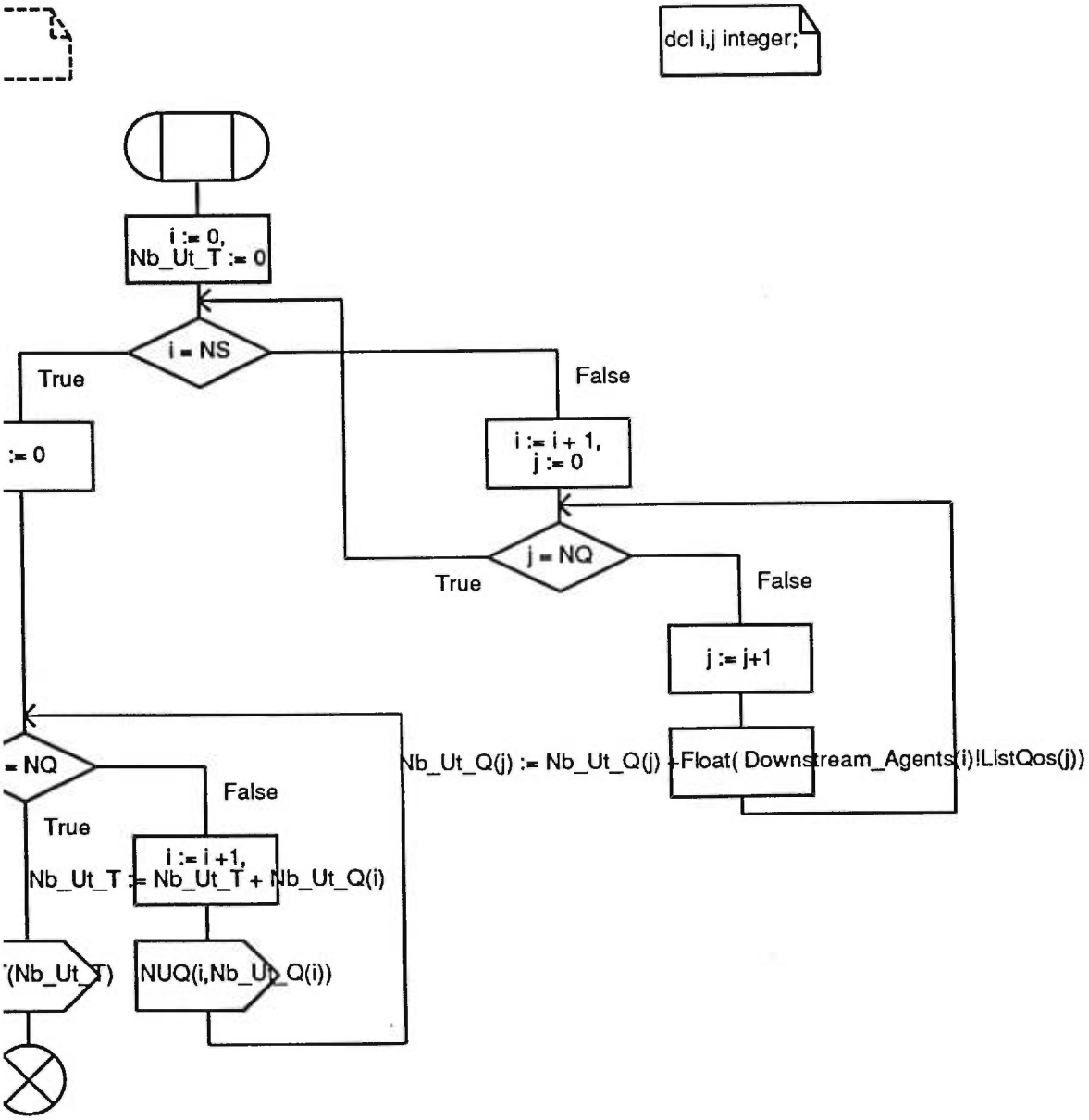

debut
Info_Connexion_S(Tree_Id, Sender_Id, Receiver_Id)
Info_Connexion_Tm(Tree_Id,Moi,Sender_Id,Qos_Pers, C_Qos_Pers) to Sender_Id
debut

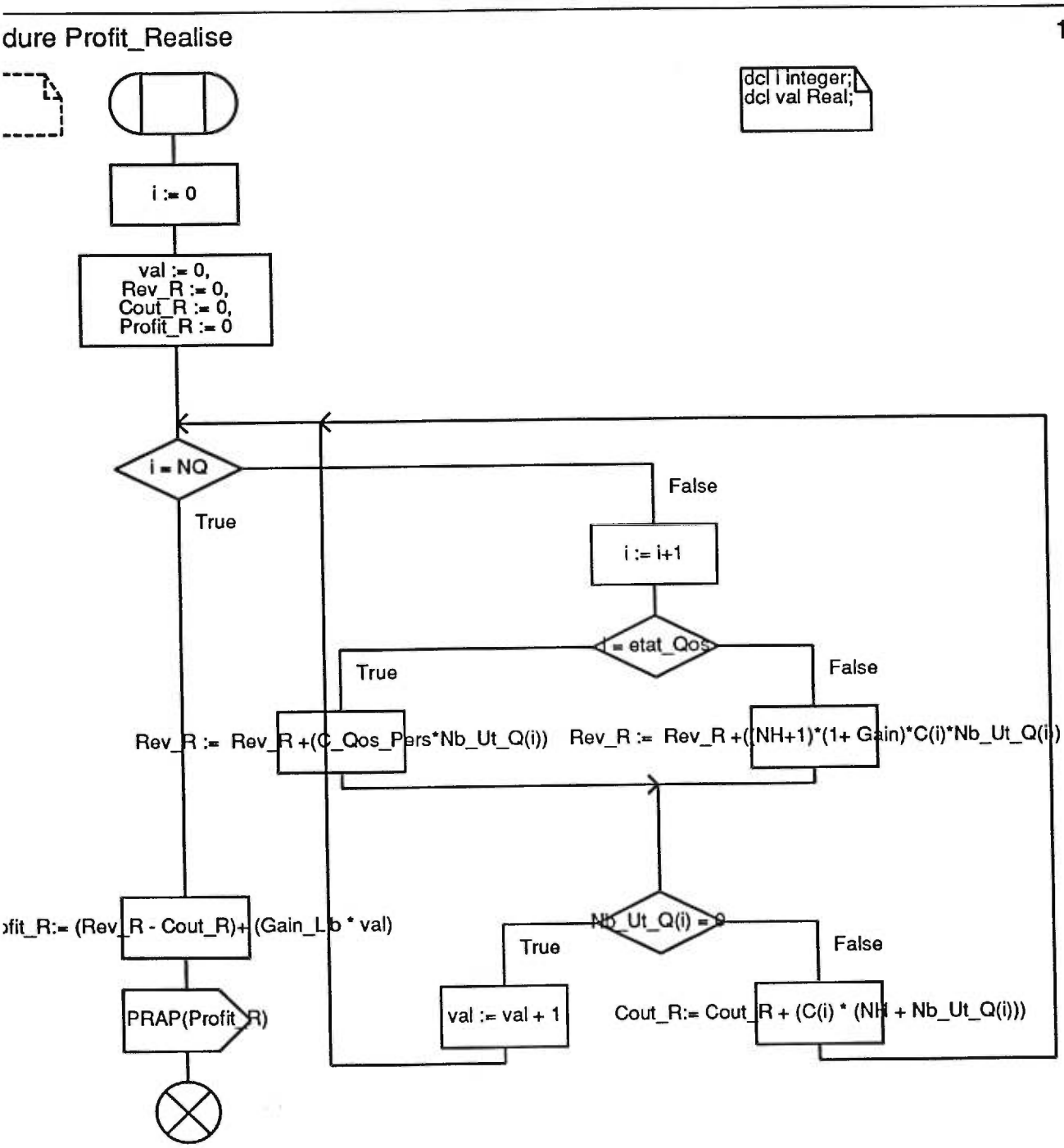

debut
TPers
etat := 2
persuasion
debut


Persuade_Agent_R_R
Persuade_Agent_R_Sink
TM_Sink = True
False
True
Persuade_Agent_R_R
Persuade_Agent_R_Sink

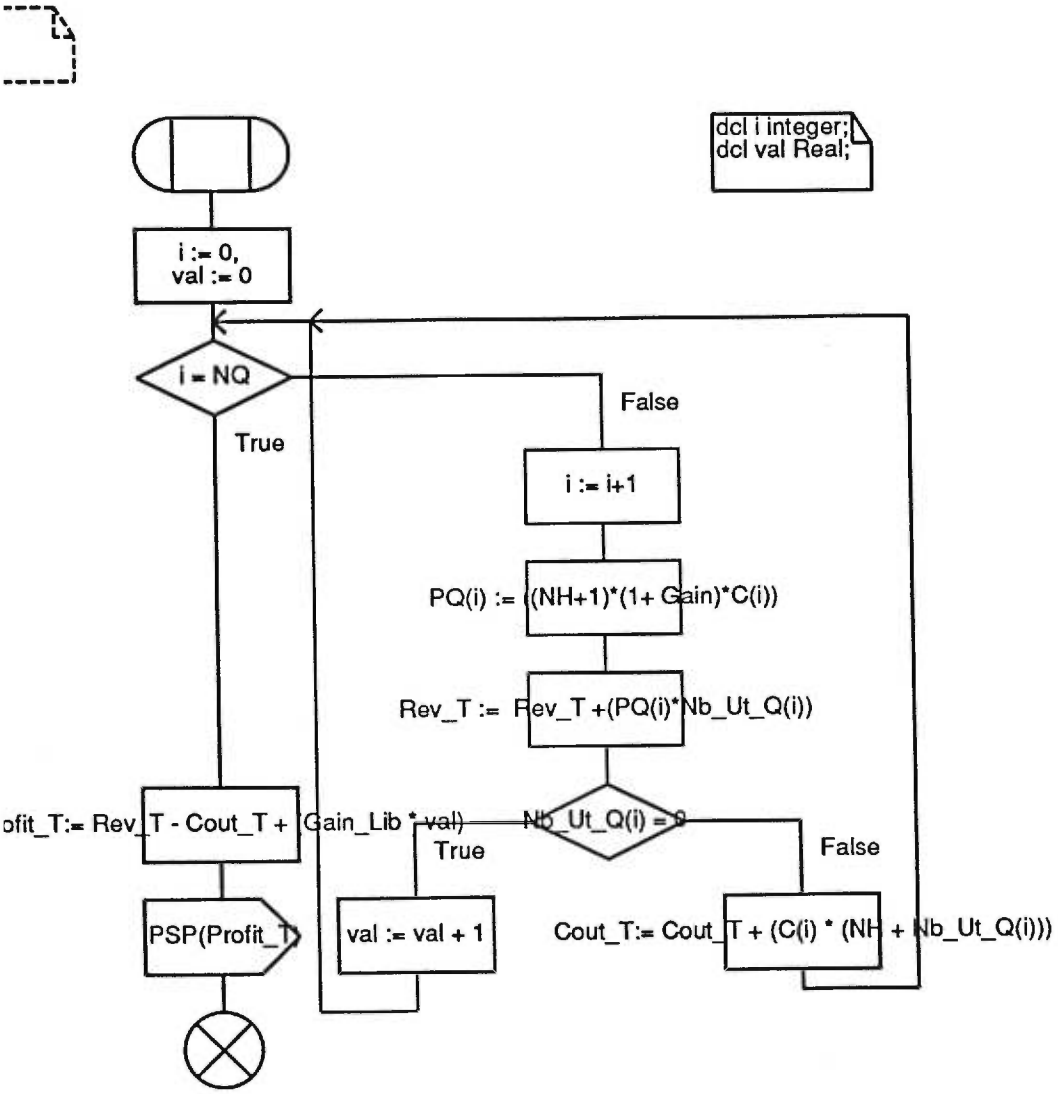

dure Persuade_Agent_R_R
1(1)
dcl i integer;
i:=0
i=NS
True
False
i:=i+1
Downstream_Agents(i)!L_tQos = (. 0 .)
True
False
Persuade(Tree_Id,Moi,Downstream_Agents(i)!Agent,0,0) to Downstream_Agents(i)!Agent


dure Persuade_Agent_R_Sink
1(1)
dcl Nb_Ut_Q Real_Streams;
dcl Rev_T, Cout_T, Profit_T Real;
dcl Rev_R, Cout_R, Profit_R Real ;
dcl PQ Real_streams;
dcl Gain,Gain_Pers,Gain_Lib Real;
dcl Nb_Ut_T Real;
dcl C Real_Streams;
dcl cp_D_A List_Agent_Qos;
n := 0.2, Gain_Pers:= 0.5, Gain_Lib := 0.3
politique =
False
True
etat
0
2
1
rs_Best_Qos
Nb_UT_Qos
Nb_Ut_Qos
t := 0,
Qos := 0
C := IMPORT(Cout)
= IMPORT(Cout)
_Valeur, Tpers)
Profit_Normal
Qos_Pers /= 0
True
False
Politique_Best_Profit
Profit_Realise
Profit_Normal
Politique_Best_Profit
Nb_Ut_Qos
Profit_Realise
Profit_Normal
Politique_Best_Qos
Politique_Best_Profit
Arret_Pers_Best_Qos


dcl i,j integer;
i := 0,
Nb_Ut_T := 0
i = NS
True
False
:= 0
i := i + 1,
j := 0
j = NQ
True
False
j := j+1
Nb_Ut_Q(j) := Nb_Ut_Q(j) +Float(Downstream_Agents(i)!ListQos(j))
= NQ
False
True
i := i +1,
Nb_Ut_T := Nb_Ut_T + Nb_Ut_Q(i)
(Nb_Ut_T)
NUQ(i,Nb_Ut_Q(i))

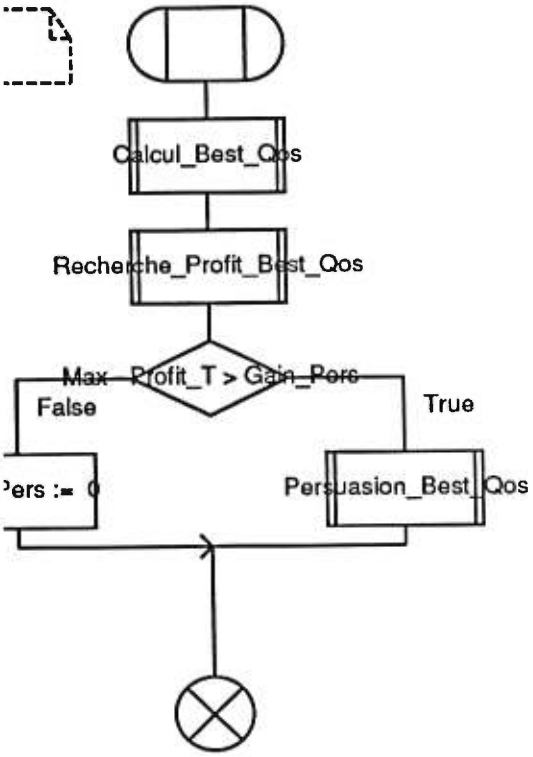

dure Profit_Realise
1(1)
dcl i integer;
dcl val Real;
i := 0
val := 0,
Rev_R := 0,
Cout_R := 0,
Profit_R := 0
i = NQ
True
False
i := i+1
i = etat_Qos
True
False
Rev_R := Rev_R +(C_Qos_Pers*Nb_Ut_Q(i))
Rev_R := Rev_R +((NH+1)*(1+ Gain)*C(i)*Nb_Ut_Q(i))
Nb_Ut_Q(i) = 0
True
False
val := val + 1
Cout_R:= Cout_R + (C(i) * (NH + Nb_Ut_Q(i)))
PRAP(Profit_R)

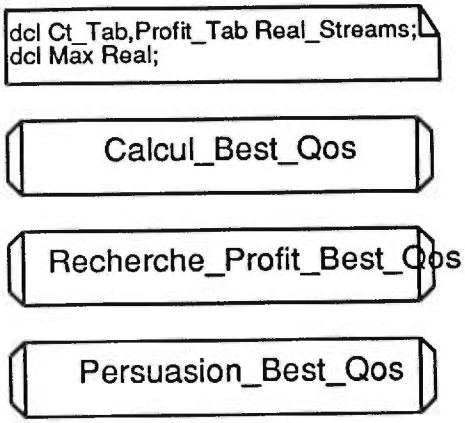
dure Profit_Normal

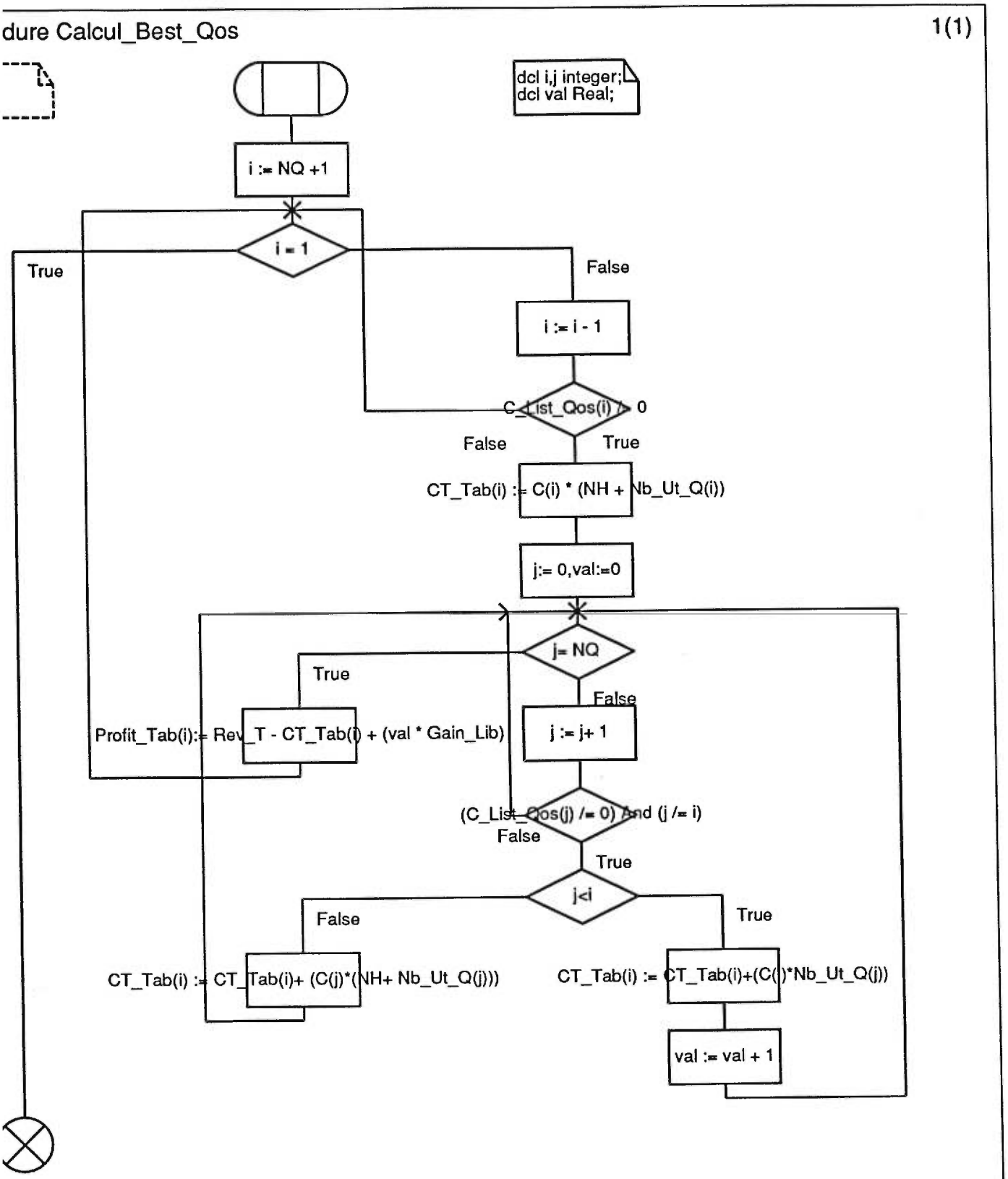


dure Politique_Best_Qos

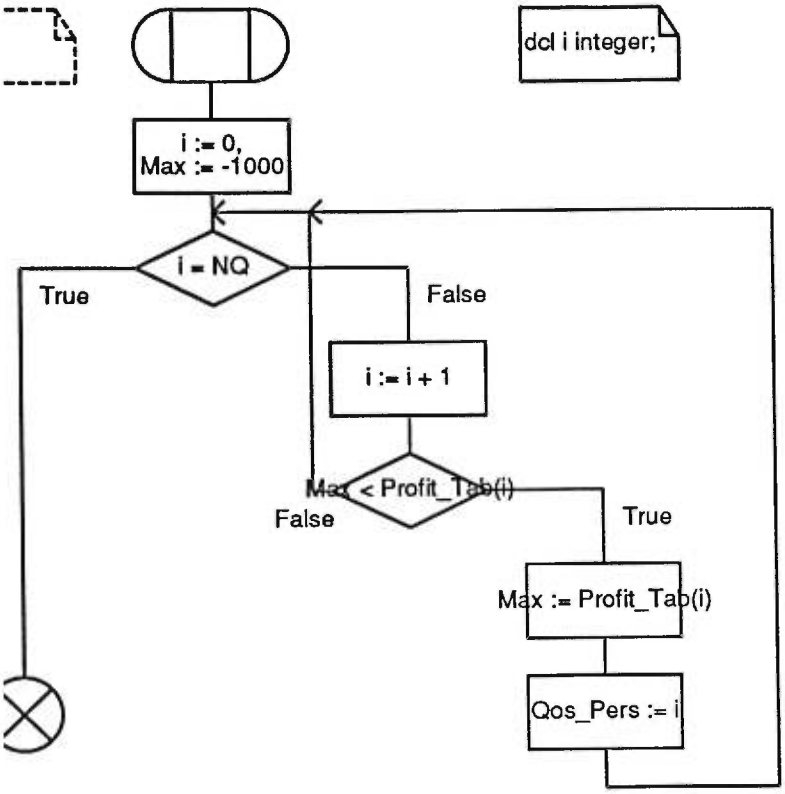
1(1)

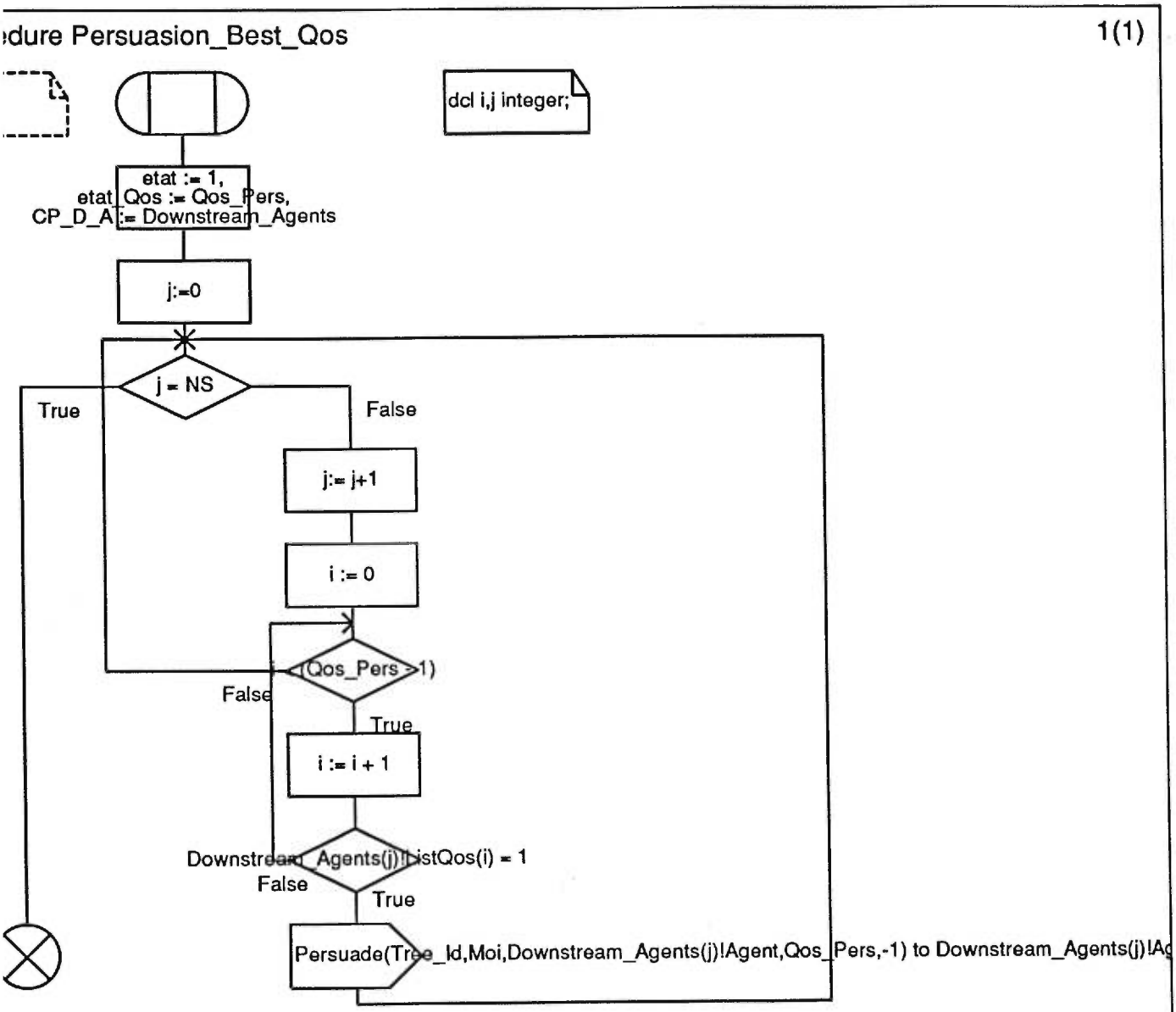


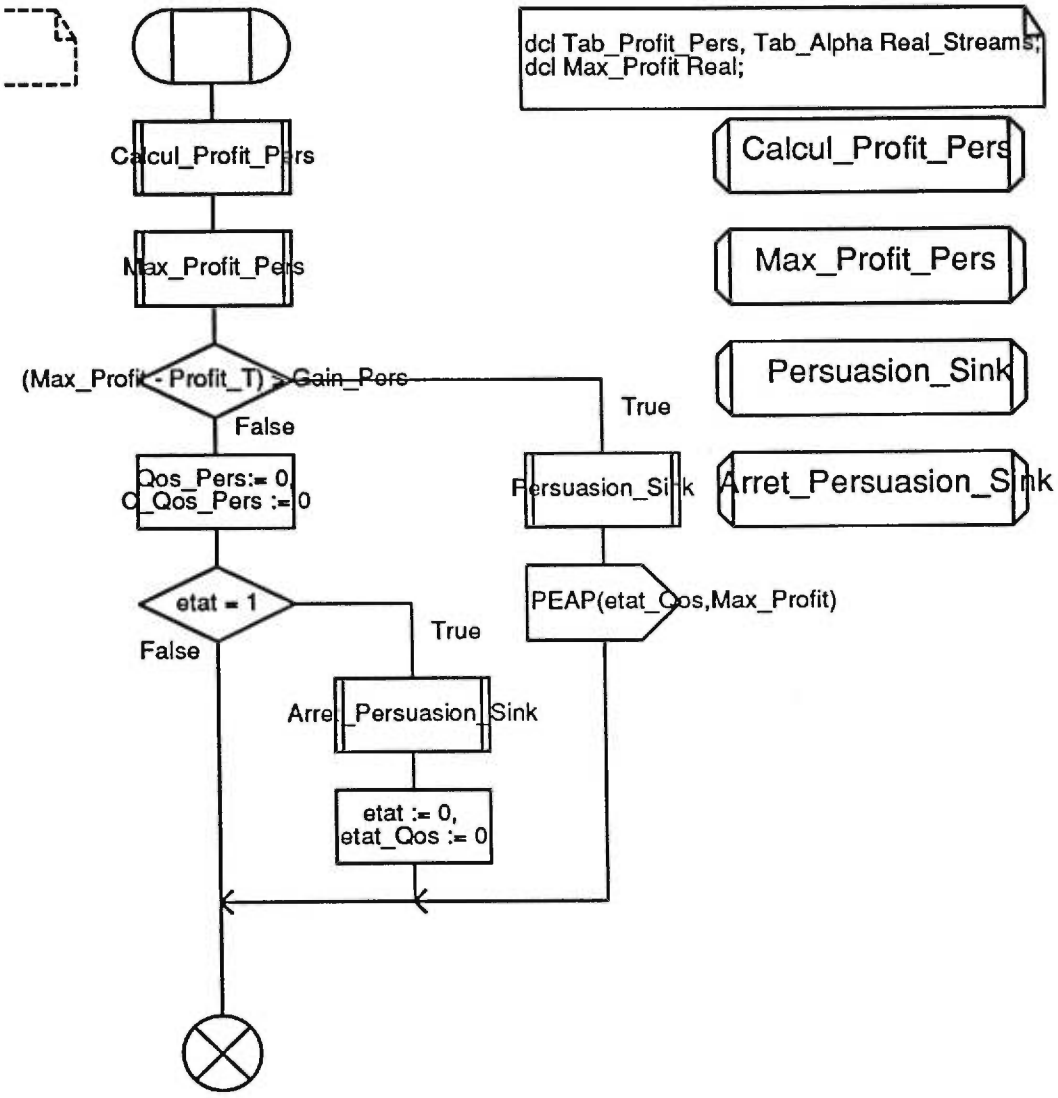


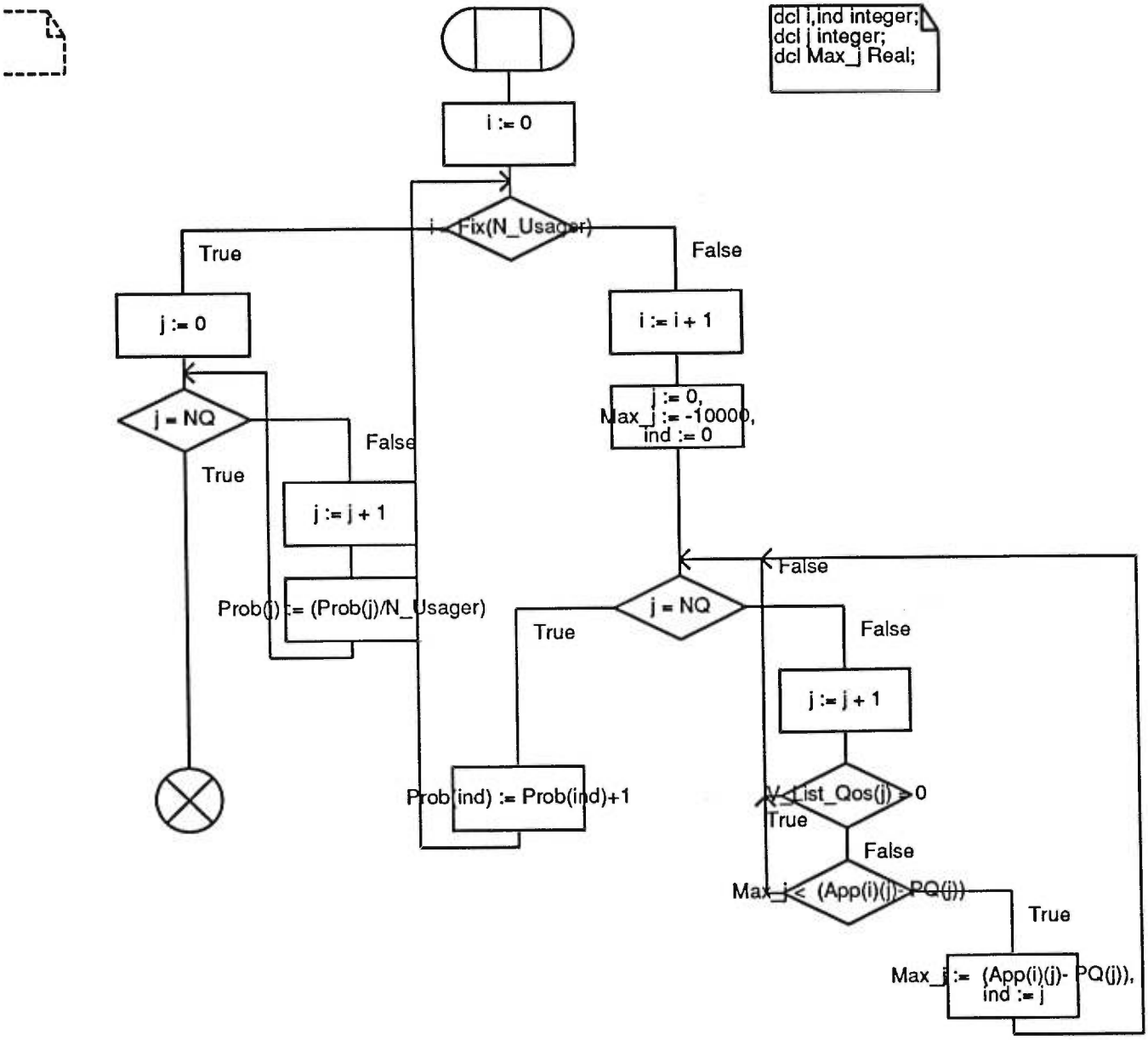

dure Calcul_Best_Qos
1(1)
dcl i,j integer;
dcl val Real;
i := NQ +1
i = 1
True
False
i := i - 1
C_List_Qos(i) /= 0
False
True
CT_Tab(i) := C(i) * (NH + Nb_Ut_Q(i))
j:= 0,val:=0
j= NQ
True
False
Profit_Tab(i):= Rev_T - CT_Tab(i) + (val * Gain_Lib)
j := j+ 1
(C_List_Qos(j) /= 0) And (j /= i)
False
True
j<i
False
True
CT_Tab(i) := CT_Tab(i)+ (C(j)*(NH+ Nb_Ut_Q(j)))
CT_Tab(i) := CT_Tab(i)+(C(j)*Nb_Ut_Q(j))
val := val + 1

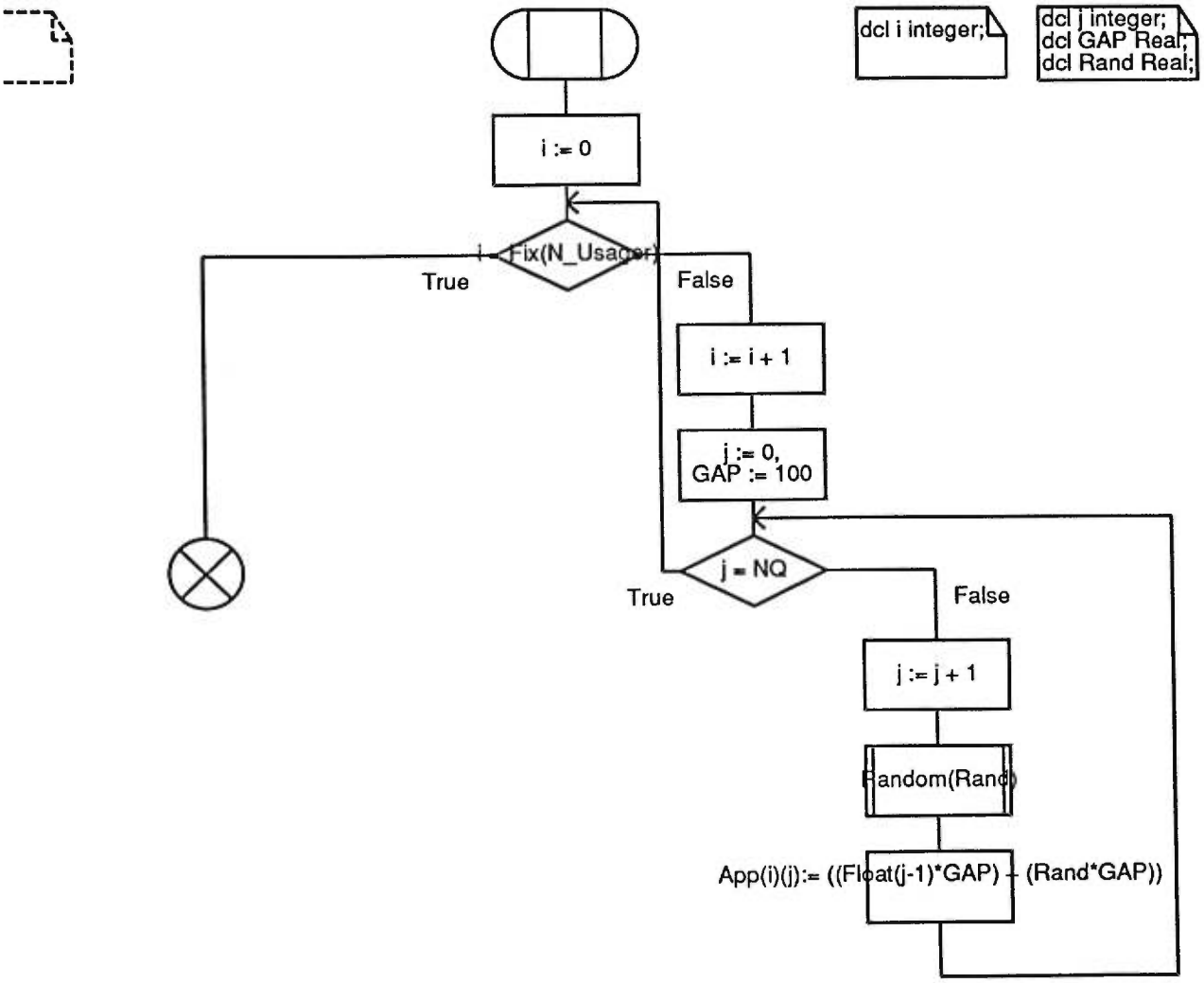
dure Recherche_Profit_Best_Qos 1(1)

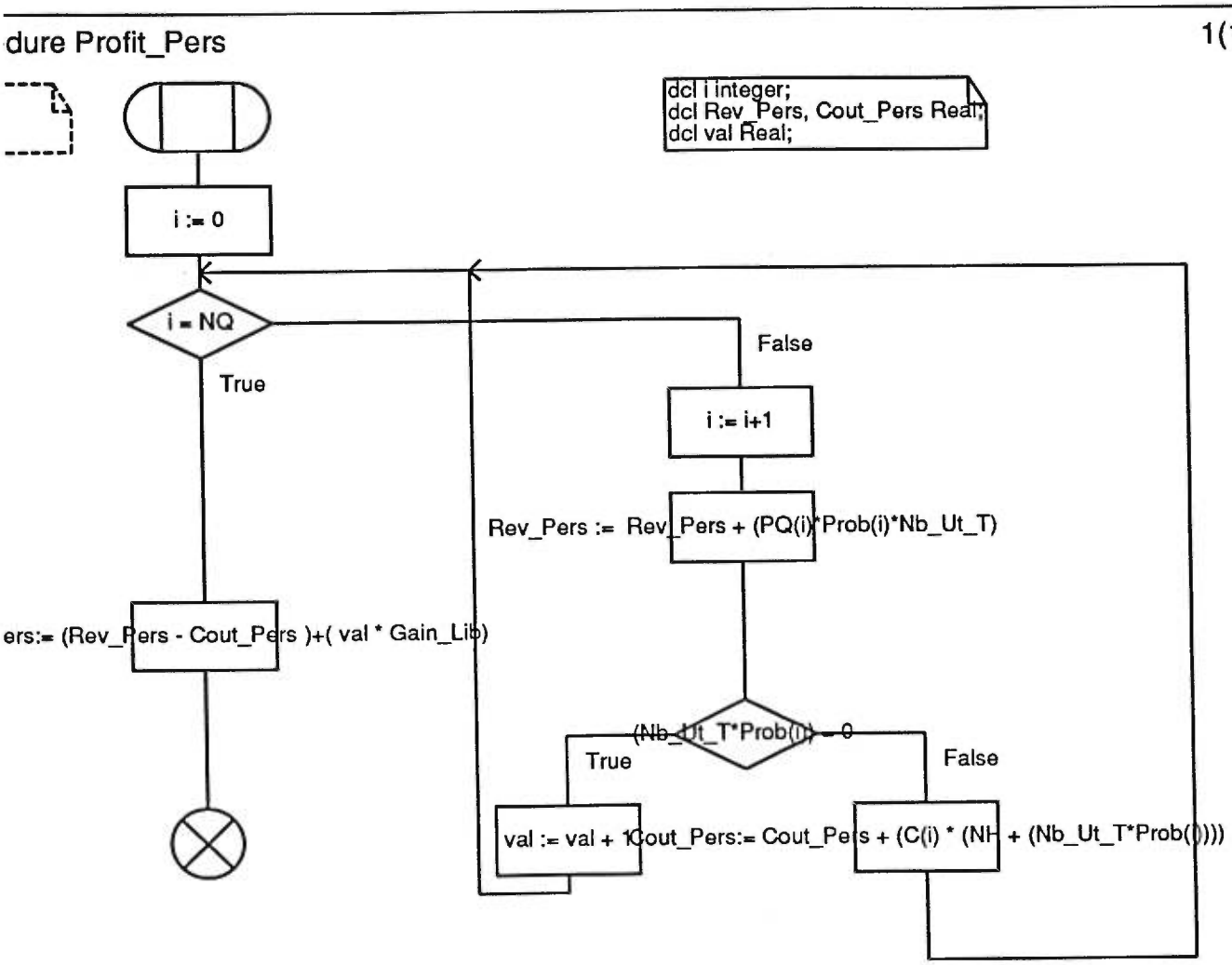


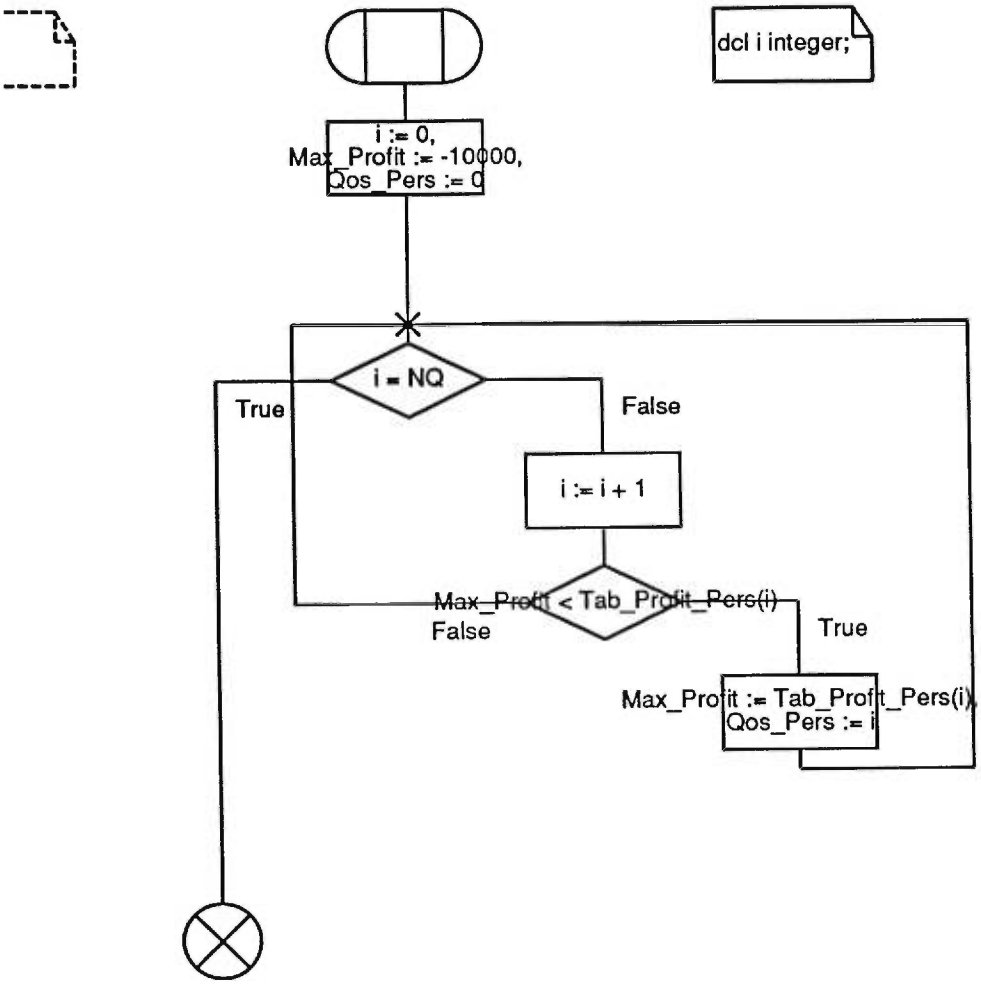

dure Persuasion_Best_Qos
1(1)
dcl i,j integer;
etat := 1,
etat_Qos := Qos_Pers,
CP_D_A := Downstream_Agents
j:=0
j = NS
True
False
j:= j+1
i := 0
(Qos_Pers >1)
False
True
i := i + 1
Downstream_Agents(j)!ListQos(i) = 1
False
True
Persuade(Tree_Id,Moi,Downstream_Agents(j)!Agent,Qos_Pers,-1) to Downstream_Agents(j)!Ag

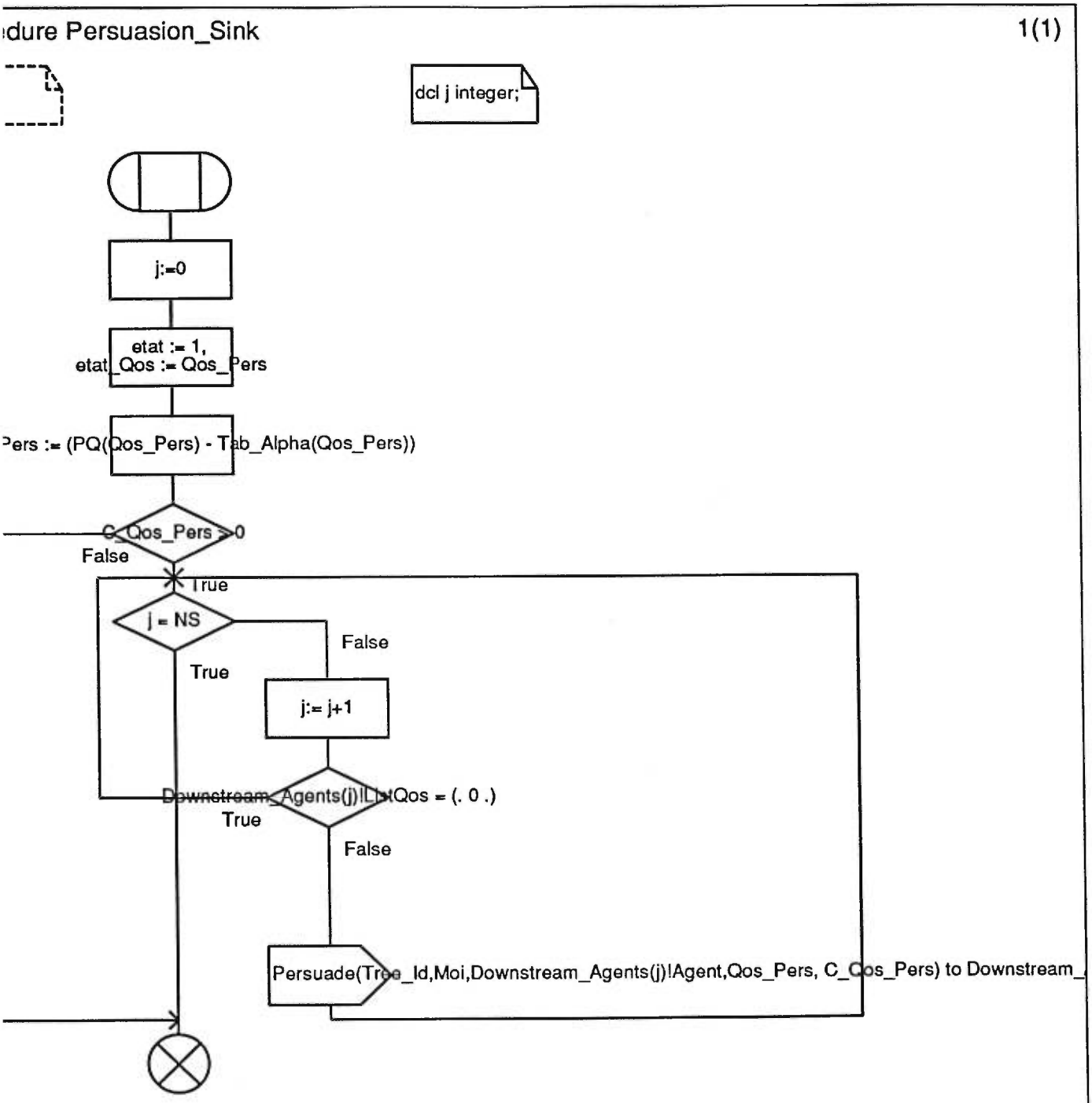

dcl Tab_Profit_Pers, Tab_Alpha Real_Streams;
dcl Max_Profit Real;
Calcul_Profit_Pers
Max_Profit_Pers
Persuasion_Sink
Arret_Persuasion_Sink
Calcul_Profit_Pers
Max_Profit_Pers
(Max_Profit - Profit_T) > Gain_Pers
True
False
Qos_Pers:= 0,
C_Qos_Pers := 0
Persuasion_Sink
etat = 1
True
False
PEAP(etat_Qos,Max_Profit)
Arret_Persuasion_Sink
etat := 0,
etat_Qos := 0

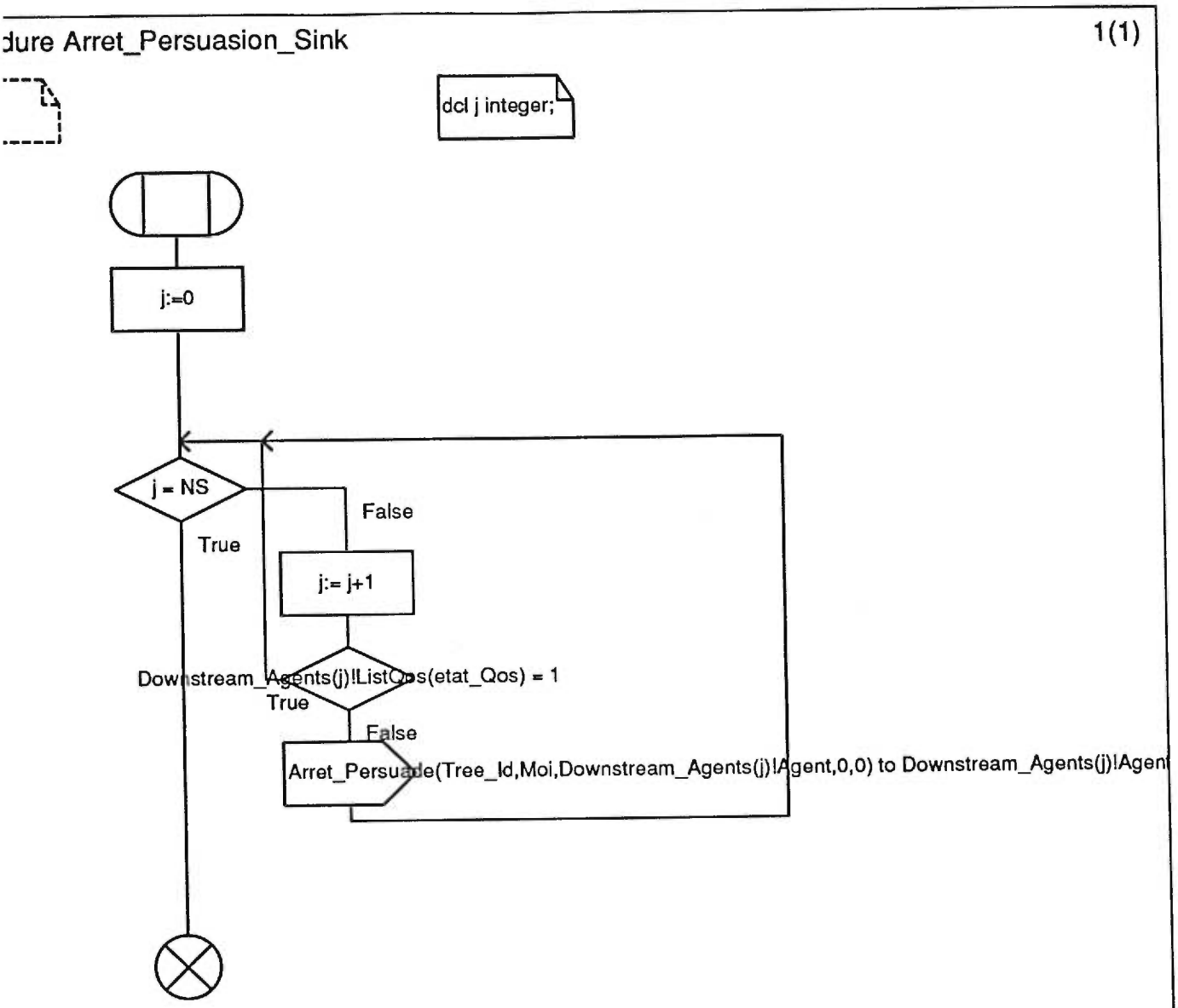

dcl i,ind integer;
dcl j integer;
dcl Max_j Real;
i := 0
i < Fix(N_Usager)
True
False
j := 0
i := i + 1
j = NQ
j := 0,
Max_j := -10000,
ind := 0
False
True
j := j + 1
Prob(j) := (Prob(j)/N_Usager)
False
j = NQ
True
False
j := j + 1
Prob(ind) := Prob(ind)+1
List_Qos(j) > 0
True
False
Max_j < (App(i)(j)-PQ(j))
True
Max_j := (App(i)(j)-PQ(j)),
ind := j

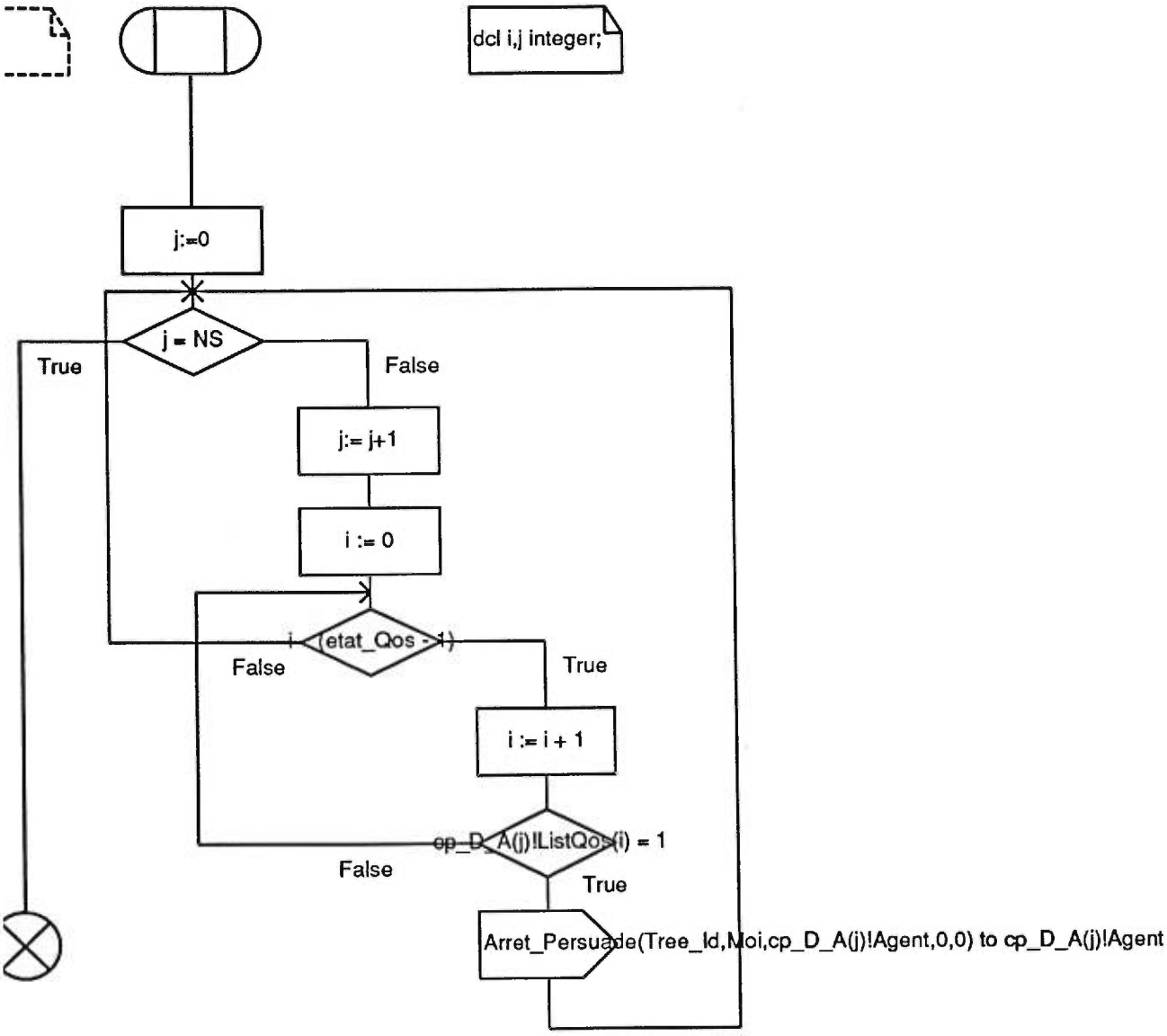

dcl i integer;
dcl j integer;
dcl GAP Real;
dcl Rand Real;
i := 0
i = Fix(N_Usager)
True
False
i := i + 1
j := 0,
GAP := 100
j = NQ
True
False
j := j + 1
Random(Rand)
App(i)(j):= ((Float(j-1)*GAP) + (Rand*GAP))

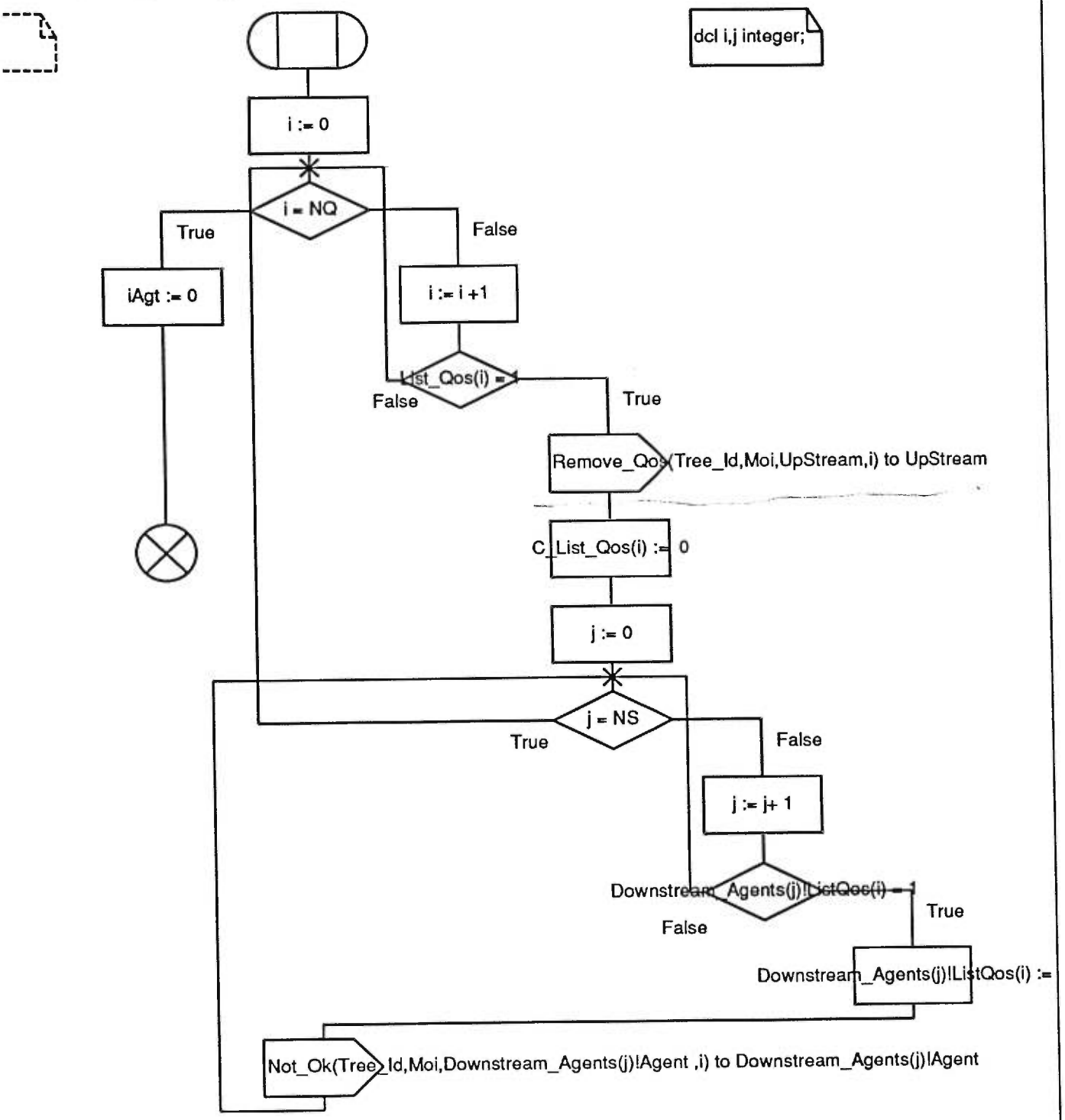

dure Profit_Pers
1(1)
dcl i integer;
dcl Rev_Pers, Cout_Pers Real;
dcl val Real;
i := 0
i = NQ
True
False
i := i+1
Rev_Pers := Rev_Pers + (PQ(i)*Prob(i)*Nb_Ut_T)
ers:= (Rev_Pers - Cout_Pers)+(val * Gain_Lib)
(Nb_Ut_T*Prob(i)) = 0
True
False
val := val + 1
Cout_Pers:= Cout_Pers + (C(i) * (NH + (Nb_Ut_T*Prob())))

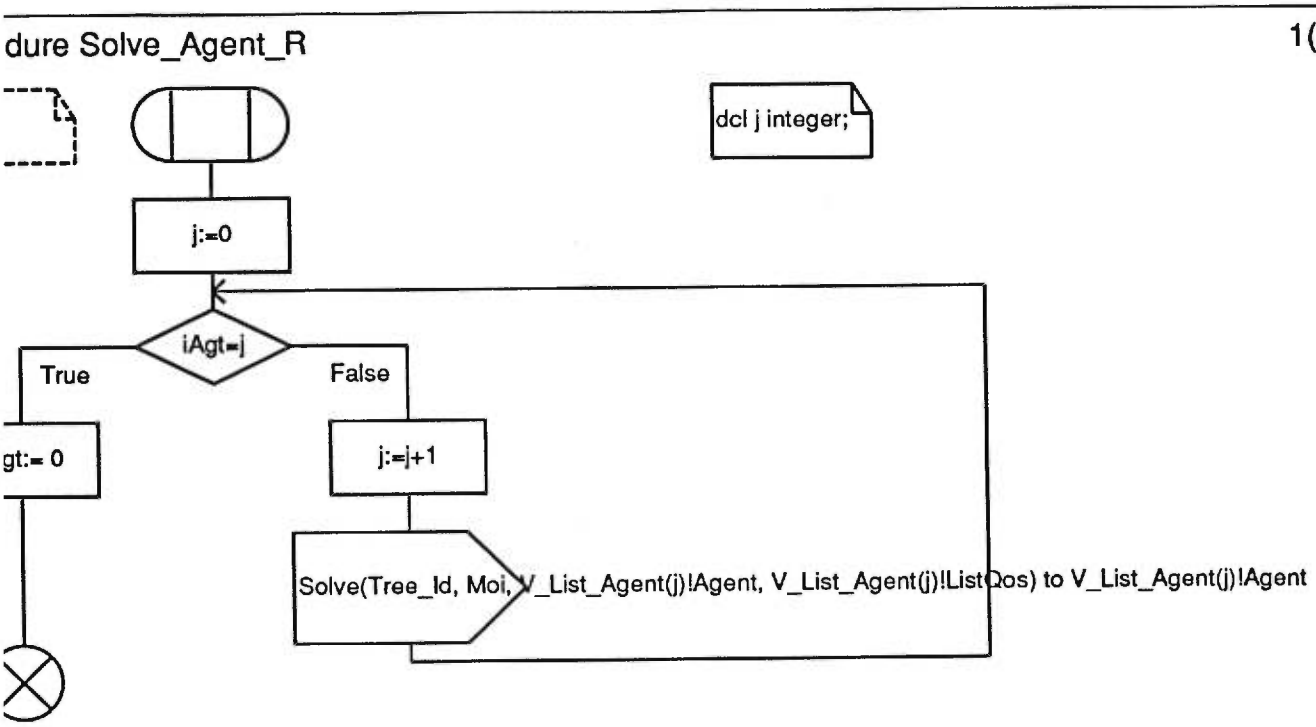

dcl i integer;
i := 0,
Max_Profit := -10000,
Qos_Pers := 0
i = NQ
True
False
i := i + 1
Max_Profit < Tab_Profit_Pers(i)
False
True
Max_Profit := Tab_Profit_Pers(i),
Qos_Pers := i

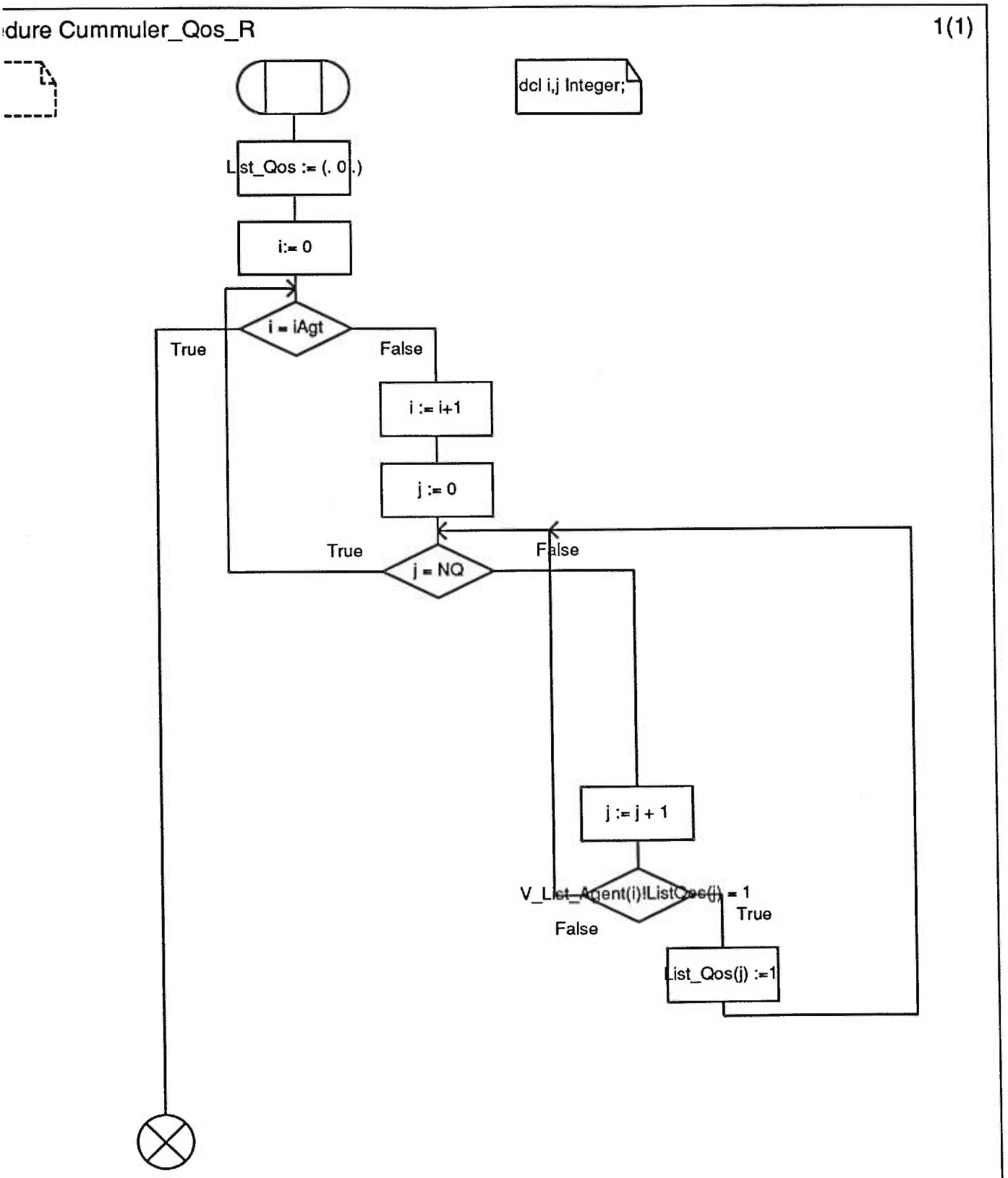

dure Persuasion_Sink
1(1)
dcl j integer;
j:=0
etat := 1,
etat_Qos := Qos_Pers
ers := (PQ(Qos_Pers) - Tab_Alpha(Qos_Pers))
C_Qos_Pers >0
False
True
j = NS
False
True
j:= j+1
Downstream_Agents(j)!Qos = (. 0 .)
True
False
Persuade(Tree_Id,Moi,Downstream_Agents(j)!Agent,Qos_Pers, C_Qos_Pers) to Downstream_

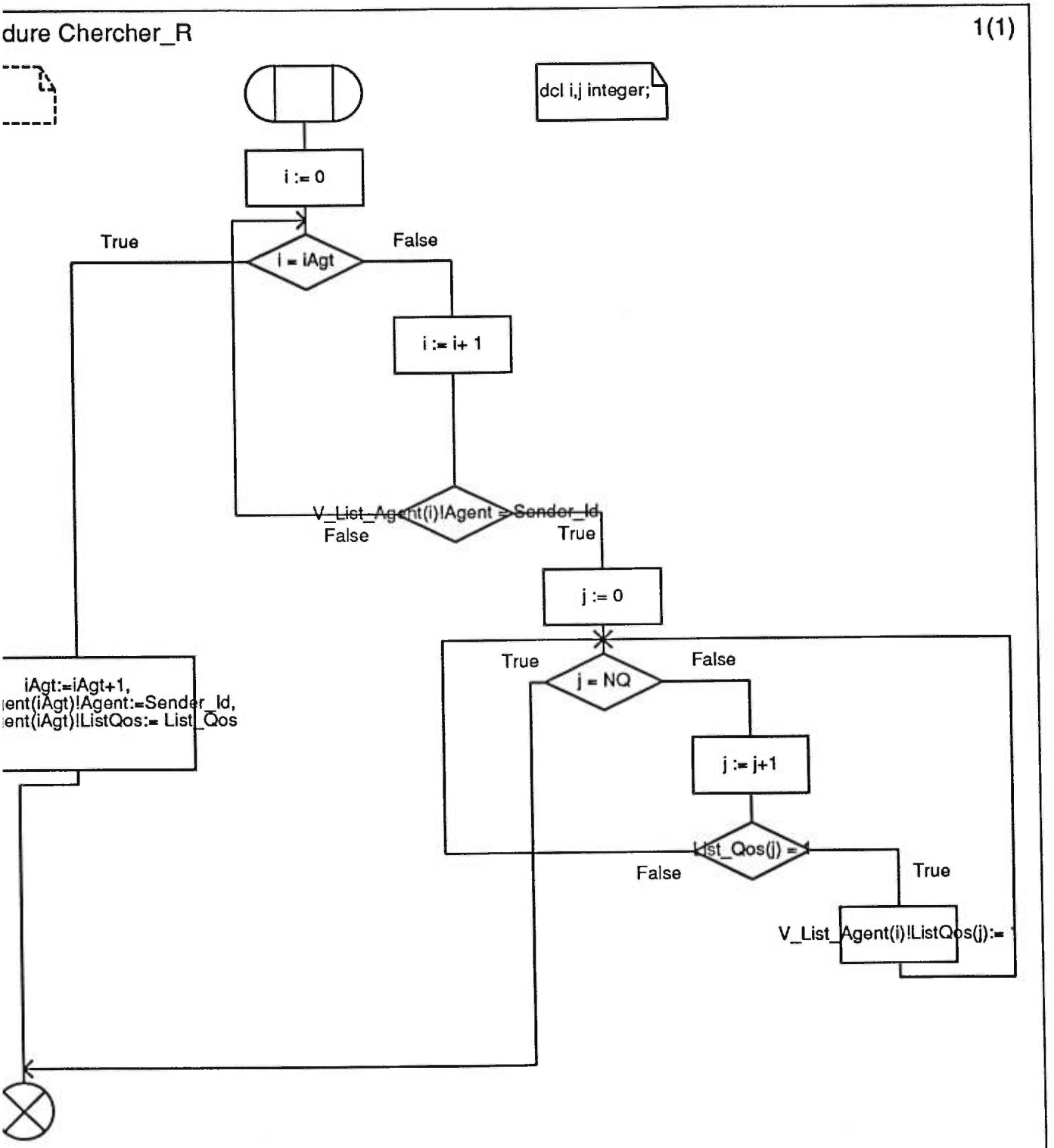

dure Arret_Persuasion_Sink
1(1)
dcl j integer;
j:=0
j = NS
False
True
j:= j+1
Downstream_Agents(j)!ListQos(etat_Qos) = 1
True
False
Arret_Persuade(Tree_Id,Moi,Downstream_Agents(j)!Agent,0,0) to Downstream_Agents(j)!Agent

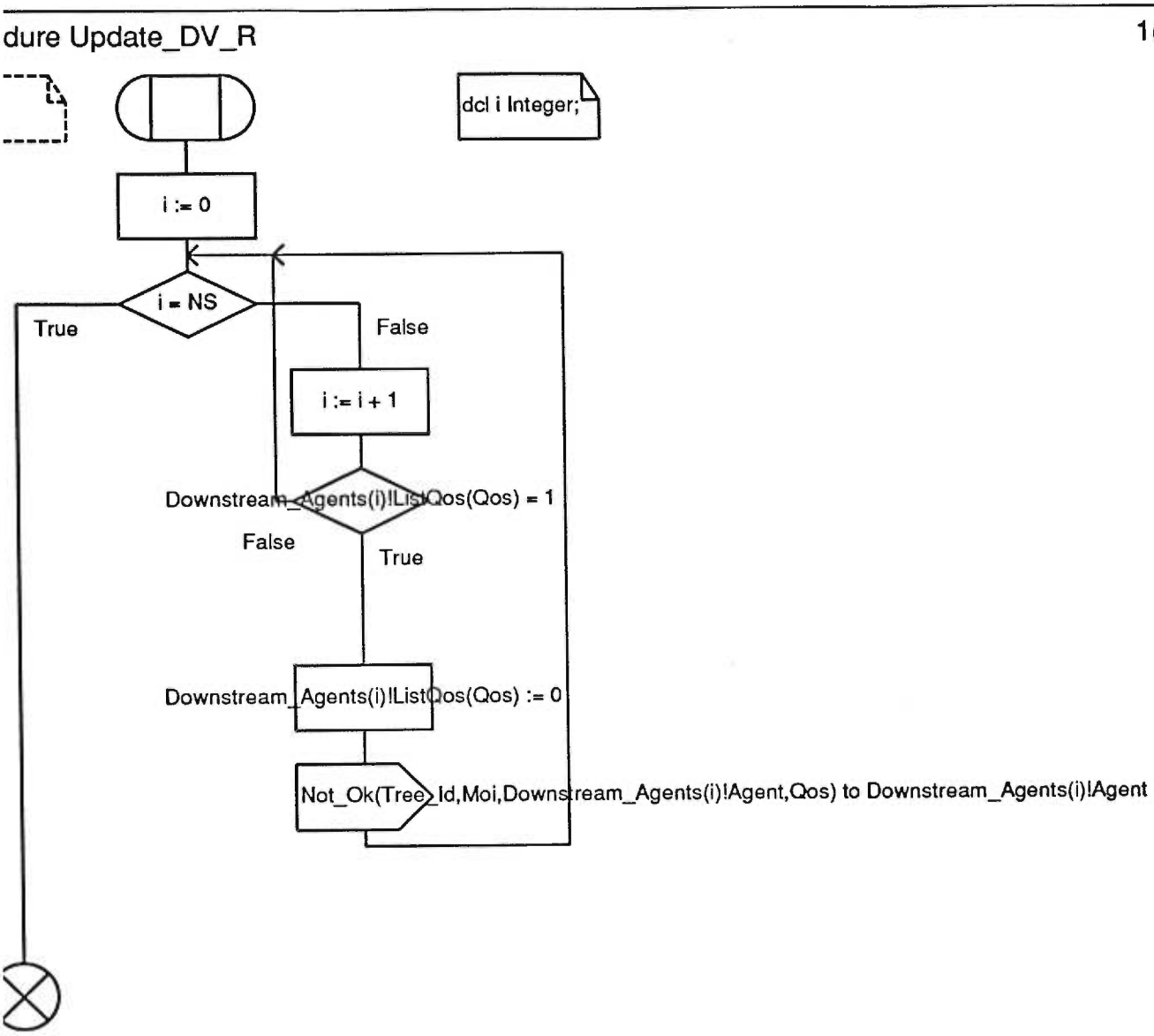

dcl i,j integer;
j:=0
j = NS
True
False
j:= j+1
i := 0
i (etat_Qos = 1)
False
True
i := i + 1
cp_D_A(j)!ListQos(i) = 1
False
True
Arret_Persuade(Tree_Id,Moi,cp_D_A(j)!Agent,0,0) to cp_D_A(j)!Agent

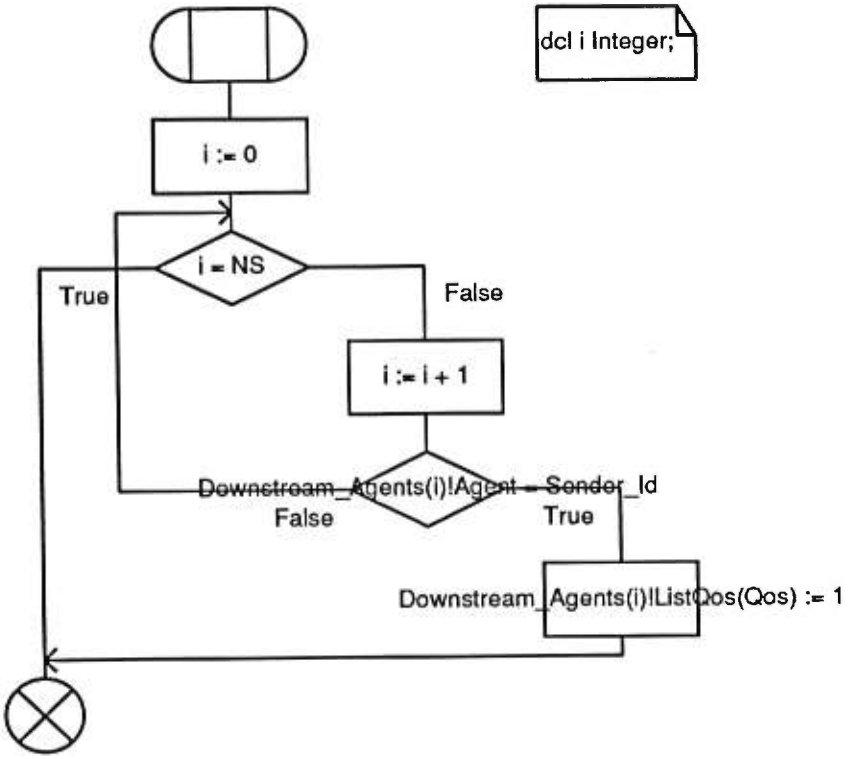

dure Update_Solve_R
1(1)
dcl i,j integer;
i := 0
i = NQ
True
False
iAgt := 0
i := i +1
List_Qos(i) =
False
True
Remove_Qos(Tree_Id,Moi,UpStream,i) to UpStream
C_List_Qos(i) := 0
j := 0
j = NS
True
False
j := j+ 1
Downstream_Agents(j)!ListQos(i) = 1
True
False
Downstream_Agents(j)!ListQos(i) :=
Not_Ok(Tree_Id,Moi,Downstream_Agents(j)!Agent ,i) to Downstream_Agents(j)!Agent


dure Solve_Agent_R
1(1)
dcl j integer;
j:=0
iAgt=j
True
False
gt:= 0
j:=j+1
Solve(Tree_Id, Moi, V_List_Agent(j)!Agent, V_List_Agent(j)!ListQos) to V_List_Agent(j)!Agent


dure Cummuler_Qos_R
1(1)
dcl i,j Integer;
List_Qos := (. 0 .)
i:= 0
i = iAgt
True
False
i := i+1
j := 0
True
j = NQ
False
j := j + 1
V_List_Agent(i)!ListQos(j) = 1
True
False
List_Qos(j) :=1

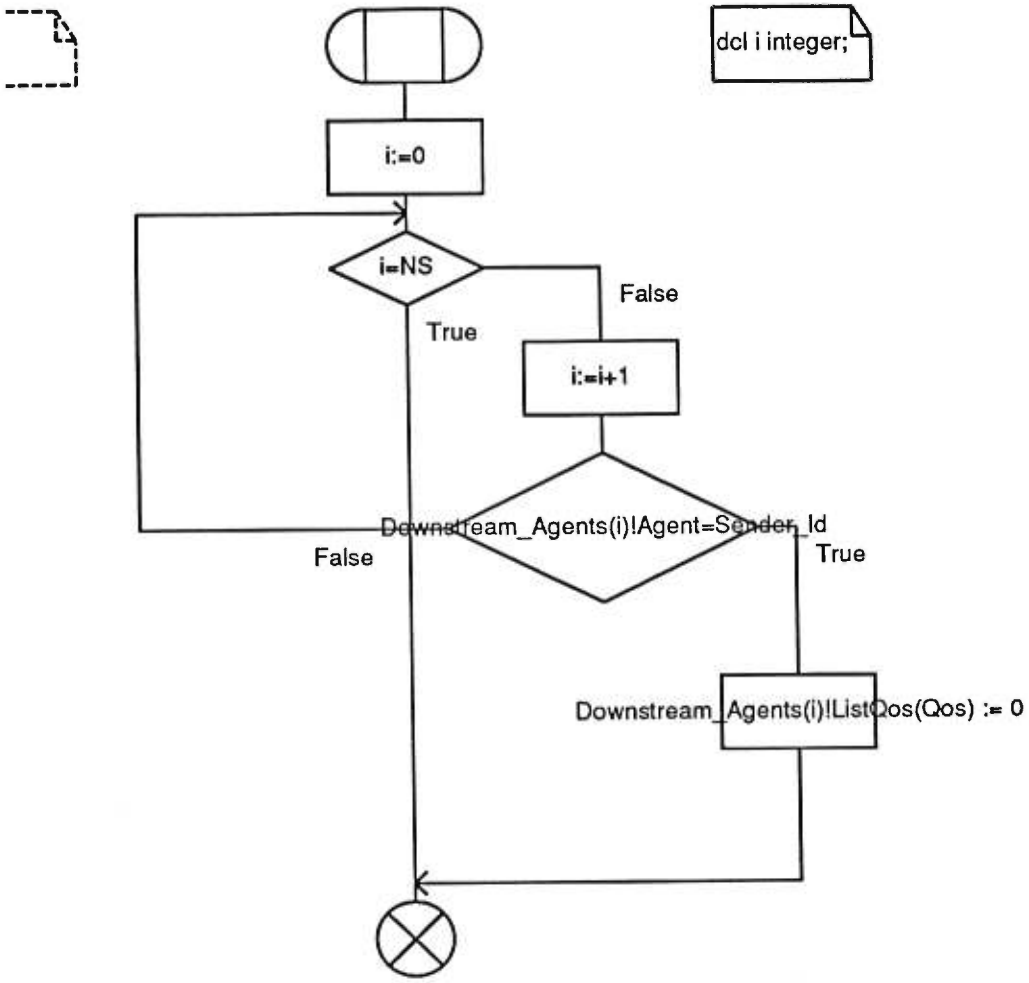

dure Chercher_R
1(1)
dcl i,j integer;
i := 0
True
i = iAgt
False
i := i+ 1
V_List_Agent(i)!Agent = Sender_Id
False
True
j := 0
True
j = NQ
False
iAgt:=iAgt+1,
ent(iAgt)!Agent:=Sender_Id,
ent(iAgt)!ListQos:= List_Qos
j := j+1
List_Qos(j) =
False
True
V_List_Agent(i)!ListQos(j):=


dure Update_DV_R
1(1)
dcl i Integer;
i := 0
i = NS
True
False
i := i + 1
Downstream_Agents(i)!ListQos(Qos) = 1
False
True
Downstream_Agents(i)!ListQos(Qos) := 0
Not_Ok(Tree_Id,Moi,Downstream_Agents(i)!Agent,Qos) to Downstream_Agents(i)!Agent


dcl i Integer;
i := 0
i = NS
True
False
i := i + 1
Downstream_Agents(i)!Agent = Sender_Id
False
True
Downstream_Agents(i)!ListQos(Qos) := 1

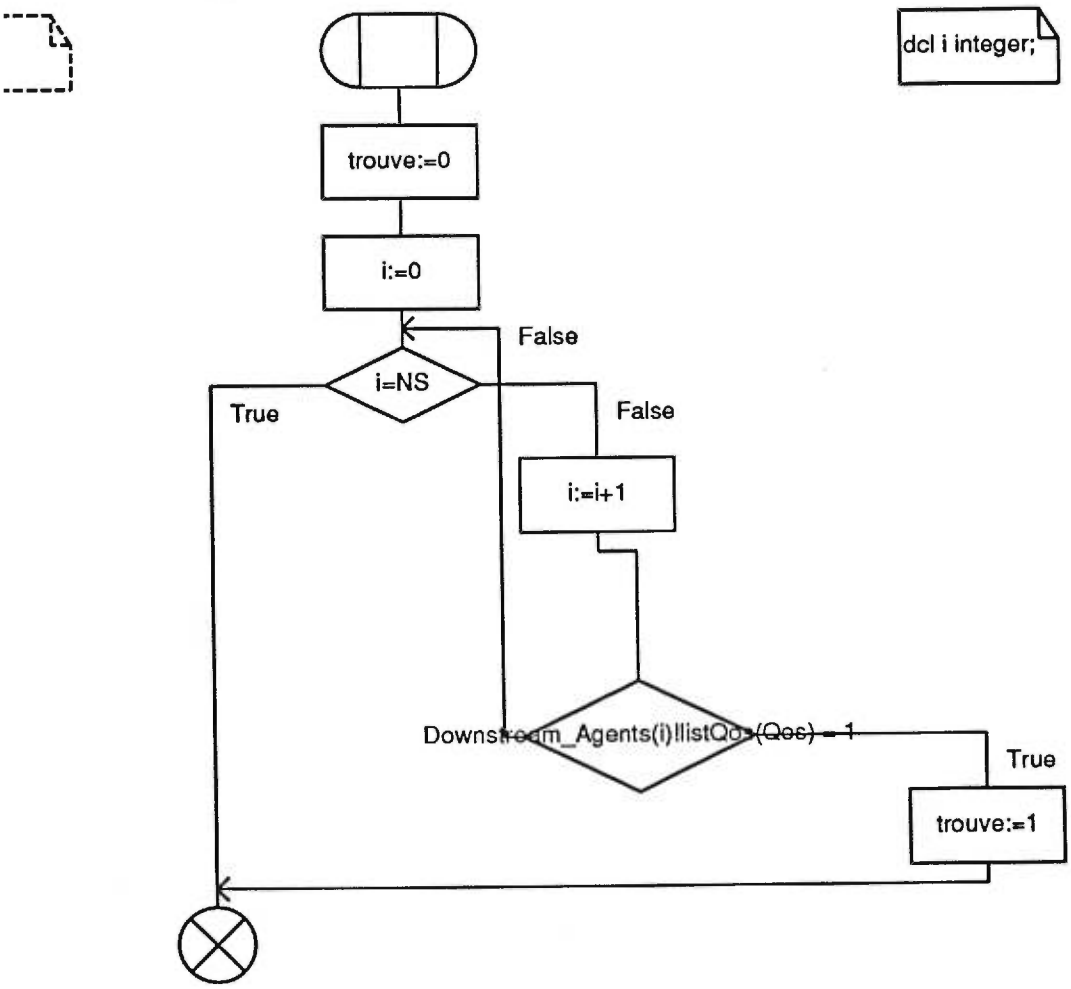
dure Enlever_R

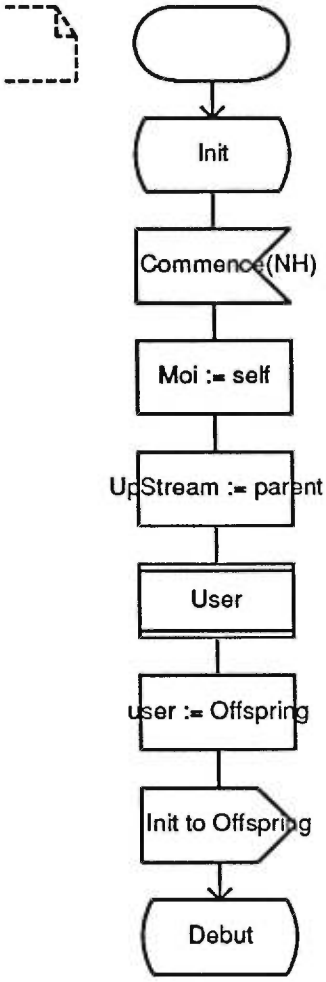
1(1)



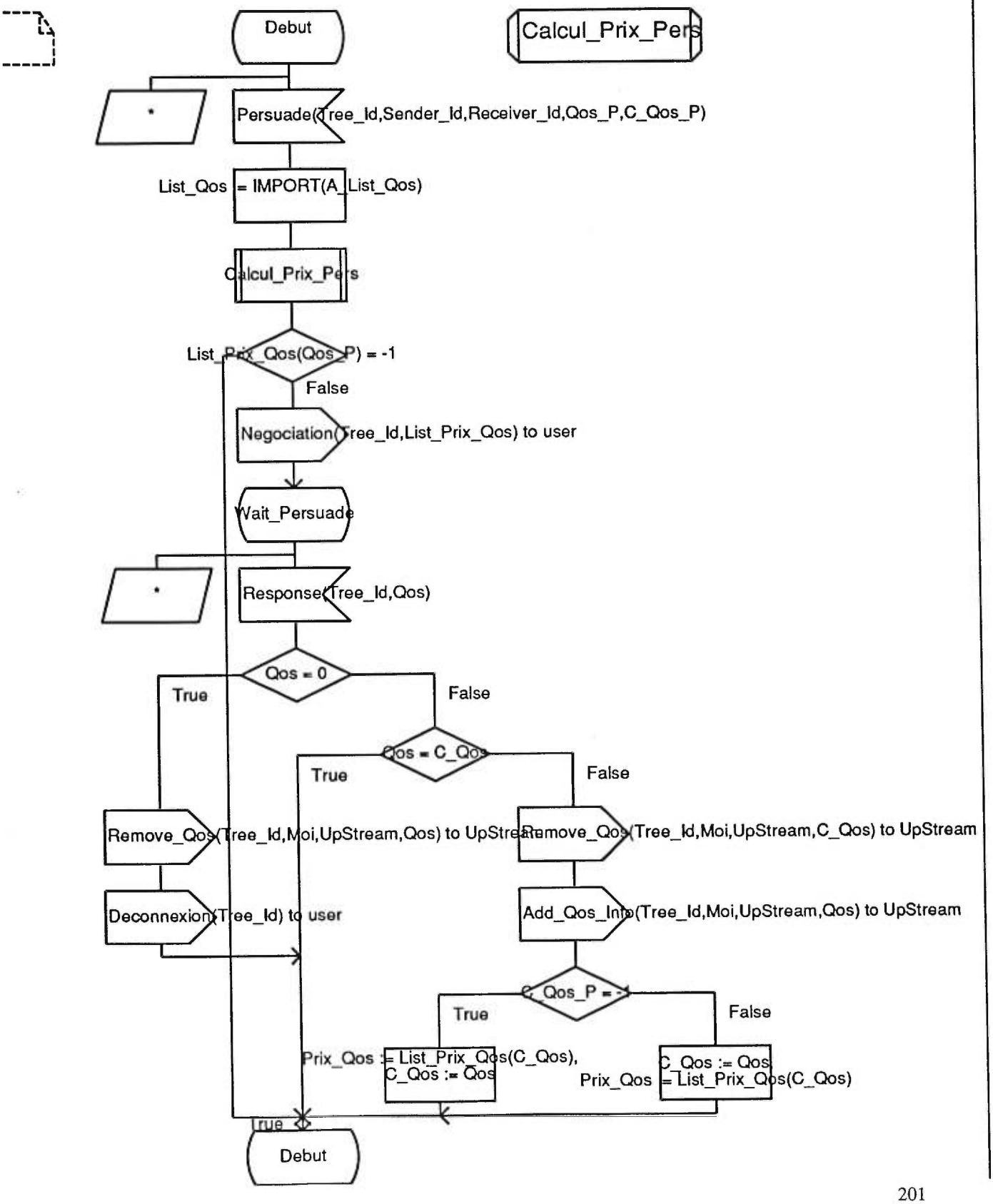
dure Chercher_Qos_R

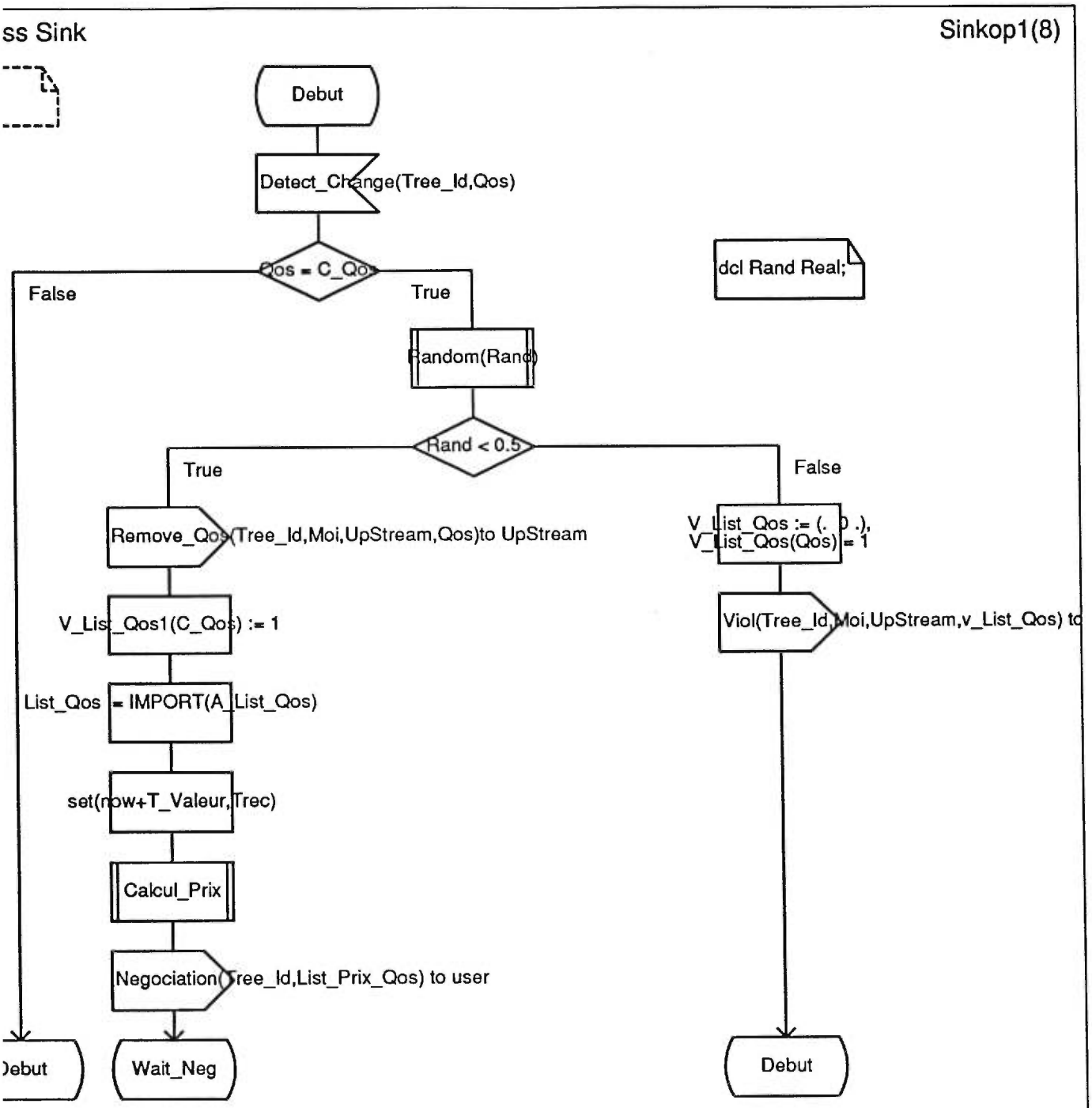
1(1)

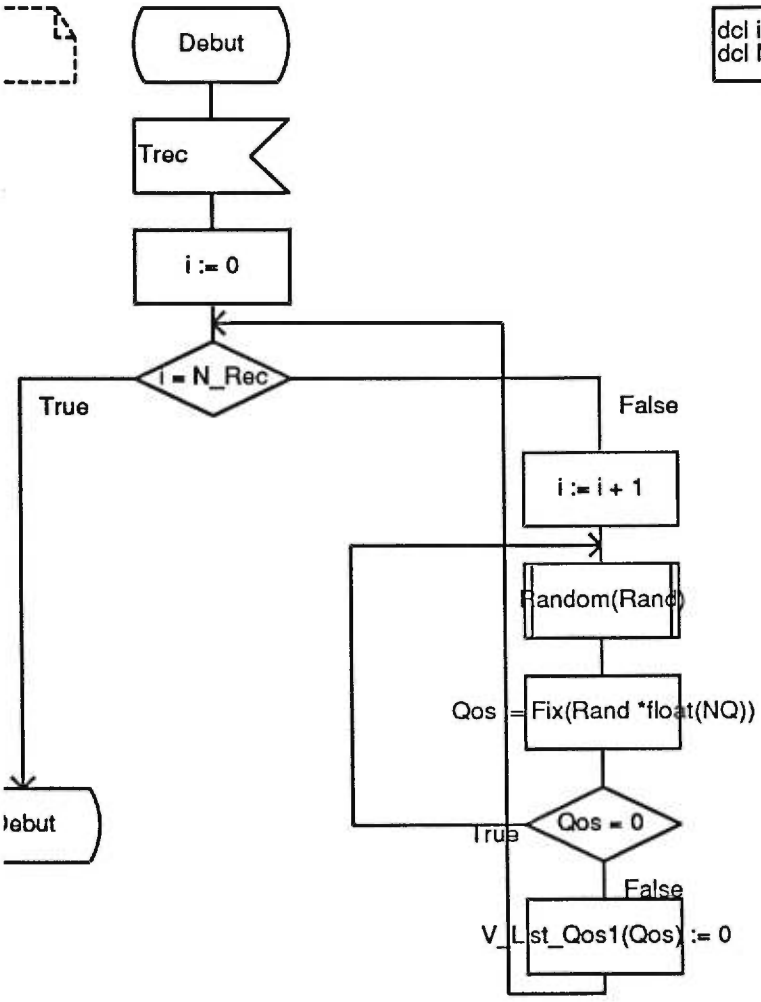


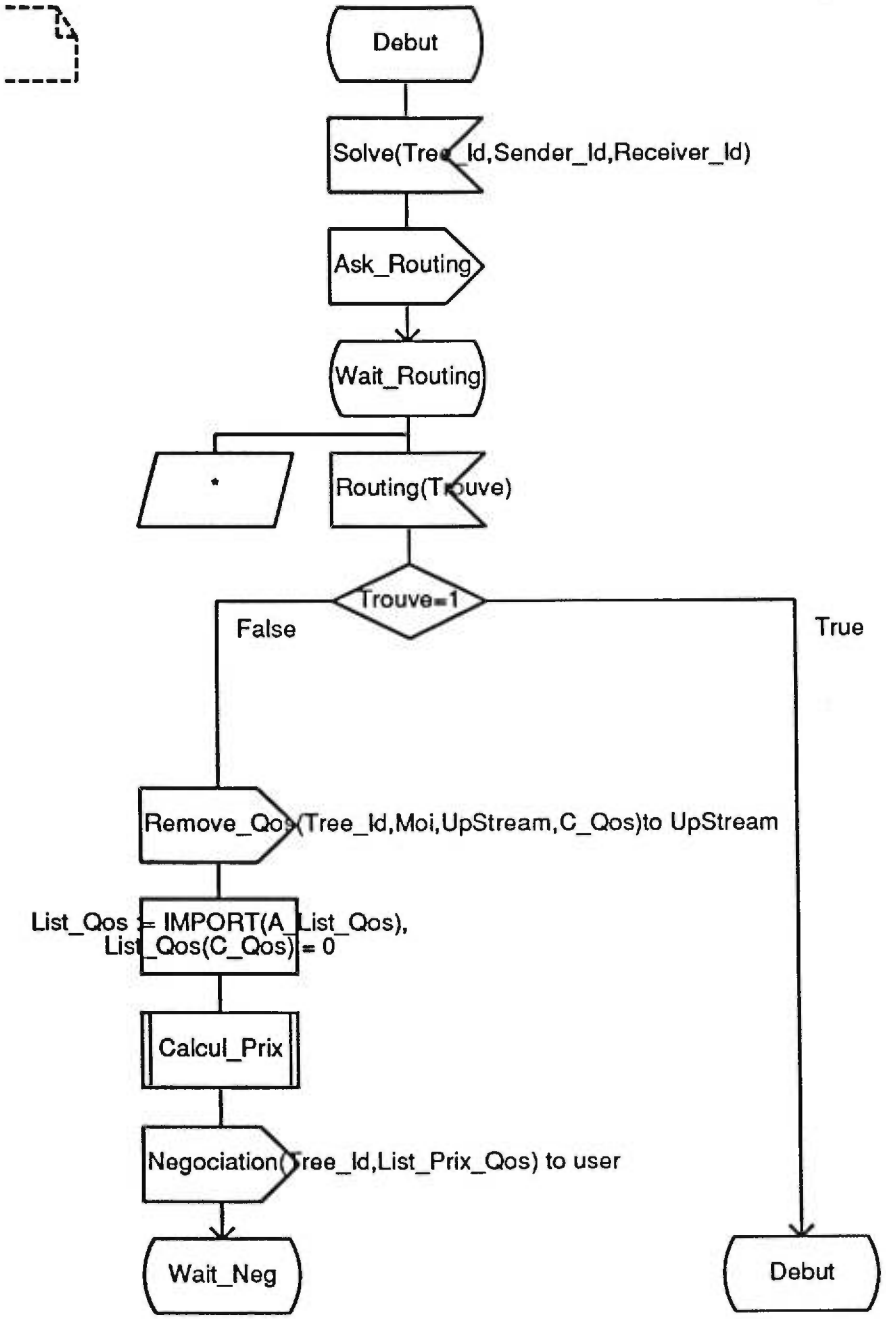


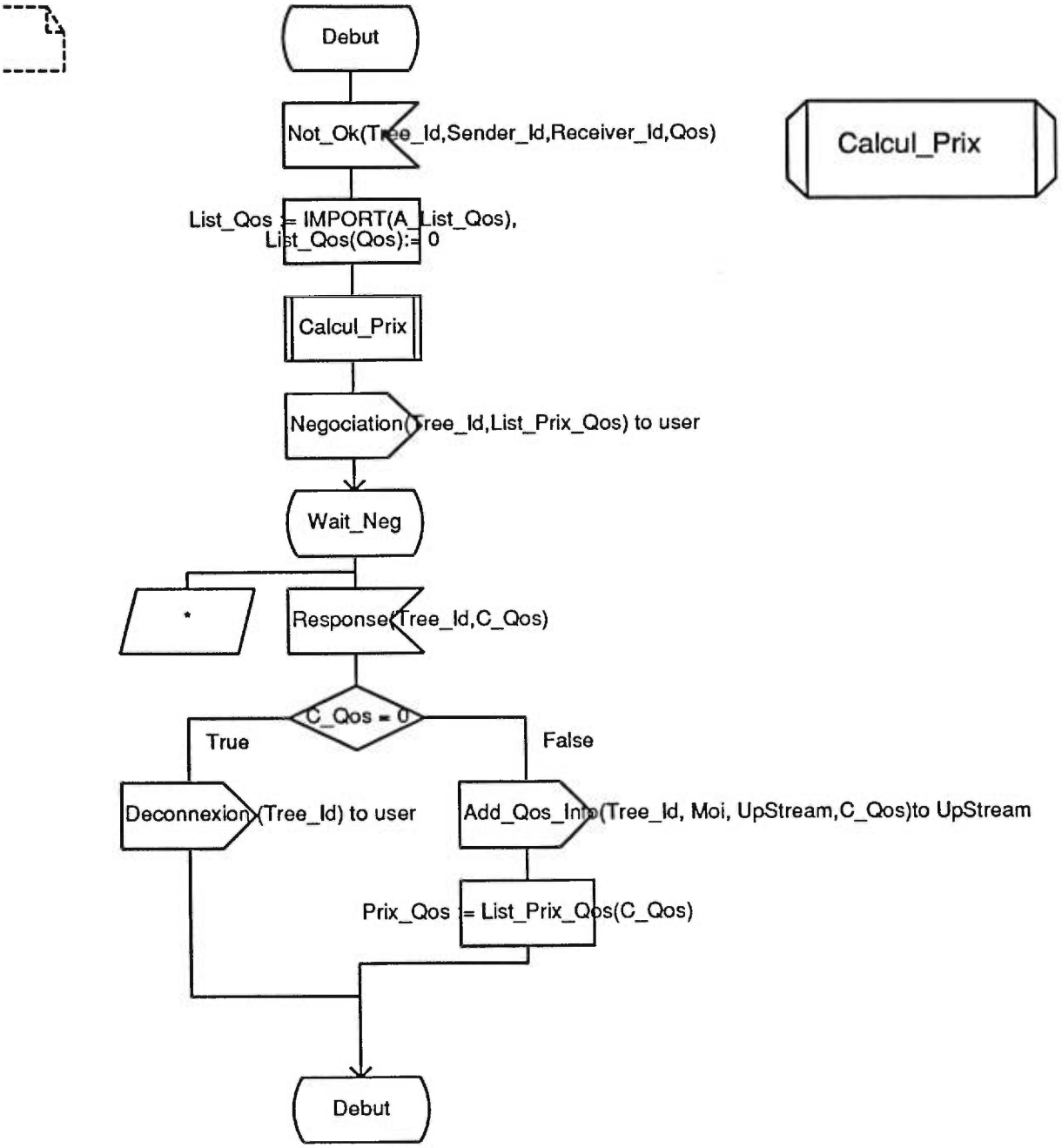

```
dcl Tree_Id integer := 1;
dcl Sender_Id,Receiver_Id, Moi, UpStream,user Pid;
dcl  trouve integer;
dcl C_Qos, Qos integer;
dcl Prix_Qos Real;
dcl V_List_Qos,V_List_Qos1, List_Qos Streams;

IMPORTED A_List_Qos Streams;
IMPORTED Cout Real_Streams;

timer Trec;
dcl T_Valeur duration := 1000;

dcl List_Prix_Qos Real_Streams;
dcl C Real_Streams;

dcl NH Real;
dcl Gain Real := 0.2;
dcl NQ  integer := 3;
dcl C_Qos_P Real := 0;
dcl Qos_P integer:= 0;
```

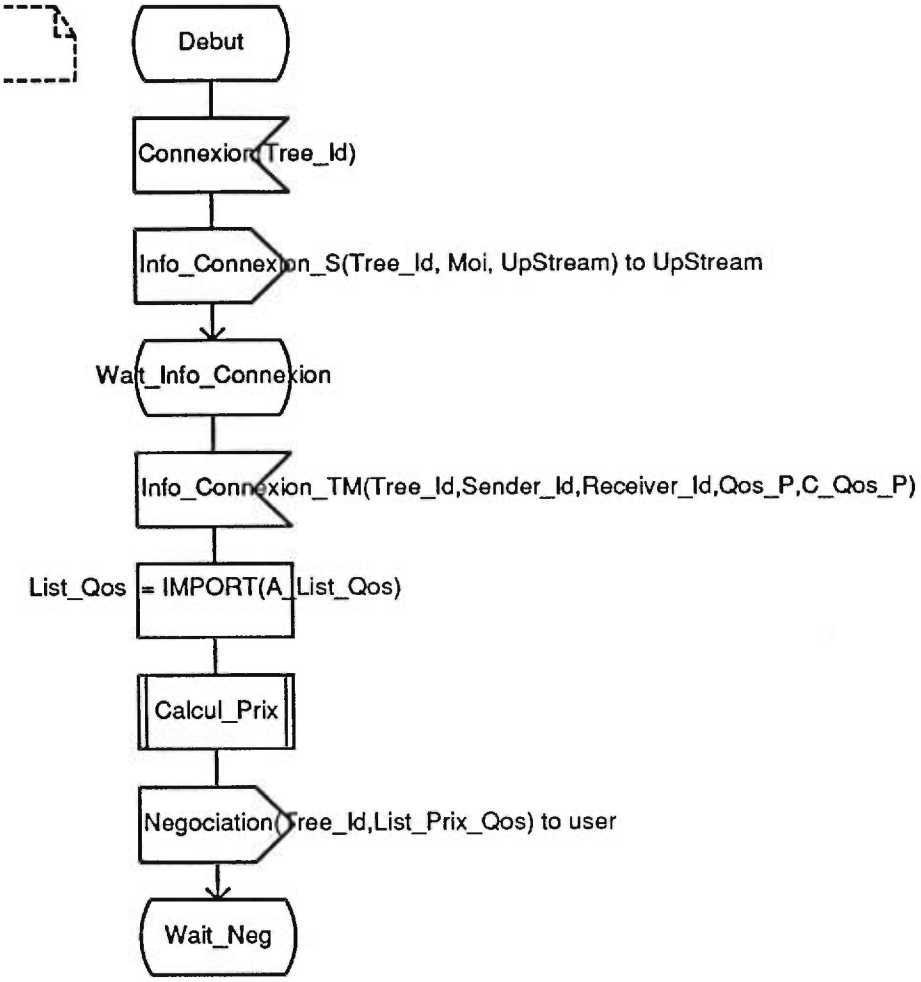

Debut
Calcul_Prix_Pers
*
Persuade(Tree_Id,Sender_Id,Receiver_Id,Qos_P,C_Qos_P)
List_Qos = IMPORT(A_List_Qos)
Calcul_Prix_Pers
List_Prix_Qos(Qos_P) = -1
False
Negociation(Tree_Id,List_Prix_Qos) to user
Wait_Persuade
*
Response(Tree_Id,Qos)
Qos = 0
True
False
Qos = C_Qos
True
False
Remove_Qos(Tree_Id,Moi,UpStream,Qos) to UpStream
Remove_Qos(Tree_Id,Moi,UpStream,C_Qos) to UpStream
Deconnexion(Tree_Id) to user
Add_Qos_Info(Tree_Id,Moi,UpStream,Qos) to UpStream
C_Qos_P = -1
True
False
Prix_Qos := List_Prix_Qos(C_Qos),
C_Qos := Qos
C_Qos := Qos
Prix_Qos := List_Prix_Qos(C_Qos)
True
Debut

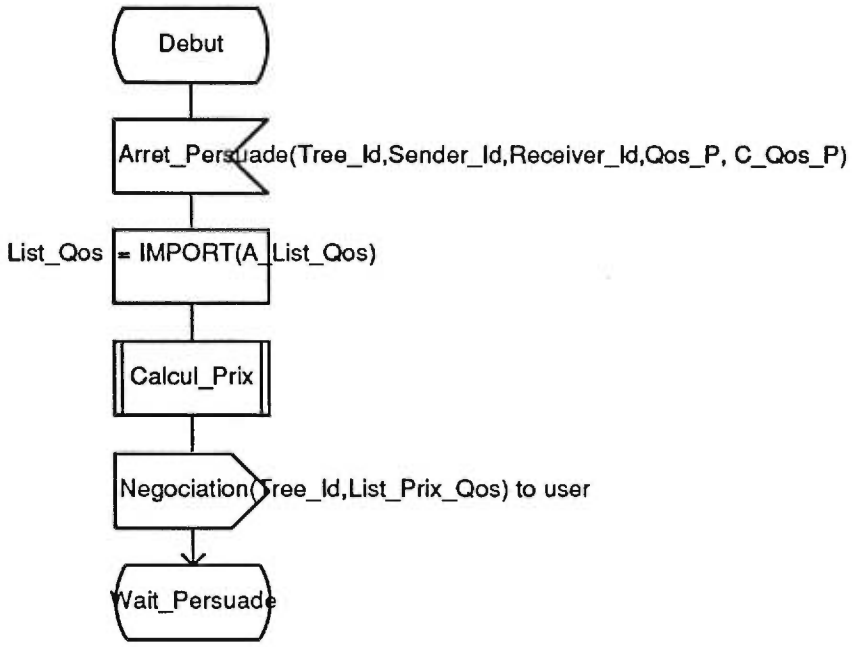

ss Sink
Sinkop1(8)
Debut
Detect_Change(Tree_Id,Qos)
Qos = C_Qos
False
True
dcl Rand Real;
Random(Rand)
Rand < 0.5
True
False
Remove_Qos(Tree_Id,Moi,UpStream,Qos)to UpStream
V_List_Qos := (. 0 .),
V_List_Qos(Qos) := 1
V_List_Qos1(C_Qos) := 1
Viol(Tree_Id,Moi,UpStream,v_List_Qos) to
List_Qos = IMPORT(A_List_Qos)
set(now+T_Valeur,Trec)
Calcul_Prix
Negociation(Tree_Id,List_Prix_Qos) to user
Debut
Wait_Neg
Debut

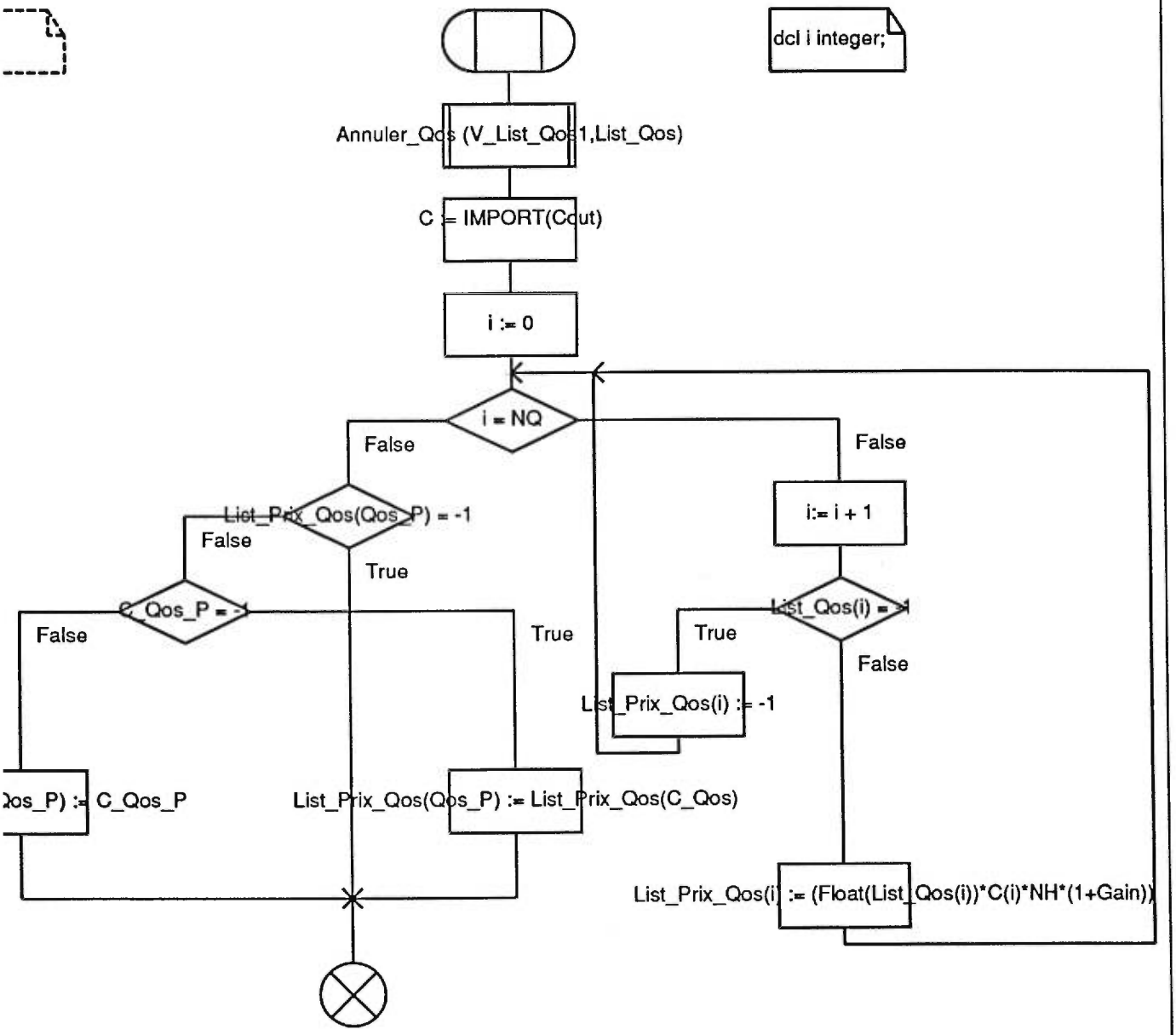

Debut
Trec
i := 0
i = N_Rec
True
False
i := i + 1
Random(Rand)
Qos := Fix(Rand *float(NQ))
Qos = 0
True
False
V_Lst_Qos1(Qos) := 0
dcl i integer;
dcl N_Rec integer := 4;

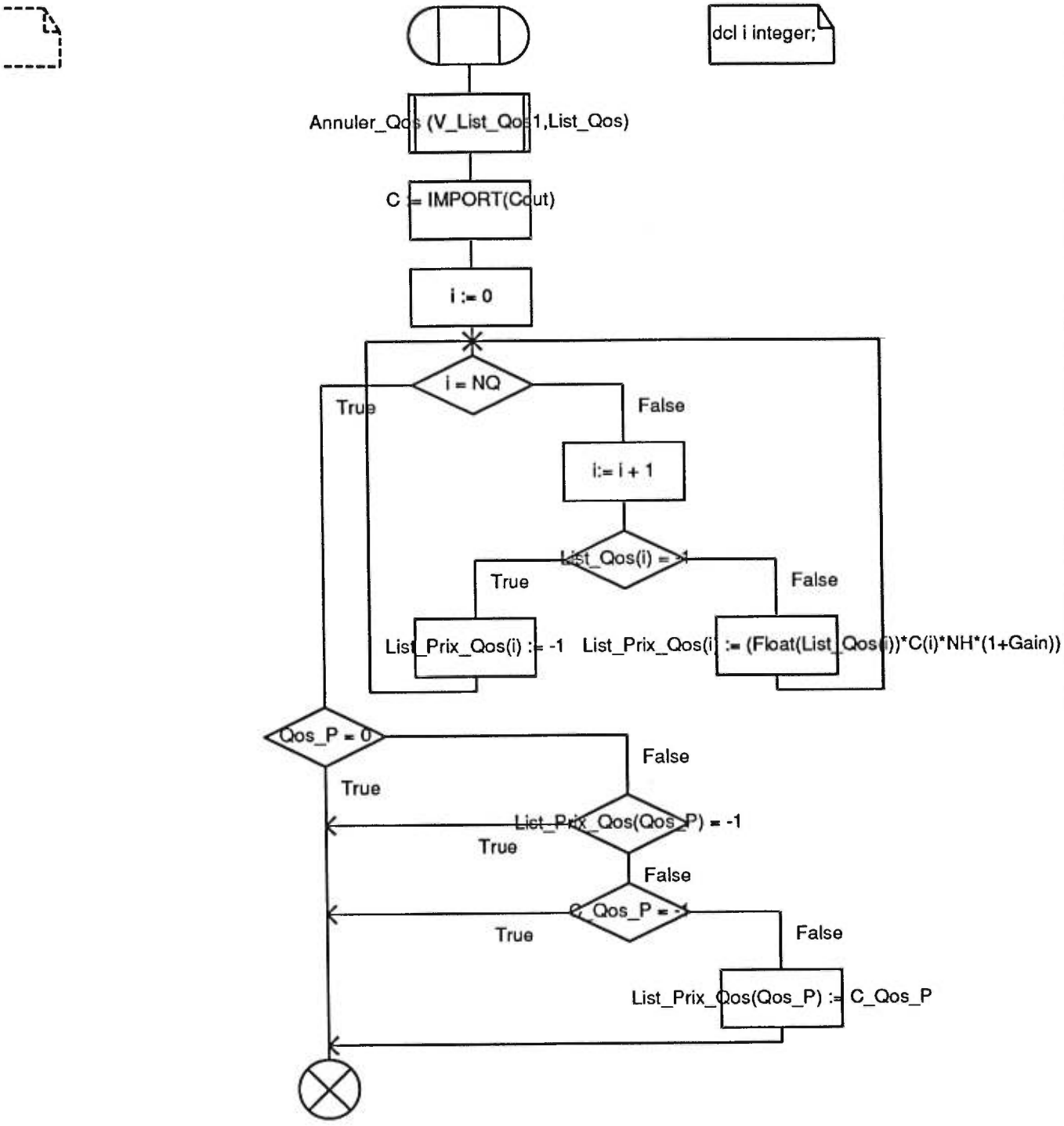

Debut
Solve(Tree_Id,Sender_Id,Receiver_Id)
Ask_Routing
Wait_Routing
*
Routing(Trouve)
Trouve=1
False
True
Remove_Qos(Tree_Id,Moi,UpStream,C_Qos)to UpStream
List_Qos := IMPORT(A_List_Qos),
List_Qos(C_Qos) = 0
Calcul_Prix
Negociation(Tree_Id,List_Prix_Qos) to user
Wait_Neg
Debut

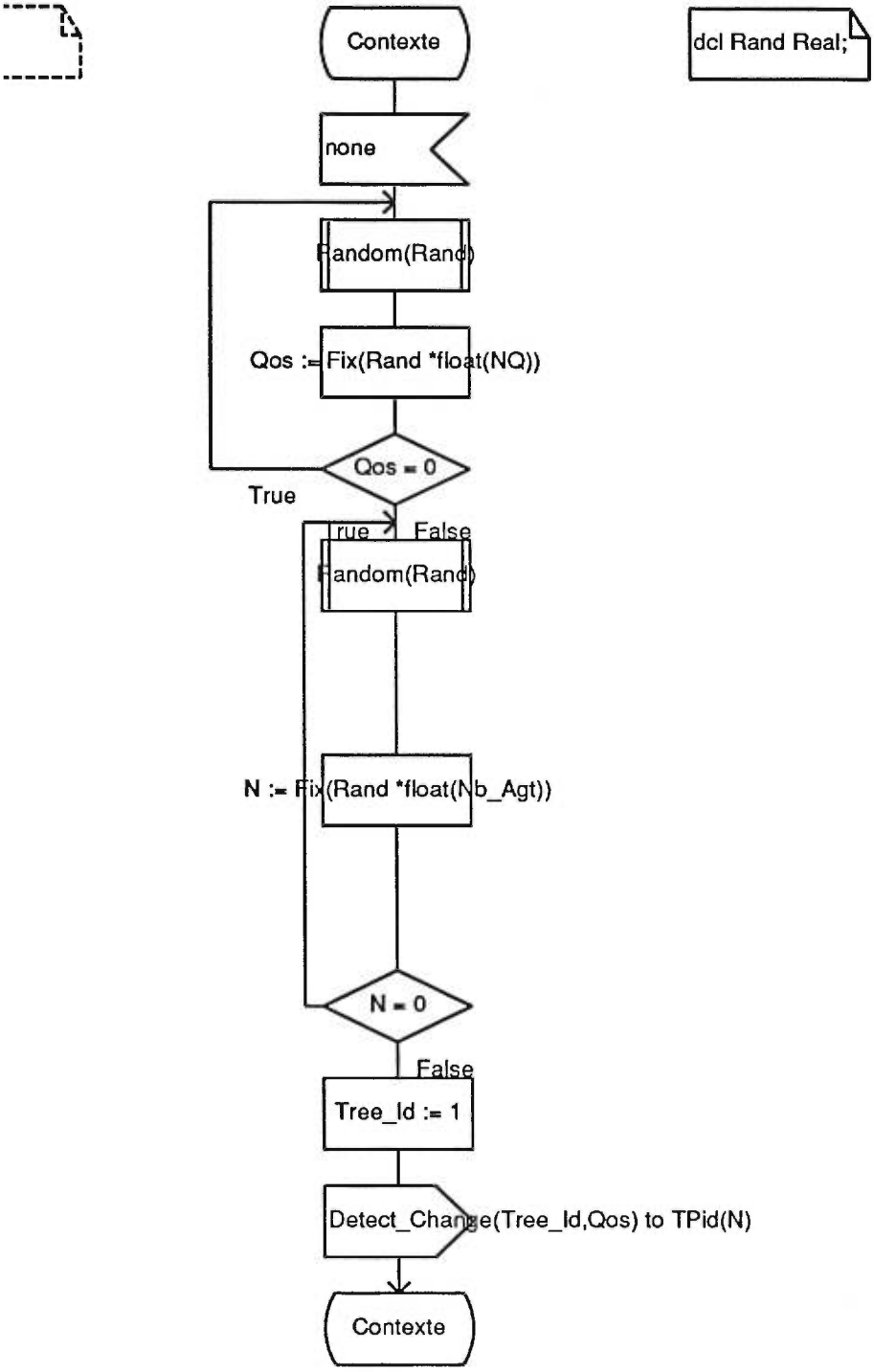

Debut
Not_Ok(Tree_Id,Sender_Id,Receiver_Id,Qos)
Calcul_Prix
List_Qos := IMPORT(A_List_Qos),
List_Qos(Qos):= 0
Calcul_Prix
Negociation(Tree_Id,List_Prix_Qos) to user
Wait_Neg
*
Response(Tree_Id,C_Qos)
C_Qos = 0
True
False
Deconnexion(Tree_Id) to user
Add_Qos_Info(Tree_Id, Moi, UpStream,C_Qos)to UpStream
Prix_Qos := List_Prix_Qos(C_Qos)
Debut

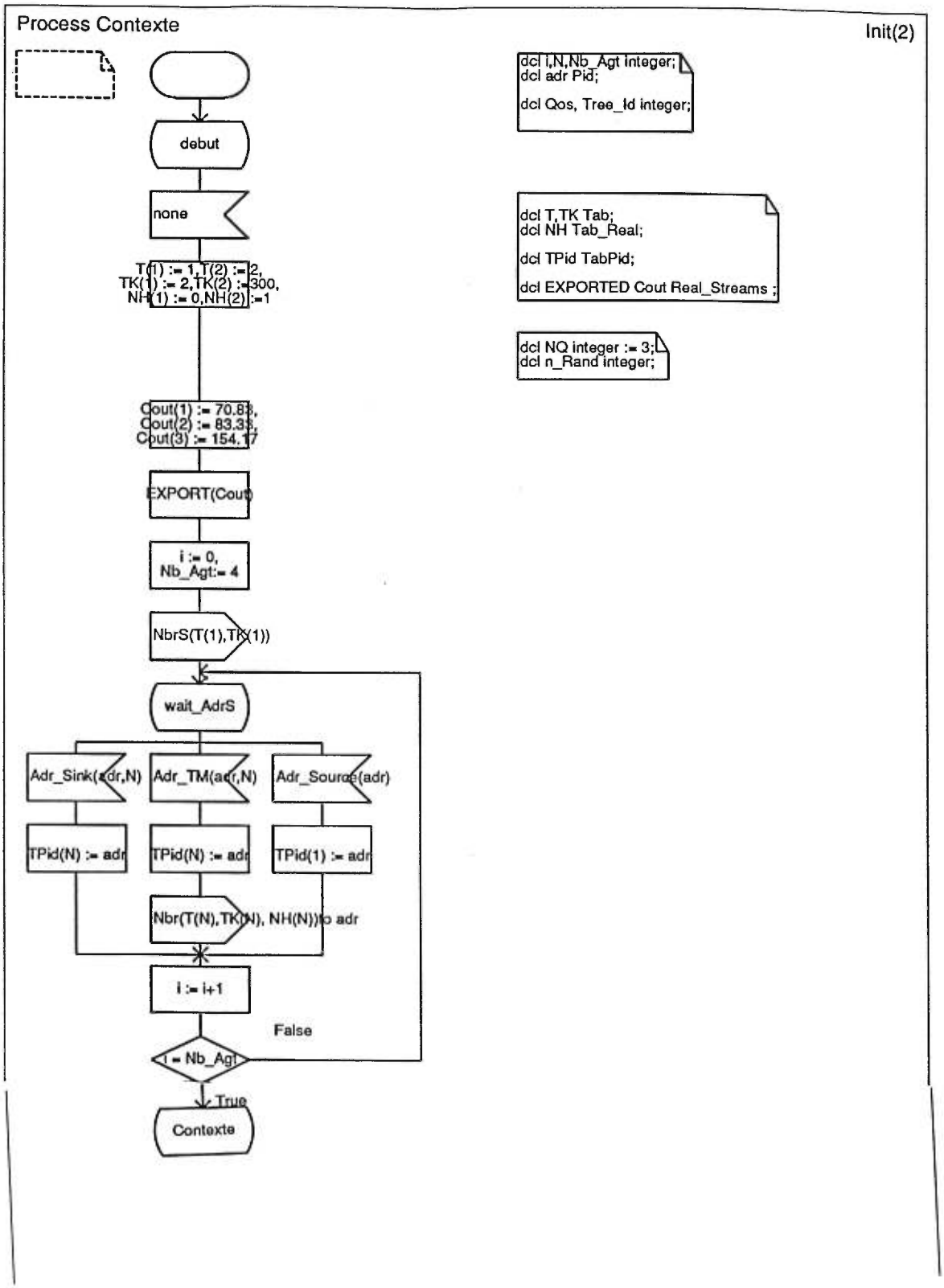

Debut
Connexion(Tree_Id)
Info_Connexion_S(Tree_Id, Moi, UpStream) to UpStream
Wait_Info_Connexion
Info_Connexion_TM(Tree_Id,Sender_Id,Receiver_Id,Qos_P,C_Qos_P)
List_Qos = IMPORT(A_List_Qos)
Calcul_Prix
Negociation(Tree_Id,List_Prix_Qos) to user
Wait_Neg

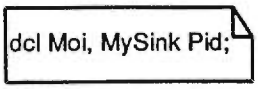

Debut
Arret_Persuade(Tree_Id,Sender_Id,Receiver_Id,Qos_P, C_Qos_P)
List_Qos = IMPORT(A_List_Qos)
Calcul_Prix
Negociation(Tree_Id,List_Prix_Qos) to user
Wait_Persuade

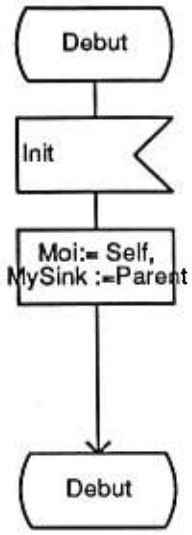

dcl i integer;
Annuler_Qos (V_List_Qos1,List_Qos)
C := IMPORT(Cout)
i := 0
i = NQ
False
False
List_Prix_Qos(Qos_P) = -1
False
True
i:= i + 1
C_Qos_P = -1
False
True
List_Qos(i) = -1
True
False
List_Prix_Qos(i) := -1
Qos_P) := C_Qos_P
List_Prix_Qos(Qos_P) := List_Prix_Qos(C_Qos)
List_Prix_Qos(i) := (Float(List_Qos(i))*C(i)*NH*(1+Gain))

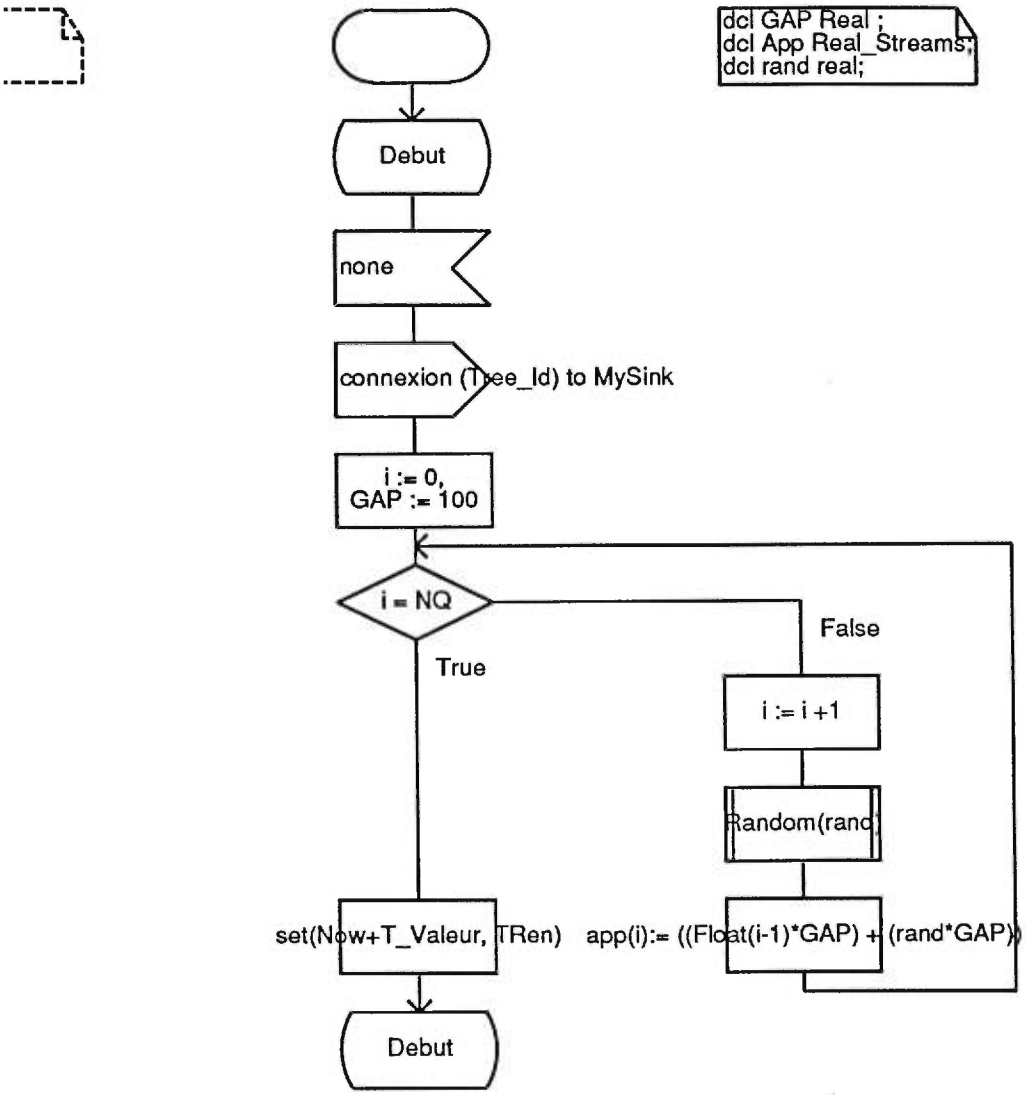

dcl i integer;
Annuler_Qos (V_List_Qos1,List_Qos)
C := IMPORT(Cout)
i := 0
i = NQ
True
False
i:= i + 1
List_Qos(i) = -1
True
False
List_Prix_Qos(i) := -1
List_Prix_Qos(i := (Float(List_Qos(i))*C(i)*NH*(1+Gain))
Qos_P = 0
False
True
List_Prix_Qos(Qos_P) = -1
True
False
Qos_P = -1
True
False
List_Prix_Qos(Qos_P) := C_Qos_P

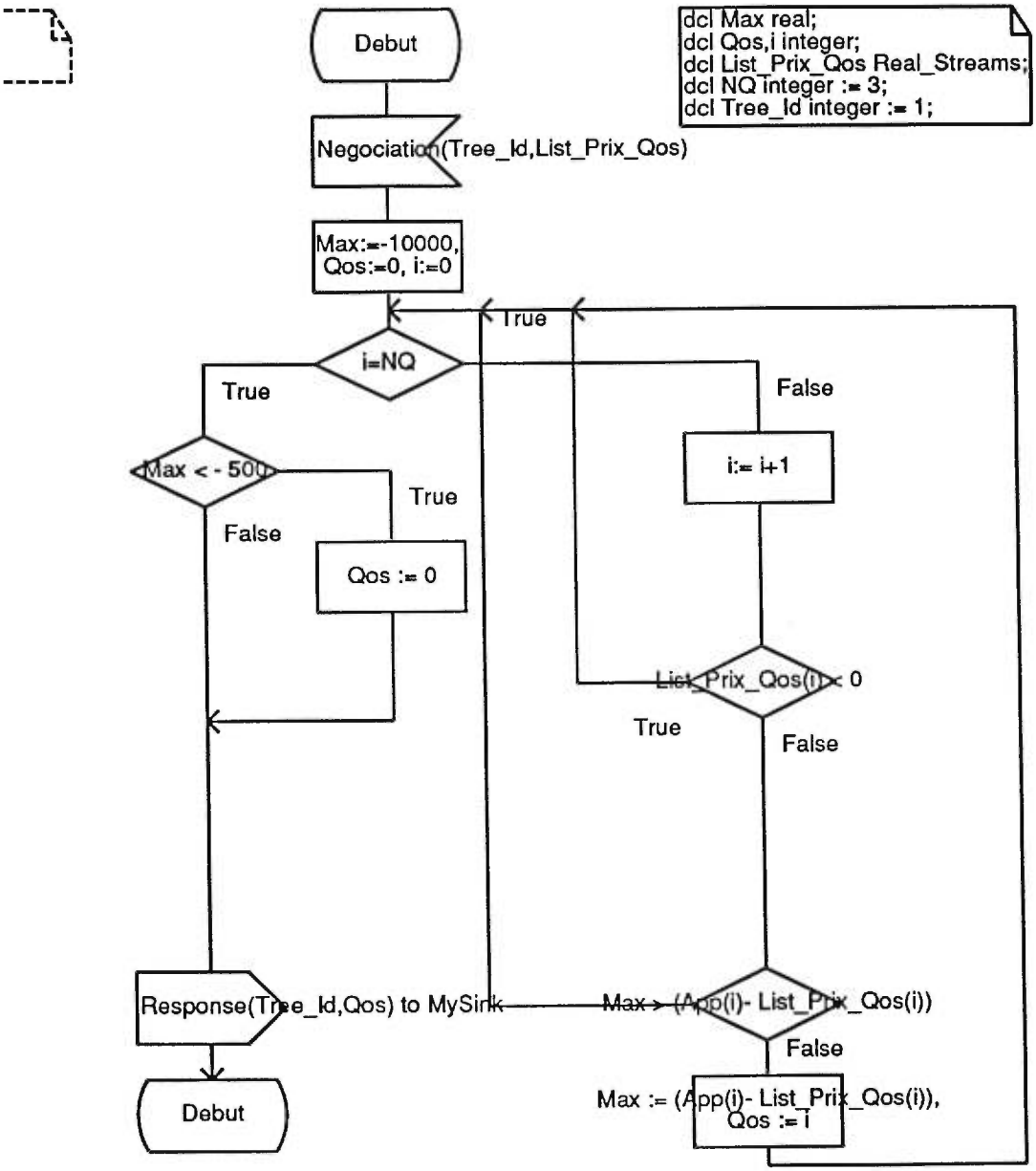

Contexte
dcl Rand Real;
none
Random(Rand)
Qos := Fix(Rand *float(NQ))
Qos = 0
True
False
Random(Rand)
N := Fix(Rand *float(Nb_Agt))
N = 0
False
Tree_Id := 1
Detect_Change(Tree_Id,Qos) to TPid(N)
Contexte


Process Contexte
Init(2)
dcl i,N,Nb_Agt integer;
dcl adr Pid;
dcl Qos, Tree_Id integer;
dcl T,TK Tab;
dcl NH Tab_Real;
dcl TPid TabPid;
dcl EXPORTED Cout Real_Streams ;
dcl NQ integer := 3;
dcl n_Rand integer;
debut
none
T(1) := 1,T(2) := 2,
TK(1) := 2,TK(2) := 300,
NH(1) := 0,NH(2) :=1
Cout(1) := 70.83,
Cout(2) := 83.33,
Cout(3) := 154.17
EXPORT(Cout)
i := 0,
Nb_Agt:= 4
NbrS(T(1),TK(1))
wait_AdrS
Adr_Sink(adr,N)
Adr_TM(adr,N)
Adr_Source(adr)
TPid(N) := adr
TPid(N) := adr
TPid(1) := adr
Nbr(T(N),TK(N), NH(N))to adr
i := i+1
False
i = Nb_Agt
True
Contexte


dcl Moi, MySink Pid;

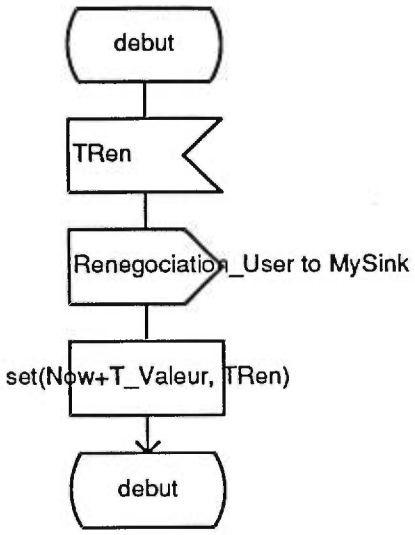

Debut
Init
Moi:= Self,
MySink :=Parent
Debut

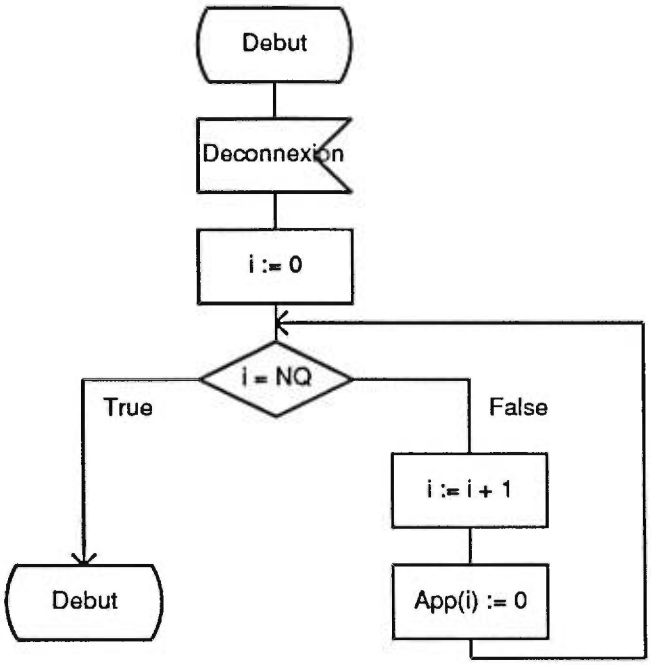

dcl GAP Real ;
dcl App Real_Streams;
dcl rand real;
Debut
none
connexion (Tree_Id) to MySink
i := 0,
GAP := 100
i = NQ
True
False
i := i +1
Random(rand)
set(Now+T_Valeur, TRen)
app(i):= ((Float(i-1)*GAP) + (rand*GAP))
Debut

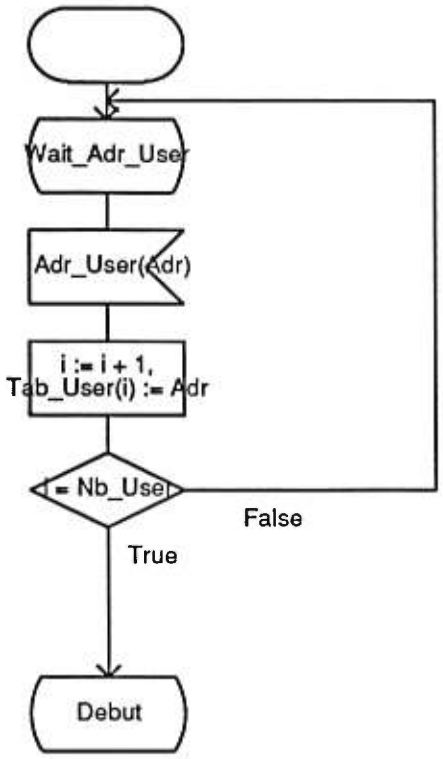

Debut
dcl Max real;
dcl Qos,i integer;
dcl List_Prix_Qos Real_Streams;
dcl NQ integer := 3;
dcl Tree_Id integer := 1;
Negociation(Tree_Id,List_Prix_Qos)
Max:=-10000,
Qos:=0, i:=0
True
i=NQ
True
False
Max < - 500
True
False
Qos := 0
i:= i+1
List_Prix_Qos(i)< 0
True
False
Response(Tree_Id,Qos) to MySink
Max > (App(i)- List_Prix_Qos(i))
False
Debut
Max := (App(i)- List_Prix_Qos(i)),
Qos := i

ss User

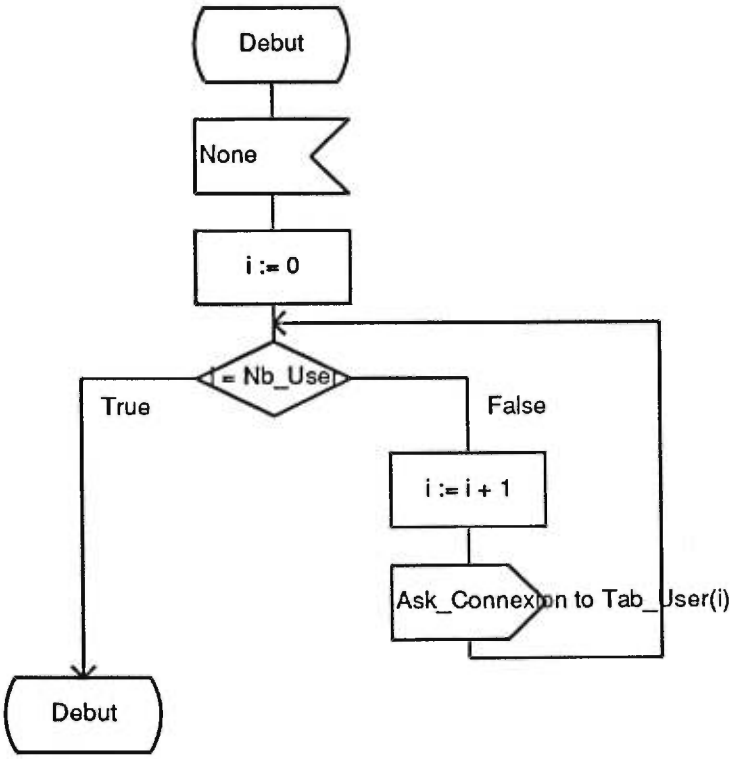
user_reneg(5)

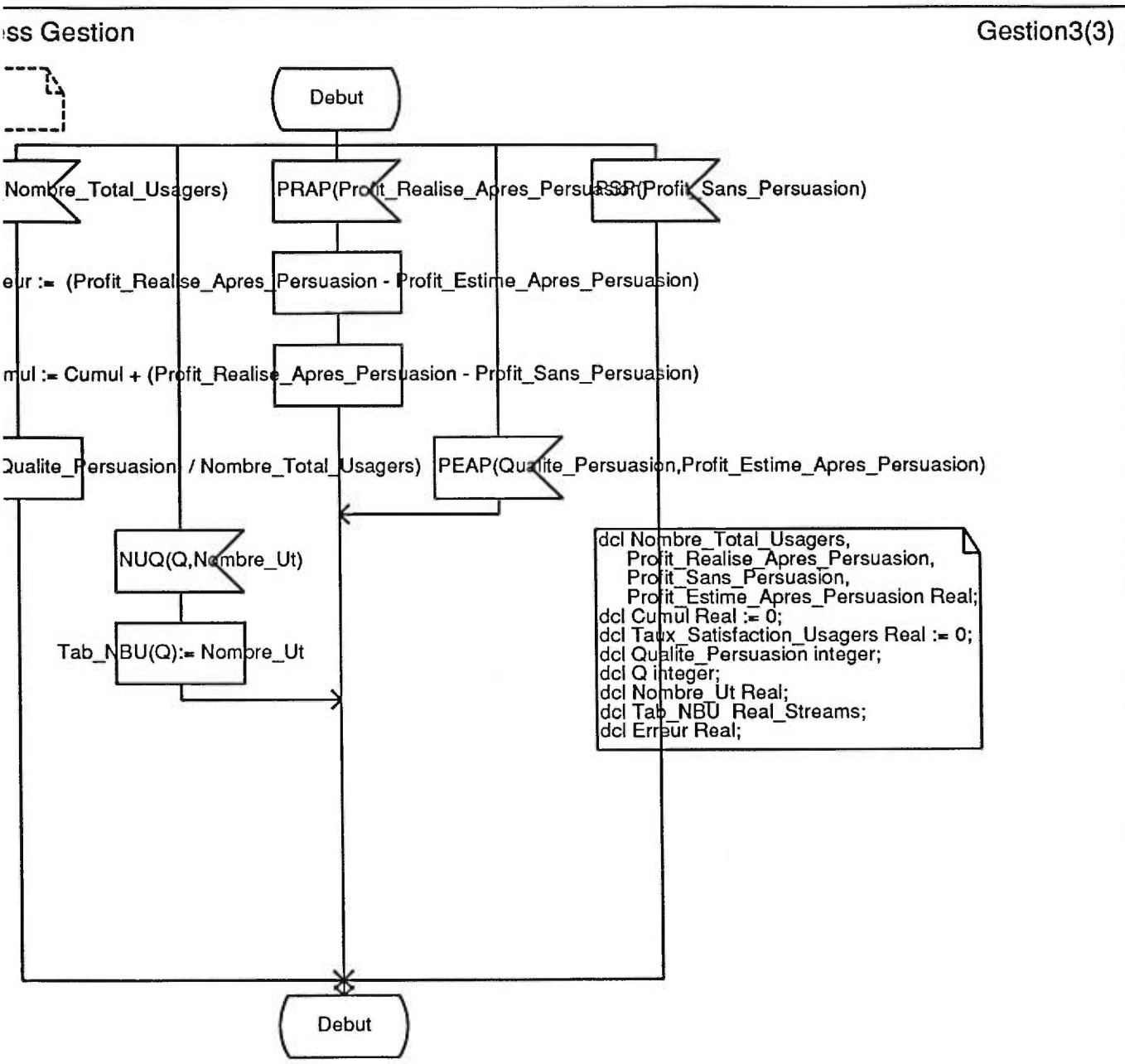


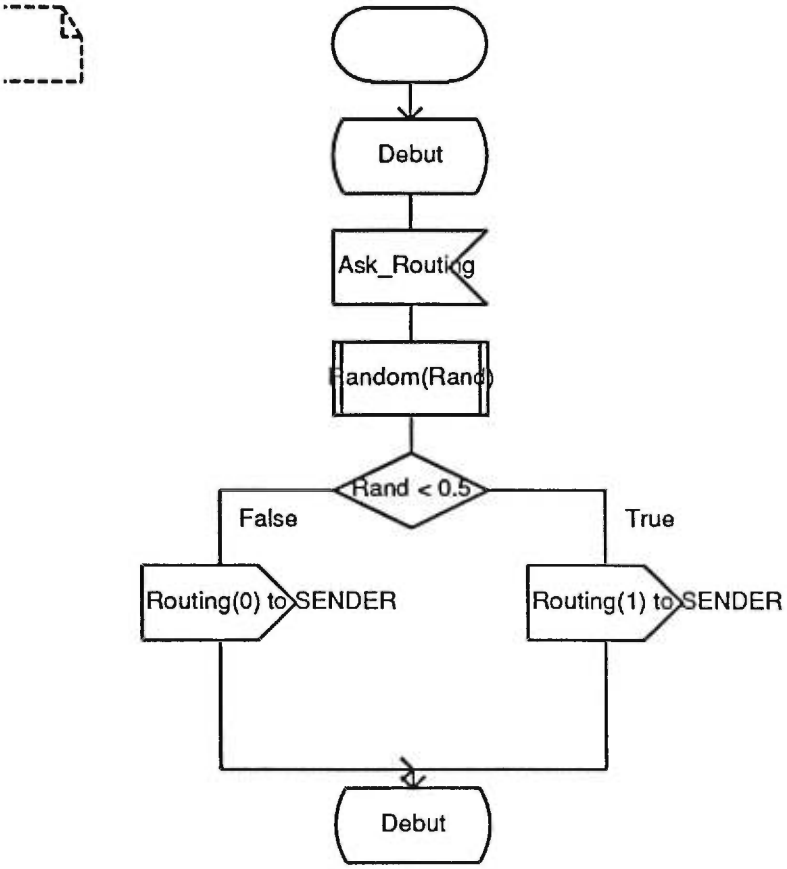

Debut
Deconnexion
i := 0
i = NQ
True
False
i := i + 1
Debut
App(i) := 0



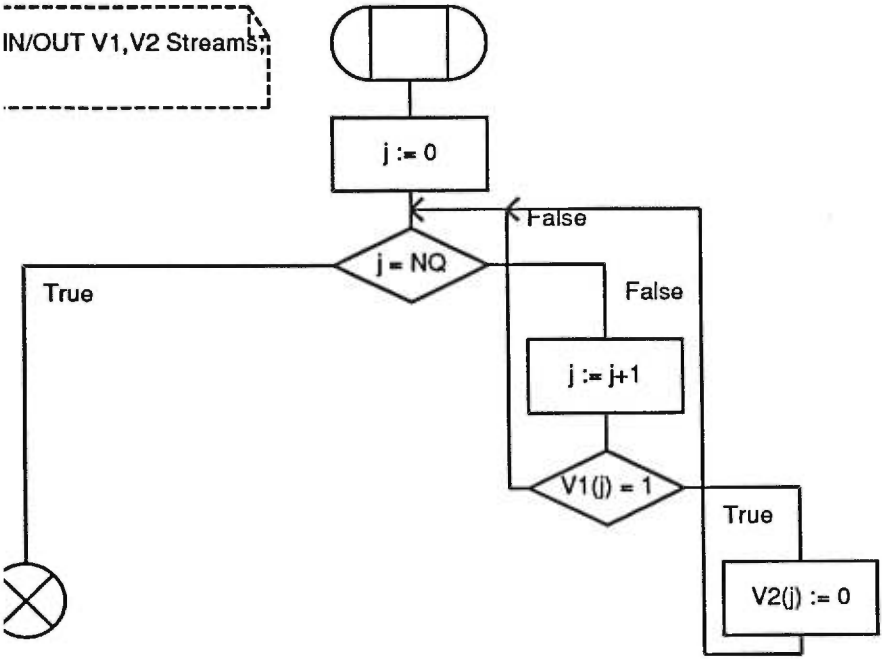

```
dcl i integer := 0;
dcl Nb_User integer := 1000;
dcl Tab_User TabPid;
dcl Adr Pid;
```


Debut
None
i := 0
= Nb_Use
True
False
i := i + 1
Ask_Connexion to Tab_User(i)
Debut

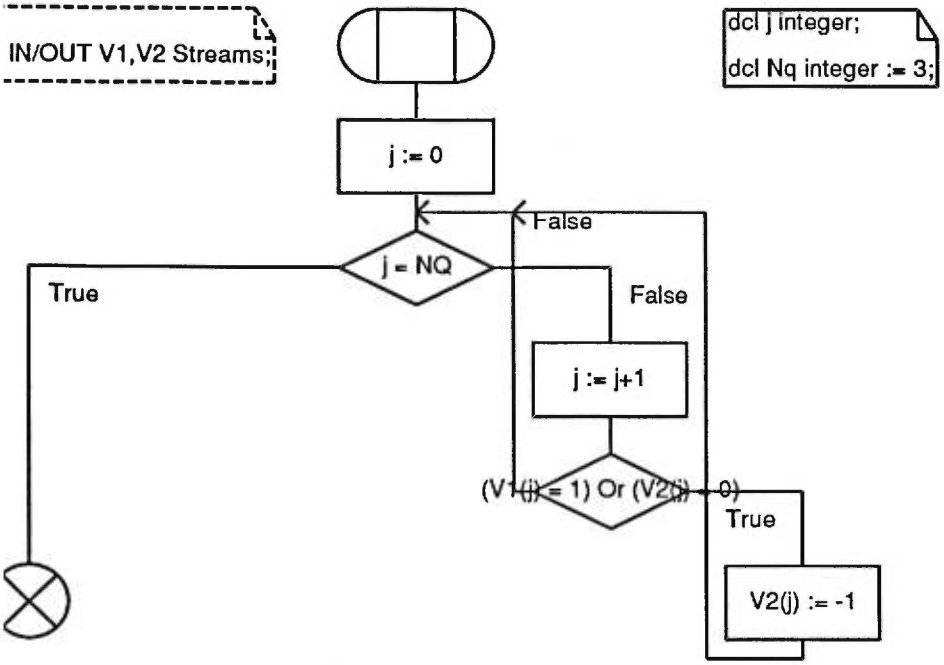

ss Gestion
Gestion3(3)
Debut
Nombre_Total_Usagers)
PRAP(Profit_Realise_Apres_Persuasion)
RSP(Profit_Sans_Persuasion)
eur := (Profit_Realise_Apres_Persuasion - Profit_Estime_Apres_Persuasion)
mul := Cumul + (Profit_Realise_Apres_Persuasion - Profit_Sans_Persuasion)
Qualite_Persuasion / Nombre_Total_Usagers)
PEAP(Qualite_Persuasion,Profit_Estime_Apres_Persuasion)
NUQ(Q,Nombre_Ut)
Tab_NBU(Q):= Nombre_Ut
dcl Nombre_Total_Usagers,
Profit_Realise_Apres_Persuasion,
Profit_Sans_Persuasion,
Profit_Estime_Apres_Persuasion Real;
dcl Cumul Real := 0;
dcl Taux_Satisfaction_Usagers Real := 0;
dcl Qualite_Persuasion integer;
dcl Q integer;
dcl Nombre_Ut Real;
dcl Tab_NBU Real_Streams;
dcl Erreur Real;
Debut


dcl Rand Real;
Debut
Ask_Routing
Random(Rand)
Rand < 0.5
False
True
Routing(0) to SENDER
Routing(1) to SENDER
Debut

ɛdure Annuler



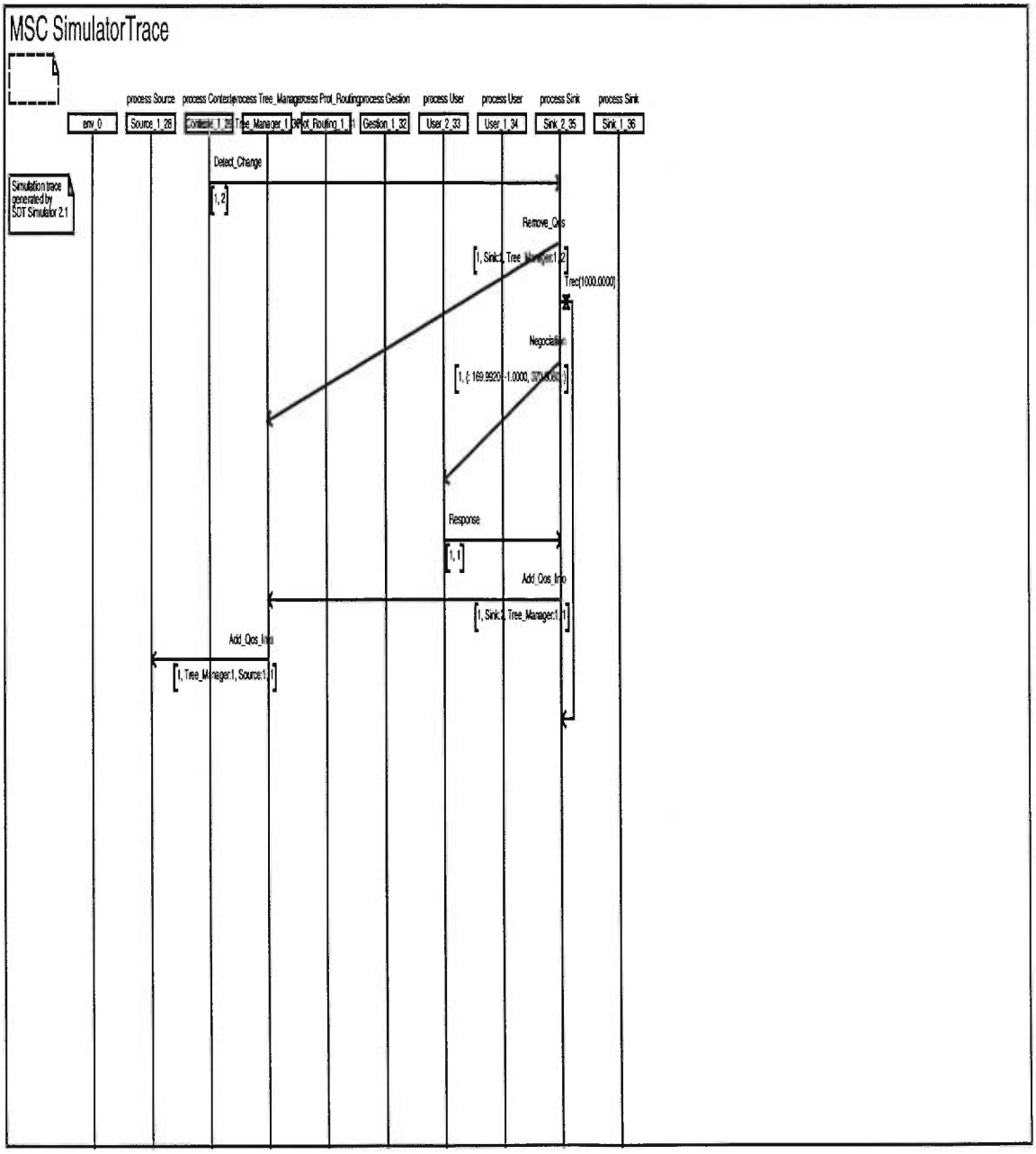
ɘdure Annuler_Qos



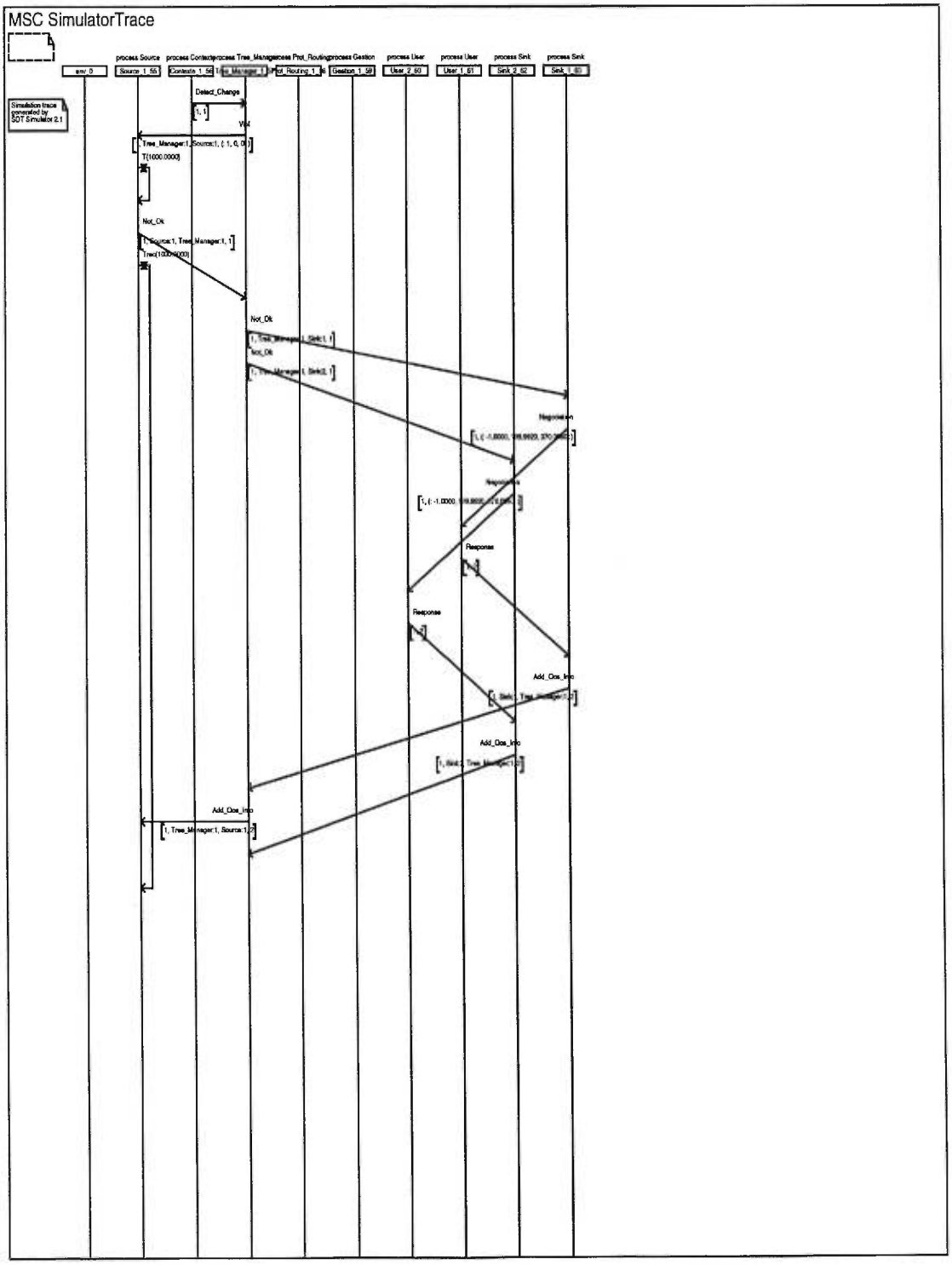
Annexe B

Résultats de la simulation du protocole de gestion coopérative de la qualité de service dans les applications multimédias

Un signal *Detect_Change* est envoyé à l'agent Sink2. Cet agent est responsable de la violation de la Qds. L'agent Sink2 procède aux changements nécessaires. Il envoie un signal *Remove_Qos* à l'agent TM de niveau supérieur. L'agent Sink2 réalise une renégociation avec l'usager2 pour que celui-ci change de qualité de service.



Un signal *Detect_Change* est envoyé à l'agent TM. Celui-ci informe les agents Sinks afin qu'ils réalisent une renégociation avec les usagers et un signal *Viol* est envoyé à l'agent Source. La récupération de la Qds est effectuée aussi au niveau de l'agent TM après une certaine période.



Un signal *Detect_Change* est envoyé à l'agent Source. Celui-ci ne fournit plus la qualité violée pendant une certaine période. L'agent TM de niveau inférieur est informé par le signal *Not_Ok*. Une opération de renégociation est effectuée avec les usagers afin qu'ils transitent vers une autre Qds. La récupération de la qualité violée est réalisée au niveau de l'agent Source après une certaine durée.