THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Université Toulouse III Paul Sabatier (UT3 Paul Sabatier)

Discipline ou spécialité :

Systèmes Informatiques

Présentée et soutenue par :
Akram HAKIRI

le : vendredi 13 juillet 2012

Titre :

Architecture de Communication à QoS Garantie pour la Simulation Distribuée

---

Ecole doctorale :
Systèmes (EDSYS)

Unité de recherche :
LAAS-CNRS; Equipe SARA

Directeur(s) de Thèse :
Pascal Berthou. Maître de conférence (Université de Toulouse III, UPS)
Thierry Gayraud . Professeur des universités (Université de Toulouse III, UPS)

Rapporteurs :

Francis LEPAGE. Professeur des Universités (Université Henri Poincaré, Nancy-France)
Pascal LORENZ. Professeur des Universités (Université de Haute-Alsace)

Membre(s) du jury :

Olivier FOURMAUX. Maître de conférence (Université Sorbonne)
Anirüddhā Gokhālé. Professeur Associé (Université Vanderbilt, USA)
Michel Diaz. Directeur de Recherche (LAAS-CNRS, Université de Toulouse III UPS)

Abstract

Stimulated by the growth of network-based applications, middleware technologies for Distributed Interactive Simulation (DIS) applications are taking an increasing importance in large scale systems, and motivate the need to achieve end-to-end Quality-of-Service (QoS) over local and large-scale networks.
The aims of this thesis revolve around network communication architecture for DIS applications in LAN and WAN. Its first contribution is to design and implement high performance Distributed Interactive Simulation (DIS) application using the HLA (High Level Architecture) and DDS (Data Distribution Service) middleware in LANs : HLA is used in conjonction with Hierarchical Timed Stream Petri Nets (HTSPN) to allow a powerful analysis techniques for validating and implementing QoS mechanisms in the application layer. Then, we show how DDS can successfully deliver the needed capabilities of DIS applications, provides fast and predictable distribution of real-time critical data in local area network.
In the second contribution we suggest a novel extension of Dead Reckoning to increase the network availability and fulfill the required QoS in large-scale DIS applications. The proposed algorithm is based on a fuzzy inference system which is trained by the learning algorithm derived from the neuronal networks and fuzzy inference theory. The proposed mechanism is based on the optimization approach to calculate the error threshold violation in networking games. Then, We show the limitations of the usage DDS Routing Services over the Internet and suggests a Proxy DDS to overcome those shortcomings.
In the last contributions we present a dynamic ressource allocation SIP-based framework for the signaling plane DDS-based DIS applications. We give the design and implementation of this framework, the new concepts of the extended SIP to improve the QoS management mechanisms. Then, we present an QoS approach using the EuQoS (End-to-End QoS over Heterogeneous Networks) architecture to define a NGN (Next Generation Network) architecture for distributed interactive simulation that builds, uses and manages end-to-end QoS across different administrative domains and heterogeneous networks.

Résumé

Les travaux décrits dans cette thèse s'articulent autour des architectures de communication en réseaux locaux et réseaux distants pour les applications de simulation distribuée interactive, particulièrement dans le cadre du projet Platsim. Nous avons traité dans un premier temps, les aspects gestion de la QoS pour les simulations distribuées basées sur les middlewares HLA et DDS en réseaux locaux, et ensuite nous avons étendu cette contribution avec DDS sur des réseaux grandes distances.
La première contribution consiste à enrichir PlatSim par un modèle formel pour la gestion de la QoS que nous avons implémentée sur HLA pour combler les manques de QoS dont souffre ce middleware. Ensuite, nous avons proposé une architecture pour l'interconnexion des simulateurs distribués avec le middleware DDS. L'utilisation de DDS est intéressante pour la simplicité de son implémentation et ses performances de communication déjà prouvées sur des systèmes complexes.
Dans la deuxième contribution, nous avons développé un algorithme de navigation à l'estime (dead-reckoning) pour l'anticipation du comportement des entités simulées. Cette approche permet d'émuler leur comportement lors de la détermination de l'erreur maximale admissible satisfaisant les contraintes de la QoS requise, ce qui, en cas de défaillance du système de communication, permet d'estimer le comportement des objets simulés. Ensuite, nous avons présenté une proposition pour l'interconnexion des simulations distribuées DDS et cette approche de dead-reckoning, par deux mécanismes différents : dans un premier temps, nous avons montré qu'il est possible d'utiliser le service de routage DDS pour mettre en place un "pont-fédéré" DDS permettant d'interconnecter des domaines DDS différents dans un même domaine IP, et ensuite nous avons proposé un "Proxy DDS" qui permet d'interconnecter des simulations DDS situées dans des domaines DDS différents et des domaines IP hétérogènes.
Enfin, nos deux dernières contributions concernent l'étude et la mise en place d'une architecture de communication à grande distance à QoS garantie pour les simulations distribuées sur DDS. Tout d'abord, nous avons présenté une architecture de signalisation de la QoS pour en se basant sur l'utilisation conjointe du protocole COPS et de la signalisation SIP. Ensuite, nous avons étendu des travaux réalisés au LAAS-CNRS dans le cadre du projet européen EuQoS. Nous avons alors utilisé des composants de cette architecture que nous avons adaptés pour fournir, à l'utilisateur final ou à l'administrateur de l'application, des interfaces simples lui permettant de demander le type de service requis pour son application sans avoir besoin de changer le protocole de signalisation.

A Mes Très Chers Parents, et toute la famille.

Remerciements

Les travaux présentés dans ce mémoire ont été effectués au Laboratoire d'Analyse et d'Architecture des Systèmes du Centre National de la Recherche Scientifique (LAAS-CNRS), dirigé successivement depuis mon entrée par Messieurs Raja CHATILA et Jean Arlat , à qui je souhaite exprimer mes remerciements pour leur accueil.
Je tiens à remercier également le groupe SARA, dirigé successivement depuis mon entrée par Messieurs François VERNADAT et Khalil Drira, pour m'avoir permis de réaliser ces travaux au sein de leur équipe.
Je tiens à saluer également ici les personnes qui, de près ou de loin, ont contribué à la concrétisation de ce travail de thèse de doctorat. Ces remerciements sont rédigés dans un moment de doux relâchement intellectuel, sans véritable rigueur ni souci taxinomique. J'ai laissé au hasard de ma mémoire, plus impressionnée par les événements récents, répétés, ou chargés d'émotions, le soin de retrouver ses personnes. Dans un autre état d'esprit, ces remerciements auraient certainement été tout autres, et j'aurais peut-être oublier un des noms qui suivent. Mais j'ai choisi ce moment précis pour les écrire.
Tout d'abord, mes remerciements s'adressent aux personnes qui m'ont proposé le sujet de thèse et qui m'ont encadré tout au long de ces années d'étude : MM. Thierry Gayraud et Pascal Berthou. Au travers de nos discussions, ils m'ont apporté une compréhension plus approfondie des divers aspects du sujet. Je salue aussi la souplesse et l'ouverture d'esprit de mes directeurs de thèse qui ont su me laisser une large marge de liberté pour mener à bien ce travail de recherche.
Je suis également très reconnaissant pour le temps accordé par l'ensemble des membres du jury de ma thèse et pour leurs remarques constructives :

- Monsieur LEPAGE Francis, Professeur à l'Université Université Henri-Poincaré (Nancy).
- Monsieur LORENZ Pascal , Professeur à l'Université Haute-Alsace (Colmar).
- Monsieur FOURMAUX Olivier, Maitre de conférence à l'Université de Sorbonne (UPMC, Paris).
- Monsieur Aniruddha Gokhale, Maitre de conférence à l'Université Vanderbilt (Nashville, USA)
- Monsieur Michel Diaz, Directeur de recherche au LAAS-CNRS (Toulouse).
- Monsieur Pascal Berthou, Maître de Conférence à l'Université Paul Sabatier (Toulouse).
- Monsieur Thierry Gayraud, Professeur à l'Université Paul Sabatier (Toulouse).

J'exprime ma profonde reconnaissance à Monsieur Michel DIAZ, qui m'a aidé tout au long de cette thèse, pour m'avoir fait profiter de son expérience, pour ses conseils avisés et les réflexions que nous avons pu mener tout au long de ces années. Je lui suis également reconnaissant pour la confiance qu'il m'a témoignée et l'apport tant sur le plan professionnel que sur le plan personnel.
Je remercie M. slim Abdellatif, qui a notablement contribué à ma compréhension des problèmes

liés à la qualité de service dans les réseaux distants et à l'analyse des données. En outre, je lui suis reconnaissant de l'accueil, de la patience, et de la disponibilité dont il a fait preuve pour discuter certains des aspects scientifiques et techniques du sujet, mais aussi, pour résoudre les difficultés que j'ai rencontrées avec l'utilisation des outils et du matériel informatique.
Les travaux développés dans cette thèse ont effectués essentiellement dans le cadre du projet PlatSim (Plateforme pour la Simulation distribuée). Je tiens à remercier l'ensemble des acteurs du projet, avec qui j'ai réellement apprécié travailler. En particulier je remercie les équipes de l'entreprise ECA-FAROS à Brest : *Philippe Printant*, *Michel Bernard*, *Gérard LE CAM*, *Nicholas Chapman* et *Alain Becam*.
Mes remerciements vont également à tous les membres du groupe OLC, permanents, doctorants et stagiaires pour leur accueil et bonne humeur. En particulier, je remercie mes collègues avec qui j'ai partagé de merveilleux moments : Lamine, Mohammed, Lyes, Baptiste, ITou, Lionel, Ahmed, Denis, Rodrigo, Jorge, Fred, Guillaume et Johan.
Mes remerciements s'adressent également à la secrétariat du groupe OLC, Gina et par la suite Sonia, pour leur disponibilité, leur gentillesse et leur aide précieuse dans les préparatifs des tâches administratives.
Je remercie également les doctorants et autres membres des différents services techniques et administratifs du LAAS-CNRS qui m'ont permis de travailler dans d'excellentes conditions. Je salue en particulier les membres de l'équipe *Sysadmin* du LAAS-CNRS.
Pour finir en beauté, je clos ces remerciements en dédiant cette thèse de doctorat à mes parents et à mes amis que j'ai eu la chance d'avoir à mes côtés, qui m'ont soutenu tout au long de ces années de travail.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Contexte

De nos jours, les techniques de simulation distribuée sont largement adoptées, tant dans le domaine militaire que dans le domaine civil, pour la modélisation et la simulation de systèmes complexes. L'évolution des technologiques conjointes a conduit à la mise en œuvre de simulateurs faisant intervenir des calculateurs répartis connectés par des réseaux informatiques, qui nécessitent des architectures de communications élaborées. Ces architectures, qui ont un rôle très important pour assurer le bon fonctionnement des applications, font appel à des mécanismes nombreux et complexes qu'il est essentiel de maîtriser, que ce soit lors de la conception de nouveaux systèmes ou de l'analyse de systèmes existants.

Plusieurs approches ont été proposées pour faciliter la distribution des données selon le domaine d'application et les fonctionnalités attendues : une première approche, basée sur le protocole de niveau applicatif DIS (Distributed Interactive Simulation) fut le premier standard qui définit le format et les règles d'échanges des informations spécifiques transmises entre applications de simulation distantes. Une deuxième approche, au niveau middleware, propose deux nouvelles architectures : la norme HLA (High Level Architecture) qui fut créée pour uniformiser ces règles et fournir des fonctionnalités génériques pour toutes les applications de simulation distribuées ; l'architecture DDS (Data Distribution Service) qui a pour but de fournir un support middleware à QoS pour la simulation distribuée. Moins général que HLA en termes d'hétérogénéité, DDS offre par contre des fonctions évoluées et précises de garantie de QoS nécessaires pour le bon fonctionnement de simulations temps réel.

Les travaux décrits dans ce mémoire se situent dans le cadre du projet PlatSim, qui a pour objet la conception et la réalisation de simulateurs génériques destinés à l'apprentissage de la conduite automobile ou motocycliste. En effet, les besoins en formation évoluent, par exemple pour l'entraînement collectif d'équipes complètes, utilisant des environnements quasi réels, afin d'être à même d'appréhender et de maîtriser des situations critiques.

Différentes problématiques sont liées à la conception d'applications de simulation distribuée. L'une d'entre elles concerne l'interconnexion de ces simulateurs, aussi bien par des réseaux locaux que par des réseaux grande distance, en particulier en utilisant des réseaux de coeur de type Internet. En effet, ces simulateurs doivent pouvoir communiquer et partager entre eux plusieurs types d'informations (informations de position, d'état, etc.), ce qui représente une

grande quantité de données avec le risque de détériorer la cohérence de la simulation du point de vue du simulateur ou du point de vue simulation. Une solution de minimisation des échanges réseaux entre les simulateurs doit être étudiée pour filtrer les données échangées sans affecter la cohérence et la consistance de la simulation.
De plus, les simulateurs doivent diffuser ces informations vers tous les participants de la simulation. Dans le cas d'une simulation basée sur DDS, la communication entre participants est liée à la notion de domaine de partage global des données qui permet de connecter des simulateurs dans un même et unique domaine DDS. Dans le cas de l'Internet, le service multicast réseau est généralement bloqué pour des raisons de sécurité et de performance, ce qui contraint la communication à grande échelle. De ce fait, l'utilisation de DDS dans le cadre de l'Internet n'est pas forcément simple. Finalement, même en présence de certains mécanismes de filtrage et de prédiction des entités simulées dans l'Internet, la communication multimédia reste toujours contrainte en l'absence de mécanisme efficace de réservation de QoS entre les participants.

1.2 Contribution et Solutions proposées

Face à cette problématique les solutions développées dans cette thèse portent sur l'analyse et la conception d'une architecture de communication à QoS garantie permet d'analyser les différents flux de simulation à manipuler par une application de simulation distribuée et d'étudier les possibilités et les avantages d'une solution architecturale intégrant les normes HLA et DDS afin d'utiliser les possibilités offertes par chacune et d'évaluer leurs performances dans des réseaux locaux.

En plus de méthodes classiques de prédiction et d'anticipation, et de hiérarchisation des flux de données, une approche de prédiction intelligente temps-réel est développée. Elle doit offrir, en cas de défaillance de l'architecture réseau dédiée ou de rupture temporaire du contrat de QoS, des performances suffisantes aux différents simulateurs et équipements qui participeront à la réalisation de l'application simulée. L'intérêt de cette solution réside surtout dans son utilisation dans le cadre de réseaux grande distance.

En tenant en compte des limites du déploiement des applications DDS utilisant le service multicast de l'Internet, nous proposons une solution capable d'interconnecter des simulations DDS coopérantes pour réaliser une unique simulation globale interopérable.

Sur la base des travaux précédents, nous proposons une architecture de communication à qualité de service garantie offrant des performances suffisantes pour interconnecter les différents simulateurs, en assurant la transmission inter-simulateurs et inter-sites des évènements présents dans cette simulation, avec un risque minimal de perte de cohérence.

1.3 Organisation de ces travaux

Pour atteindre les objectifs définis, la démarche adoptée dans ce mémoire, outre l'introduction, comporte les cinq chapitres suivant :

- Le chapitre 2 présente un état de l'art des architectures de simulation distribuée basées sur les normes DIS, HLA et DDS. Ensuite, nous faisons une synthèse des solutions existantes et des contributions d'architectures à garantie de QoS de bout en bout. Enfin, la conclusion de ce chapitre est présentée et fournit le positionnement de nos travaux.
- Dans le chapitre 3, nous développons la simulation dans le contexte local selon deux axes, à savoir : (1) une analyse d'applications distribuées basées sur le middleware HLA : nous présentons une modélisation formelle (basée sur des réseaux de Petri) et son implémentation pour gérer la synchronisation entre les différents flux issus de la simulation ; (2) une nouvelle proposition de décomposition des flux applicatifs pour le projet Platsim avec le middleware DDS : nous décrivons leurs contraintes respectives, les choix pris pour la décomposition des flux, et l'évaluation des performances de chaque solution.
- Dans le chapitre 4, nous envisageons le déploiement de la simulation sur de grandes distances. Nous proposons l'analyse et le développement de l'algorithme de Dead-Reckoning intelligent pour la minimisation du trafic envoyé par les simulateurs. Dans la deuxième partie, nous étudierons l'interconnexion de simulateurs distribuée en utilisant uniquement l'architecture basée sur le middleware DDS. Pour cela, nous proposons alors une première solution, le "Pont-Fédéré" pour connecter des simulations dans un même domaine DDS ; ensuite nous décrivons le "proxy DDS" qui permet de connecter des applications DDS dans des domaines DDS hétérogènes, avec pour but l'interconnexion de simulateurs sur un réseau grande distance.
- Le chapitre 5 est consacré à nos deux dernières contributions : nous illustrons, dans un premier temps, notre contribution qui rentre dans le cadre d'une collaboration avec l'Université Vanderbilt aux USA, concernant la proposition d'un framework pour la signalisation SIP pour la réservation de la QoS dans le contexte des réseaux distants à QoS. La deuxième partie de ce chapitre étend les travaux qui ont été faits au LAAS dans le cadre du projet EuQoS pour la signalisation de bout-en-bout inter-domaine, afin de permettre l'intégration et le test de nos travaux sur le projet PlatSim sur des réseaux multi-domaines hétérogènes.
- En conclusion générale, nous résumons les contributions majeures de nos travaux, les leçons retenues ainsi que les perspectives découlant de cette thèse.

Chapitre 2

Les applications de simulation distribuée et leurs besoins en QoS

2.1 Introduction

Ce chapitre a pour but de formaliser les communications temps-réel au sein d'une application de simulation distribuée. Il présentera un état de l'art sur les applications de simulation distribuée, et tout particulièrement ce qui concerne la spécification de leurs besoins en termes de contraintes fonctionnelles, temporelles et de qualité de service. Nous présenterons respectivement :

1. les principes de base des approches actuelles permettant particulièrement de mettre en oeuvre des simulations distribuées interactives à la section 2.2. Nous verrons notamment les concepts liées à l'architecture de communication distribuée basée sur la norme HLA à la Section 2.3 et celle sur DDS à la Section 2.4,
2. les concepts liés à la gestion de la Qualité de Service (QoS) dans les réseaux locaux et les réseaux grande distance et les mécanismes associés dans la section 2.5 : le provisionnement des ressources et le contrôle d'admission en détaillant la signalisation.
3. Une conclusion de chapitre sera donnée à la Section 2.6.

2.2 Le protocole DIS

Le défi de promouvoir le développement des applications de simulation interactive distribuée (applications DIS) a été abordé par la DARPA au début des années 1980 dans le projet SIMulator NETworking (SIMNET) [85]. Ce domaine, aujourd'hui mature mais encore en pleine évolution, recèle toujours des difficultés concernant la mise en réseau de simulateurs temps réels repartis sur des grandes distances. Les effets du réseau sur la transmission d'information entre simulateurs sont gérés et compensés par des techniques relativement basiques.
Le standard IEEE 1278 nommé aussi protocole DIS (Distributed Interactive Simulation) [62] reprend les bases de SIMNET pour permettre à plusieurs simulateurs d'interopérer en imposant la sémantique des messages échangés. Il propose des échanges de données asynchrones standardisés, où les paquets de données ou PDU (Protocol Data Units) permettent de transmettre des évènements pour être perçus par les autres entités. Les PDUs suivent un format

très précis correspondant à des évènements relevant du domaine militaire (collision, détonation, ravitaillement, etc.) [31]. Le simulateur crée des évènements, signale la création et les mises à jour d'entités vers tous les autres simulateurs.

2.2.1 Principe de conception

DIS a apporté une norme d'échange de données entre simulateurs, définissants des évènements génériques pour assurer l'interopérabilité des simulations militaires. Les informations sur les objets du monde virtuel sont censées être connues de toutes les simulations et n'ont pas besoin d'être transmises. Les entités dynamiques de la simulation changent d'états et informent les autres entités distantes de leurs déplacements et des évènements qu'elles produisent par l'émission des PDUs. Comme il s'agit d'une communication en multicast, tout simulateur diffuse et rend disponible les évènements et les objets dynamiques de la simulation à tous ceux qui participent à l'exercice. C'est au noeud récepteur de calculer l'effet de l'évènement reçu sur les entités qu'il gère, voir si ces évènements reflètent la réalité absolue, altérée ou même s'il doit ignorer ces informations.
Selon cette approche, l'émetteur ne transmet que les modifications d'états des entités qu'il gère pour maintenir un débit faible à l'émission et minimiser le traitement inutile des informations. Chaque entité réceptrice maintient un modèle simplifié de l'état des entités adjacentes, et extrapole leurs états reportés jusqu'à ce que la prochaine notification arrive. Afin de réduire le nombre de messages de mise à jour des états de ses entités, chaque site maintien, en plus de la représentation réelle, un modèle de haute fidélité (représentation extrapolée ou modèle Dead Reckoning) pour estimer les états des entités simulées localement [82]. Les états anticipés sont calculés à partir des informations d'états de leur passé en utilisant des équations d'extrapolation. L'écart entre l'état extrapolé et l'état réel ne doit pas excéder l'un des seuils définis dans le standard (Th_{pos} pour la position et Th_{or} pour l'orientation). Sur tous les autres sites distants, la réception d'un paquet contenant une nouvelle position engendre la mise à jour de l'état de l'entité concernée.

2.2.2 Communication

Les informations de simulation des états sont encodées dans des PDUs (67 différents types regroupés en 12 familles), et l'échange des données entre les hôtes utilisent le protocole de la couche de transport classique à travers UDP en mode diffusion. On retrouve un *coupleur DIS* qui joue le rôle de processeur de tri de paquets sortant ne communiquant à chaque simulateur que ceux qui lui sont adressés. A l'inverse, localement, chaque hôte envoie la totalité de ses paquets indiquant la progression de ses propres entités. En plus, pour réduire le nombre de messages transmis sur les réseaux distants, il est nécessaire d'avoir une application qui joue le rôle de pont pour bien gérer les PDUs. En effet, une quantité assez grande de la bande passante est dédiée aux entêtes, ce qui réduit les performances de la bande passante de moitié. L'agrégation de paquets se fait en fusionnant les messages en 1 seul paquet afin d'éliminer l'envoi multiple des entêtes. En outre, DIS utilise un algorithme de compression de données [61] afin de réduire le nombre de bits envoyés sur le réseau. Le principe est basé sur les observations faites sur les informations envoyées entre deux instants consécutifs, et s'il est avéré que les données transmises ne changent pas fréquemment, alors au lieu d'envoyer à chaque fois le nouvel état

de l'objet, il suffit d'envoyer les changements qui ont eu lieu par rapport à son état précédent. Avec l'émergence de la technologie ATM au début des années 90, les applications DIS utilisent ce réseau comme infrastructure de base pour les communications grande distance. Le groupe OLC au LAAS avait démontré qu'il est possible de spécifier, à priori, la qualité de service requise par ce type de simulation [25]. Chirieleison [28] propose de mettre les applications DIS sur un réseau ATM distant. Les fonctionnalités incluent un protocole de communication UDP Multicast pour la diffusion vers des groupes agrégés dans la simulation. Anshu [74] propose une évaluation de la simulation à large envergures sur un réseau ATM dédié. Il trouve que l'augmentation des tailles de paquets DIS engendre une augmentation de délai de transit de bout en bout. Ceci provient de la conversion des PDU DIS en cellules ATM. Dans [10] les auteurs étudient l'effet de l'agrégation de paquets sur le délai de transmission de bout en bout. Ils déduisent que si la taille des paquets augmente le délai de transmission augmente aussi sur le réseau. A cet effet, ils considèrent que la segmentation des PDUs en paquets de petites tailles, compatible à la taille des cellules ATM permet d'avoir de meilleurs délais de transmission et réduit énormément la gigue.
Cependant, DIS s'adresse uniquement aux simulations temps-réel, ce qui limitait sa diffusion aux simulations militaires qui obéissent à des contraintes et des formats de données bien spécifiques. Malgré son succès, DIS souffre encore de plusieurs inconvénients : il n'est pas totalement distribué, du fait que chaque message doit être reçu et traité par chaque noeud, ce qui encombre le réseau même si les données sont assez réduites. De plus, tous les PDUs sont en mode multidiffusion, ce qui ramène le contrôle de la distribution au cours de l'exercice au niveau réseau (assignation des ports). Outre, DIS ne gère pas la latence et la causalité, ce qui rend la réutilisation des simulations impossible. Ces contraintes conduisent DIS à être un protocole non flexible.
HLA et DDS ont constitué les étapes suivantes en évitant d'imposer la sémantique de DIS. Ces middlewares ont connu un essor dans les travaux de recherche qui ont conduit au déploiement de nombreuses applications distribuées. Ils reposent sur des services de communication généralement situés au niveau de la couche transport. Les applications qui implémentent ces services doivent à leur tour s'appuyer sur les propriétés prévisibles de l'infrastructure de communication pour spécifier leurs besoins qu'ils ne peuvent pas formuler explicitement par les interfaces disponibles. Pour cela, dans cette thèse, nous ne focalisons pas sur la simulation distribuée basée sur le protocole DIS à cause de ses limites, mais nous nous intéressons à la simulation distribuée basé sur HLA et DDS.

2.3 Simulation distribuée à base de HLA

La norme HLA [63] (High Level Architecture) est une architecture de simulation distribuée permettant de développer une infrastructure technique commune pour la modélisation et la simulation, favorisant l'interopérabilité et la réutilisation des composants existants. HLA permet d'exploiter de multiples programmes (simulations) pour créer une simulation à grande envergure en rajoutant la spécification de mécanismes permettant la distribution nécessaire à la communication globale. Chaque simulation participante est appelé un fédéré ; elle interagit avec d'autres fédérés au sein d'une collection de simulations, dite fédération HLA [76]. La spécification décrit

la manière avec laquelle les fédérés communiquent au sein d'une fédération à travers un middleware indispensable pour implémenter les services de HLA : le RTI (Run-Time Infrastructure), un moteur d'exécution chargé de fournir un ensemble de services aux fédérés. Chaque fédéré peut publier (produire) une variable pour envoyer les données et souscrire (consommer) à une variable pour refléter certaines informations, sans avoir aucune connaissance des autres fédérés (tout échange entre fédérés passent obligatoirement par le RTI), comme l'illustre la Figure 2.1.

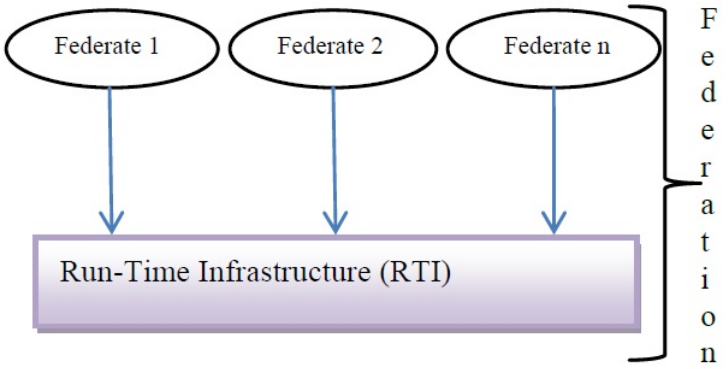


Figure 2.1: RTI et Fédérés d'une simulation HLA

Le standard HLA est constitué de 4 éléments :

- Les règles (HLA-Rules) définissent les responsabilités des fédérés et de la fédération,
- le modèle objet (HLA-OMT pour Object Management Template) fournit une standardisation de la documentation du modèle objet HLA,
- les spécifications de l'interface de programmation (HLA-IS pour Interface Specification) (API) permettent de programmer des applications de haut niveau,
- le FEDEP (Federation Development and Execution Process) offre un certain nombre de recommandations pour la conception de fédérations HLA [69].

2.3.1 Les règles HLA

Les règles [64] sont des principes et des conventions qui doivent être respectées pour permettre les interactions entre les fédérés au cours de l'exécution de la fédération. Ils expriment des concepts et des contraintes pour la conformité du standard HLA. Le standard spécifie 10 règles : 5 pour les fédérés et 5 pour la fédération.
Les cinq règles concernant la fédération sont :

1. Respect du formalisme OMT de HLA pour décrire les fédérations : les fédérations auront, obligatoirement, un modèle objet (FOM : Federation Object Model), documenté conformément à l'OMT.
2. Représentation des objets au niveau des fédérés et non de la RTI : toutes les représentations d'objets, décrites dans le FOM du modèle OMT, doivent se trouver au niveau des fédérés, et non au niveau du moteur d'exécution (RTI).
3. Passage par la RTI pour les échanges de données publiques entre les fédérés : Au cours d'une simulation (Federation Execution) au sein d'une fédération, tous les échanges de données figurant dans la FOM et agissant entre fédérés devront s'effectuer via la RTI.

4. Respect de la spécification d'interface pour l'utilisation de la RTI : pendant l'exécution d'une fédération, les fédérés doivent s'accorder pour respecter, obligatoirement, les interfaces de programmation (API) de HLA pour interagir avec le RTI.
5. Pendant la simulation, un attribut n'appartient qu'à un fédéré : pendant l'exécution d'une fédération, un attribut d'une instance d'objet ne sera la propriété que d'un seul fédéré (interdiction de l'héritage multiple).

Les cinq règles concernant les fédérés :

1. Obligation du respect du formalisme OMT de HLA pour décrire les fédérés : Comme pour la fédération, les fédérés auront, obligatoirement, un modèle objet de simulation (SOM : Simulation Object Model), documenté conformément à l'OMT.
2. Transparence des attributs contrôlés par un fédéré et capacité d'interaction : d'après leur SOM, les fédérés seront capables d'envoyer et/ou de recevoir des valeurs d'attributs d'instances d'objets ainsi que d'envoyer et/ou recevoir des interactions conformément au SOM.
3. Capacité des fédérés à transférer et/ou à accepter le contrôle d'attributs d'après leur SOM : les fédérés seront capables de transférer ou d'accepter le contrôle d'attributs dynamiquement pendant l'exécution de la fédération, conformément au modèle objet de simulation (SOM).
4. Faculté des fédérés à faire varier les conditions de mise à jour des attributs d'après leur SOM : les fédérés seront capables de faire varier les conditions (exemple : seuil) par lesquelles ils fournissent des mises à jour pour les attributs publics des objets, conformément à leur modèle objet de simulation (SOM).
5. Aptitude des fédérés à gérer un temps local, selon les principes décrits dans HLA : les fédérés seront capables de gérer un temps local de telle sorte que cela leur permette de coordonner les données échangées avec les autres membres d'une fédération.

2.3.2 HLA Object Management Template (HLA-OMT)

Le modèle HLA-OMT [68] spécifie une structure commune standardisée pour documenter le modèle objet générique de HLA (la syntaxe et le format) qui permet de définir les fédérés d'une fédération HLA. Ce formalisme décrit le support d'échange intervenant dans la création de la fédération. Toutefois, HLA n'est pas fortement typé, il ne définit pas des types de données à échanger, c'est à chaque fédéré de décrire ces propres objets. Le modèle OMT définit, d'une part, le descriptif d'une fédération fourni par le FOM (Federation Object Model), et d'autre part le comportement des fédérés dans une fédération est décrit par le SOM (Simulation Object Model) de chaque fédéré.

La Figure 2.2 présente un exemple de HLA-OMT sous forme de document XML qui nous servira pour la description du trafic applicatif que nous avons utilisé dans notre implémentation. La classe d'objet HLA dénommée *VehicleSimule* est décrite par 4 attributs nommés *VoitureVitesseArbre*, *VoitureCoeffDerapT*, *voitureFeux* et *voitureClignotant*. Chaque attribut est caractérisé, outre son nom, par un mode de transport (ici *reliable*) et une politique de gestion du temps (ici *timestamp*). A titre d'exemple, la classe d'interaction *Observateur* est décrite par 6 paramètres (*ObsTypeVue*, *ObsReculX*, *lMasqueVue*, etc.). Observons d'ores et déjà que les descripteurs des

```
<?xml version="1.0" encoding="UTF-8"?>
<objectModel DTDversion="1516.2" name="Platsim.xml"
    type="FOM" version="1.0" date="11-11-2009"
    Auhter="Hakiri Akram" sponsor="LAAS-CNRS">
    <object>
        <objectClass name="VehiculeSimule">
            <attribute name="voitureVitesseArbre"
            transportation="HLAreliable"
            order="TimeStamp" semantics=""/>
            <attribute name="voitureCoeffDerapT"
            transportation="HLAreliable"
            order="TimeStamp" semantics=""/>
            <attribute name="voitureFeux"
            transportation="HLAreliable"
            order="TimeStamp" semantics=""/>
            <attribute name="voitureClignotant"
            transportation="HLAreliable"
            order="TimeStamp" semantics=""/>
        </objectClass  >
    </object>
    <interaction>
        <!-- Ici Commence la declaration des interactions -->
         <interactionClass name ="Observateur">
            <parameter name ="obsTypeVue" datatype=""/>
            <parameter name ="obsReculX" datatype=""/>
            <parameter name ="obsReculY" datatype=""/>
            <parameter name ="obsReculZ" datatype=""/>
            <parameter name ="obsAnglesVues" datatype=""/>
            <parameter name ="obsAnglesVuesV" datatype=""/>
            <parameter name ="lMasqueVues" datatype=""/>
        </interactionClass>
    </interaction>
</objectModel>
```

Figure 2.2: Spécification du modèle HLA-OMT de PlatSim_QoS

modes de transport et des politiques de gestion du temps décrivent l'ensemble de l'interaction et non ses paramètres individuellement. Ceci constitue une première différence fondamentale entre les 2 supports de communication HLA, objets et interactions. Lorsqu'un fédéré publie ou souscrit à une classe d'objets, il peut spécifier quels attributs de la classe il désire publier ou souscrire. En revanche, un fédéré ne peut que publier ou souscrire à l'ensemble des paramètres d'une classe d'interaction.

L'autre différence fondamentale entre classes d'objets et classes d'interactions concerne la création d'instances de la classe. En effet, pour toute classe d'objets, un fédéré doit créer des instances de la classe, conformément à tout langage objet, alors que la création d'instances de classes d'interactions est impossible. La création d'instances de classes d'objets est assurée par des services de l'interface de programmation, qui devront être invoqués par le fédéré. Ces 2 différences, entre les classes d'objets et les classes d'interactions, sont justifiées par les cas d'utilisation de l'un ou de l'autre en tant que support de communication. Sur ce point, les spé-

cifications HLA ne fournissent que les recommandations suivantes : les objets sont utilisés pour échanger des informations persistantes, alors que les interactions sont utilisées pour échanger des informations fugaces.
Les modes de transport disponibles sont non fiable (best-effort) et fiable (reliable). Le mode fiable permet de garantir, pour l'attribut concerné, que chaque valeur envoyée sera reçue (typiquement il utilise le protocole TCP). En revanche, le mode non fiable, s'il est plus rapide, ne vérifie pas qu'une valeur envoyée a été reçue (typiquement il utilise le protocole UDP). Les politiques de gestion du temps disponibles sont à estampille temporelle (noté dans la Figure 2.2 timestamp) et selon l'ordre de réception (receive order). Le mode timestamp indique que la valeur de l'attribut concerné pourra être associée (sous certaines conditions) à une estampille de temps. Cette estampille de temps est utilisée dans les simulations à temps coordonné par les messages horodatés, appelés messages TSO (TimeStamp Ordered). Le mode receive-order indique qu'il sera impossible d'associer une estampille de temps à la valeur de l'attribut concerné. L'absence d'estampilles de temps pour les valeurs d'attributs correspond aux messages non horodatés, les messages RO (Receive Order) sont utilisés dans les simulations événementielles.

Federation Object Model (FOM) :

le FOM d'une fédération permet de spécifier un contrat d'échange de toutes les informations pouvant être échangées au cours de l'exécution d'une fédération. On décrit donc l'ensemble des classes d'objet avec leurs attributs, ainsi que l'ensemble des classes d'interactions, avec leurs paramètres pouvant être échangés durant l'exécution de la fédération. L'objectif est de fournir une spécification d'échange de données publiques entre fédérés, passant par :

- une énumération de toutes les classes d'objets participant à une fédération,
- une description d'attributs de ces classes d'objets,
- une description des interactions entre classes,
- une spécification des paramètres des interactions.

Simulation Object Model (SOM) :

Le SOM d'un fédéré décrit les classes d'objets et les classes d'interactions qu'un fédéré pourra publier et/ou souscrire, c'est-à-dire les données qu'un fédéré va produire pour le reste de la fédération et celles qu'il va consommer. C'est une spécification complète d'un fédéré qui pourrait fournir de la matière à une fédération en déclarant les objets, leurs attributs, leurs associations et leurs interactions. La construction d'un FOM s'appuie sur l'intégration des parties des SOMs des fédérés participant à la fédération. En effet, écrire un SOM ou un FOM consiste à renseigner les composants de l'OMT, qui sont un ensemble de tables, les principales étant la table d'identification et les tables liées aux données échangées.
HLA impose que les composants (classes d'objets et leurs attributs, classes d'interactions et leurs paramètres) soient documentés sous forme des tableaux permettant d'implémenter l'application de simulation. Ces tableaux sont organisés de la manière suivante :

- le tableau d'identification modèle objet définit les caractéristiques d'une application de simulation sous HLA,
- le tableau structure classes d'objets décrit les classes d'objets, leurs sous-classes dans une simulation ou une fédération,

- le tableau structure classes d'interactions décrit les classes d'interactions, leurs sous-classes dans une simulation ou une fédération,
- le tableau attributs spécifie les caractéristiques des attributs d'objets dans une simulation ou une fédération,
- le tableau paramètres spécifie les caractéristiques des paramètres d'interactions dans une simulation ou une fédération,
- le tableau de l'espace de routage spécifie une zone d'échange protégée pour les attributs d'objets et d'interactions dans une fédération.

Nous décrivons en détails dans le prochain chapitre, les tables d'attributs et des interactions qui serviront à la caractérisation du trafic de simulation.

2.3.3 HLA-RTI

Rappelons que c'est RTI (Run-Time Infrastructure), implémentation des spécifications de l'interface de programmation [67], qui est responsable de la gestion de la communication, en offrant les services de HLA aux fédérés. Le modèle objets de HLA ne concerne que la communication et les échanges de données entre fédérés au sein d'une fédération. Ces spécifications, illustrées à la Figure 2.3, définissent les services et les callbacks permettant à un ensemble de fédérés de dialoguer par échange de données, au sein d'une même fédération. Il est important de comprendre que le standard HLA ne fournit que les spécifications de ces services et en aucun cas des règles d'implémentation dans un langage de programmation donné.
L'implémentation des spécifications HLA est assurée par RTI, qui constitue un ensemble de processus informatiques capables d'offrir les services définis par les spécifications à un ensemble de fédérés participant à une même fédération. Ces services sont classés en 6 familles fondamentales :

1. **Gestion de la fédération :** traite la création, l'initialisation, le contrôle dynamique, la modification et la suppression des fédérations en se basant sur les documents de données des FOM (FDD), la façon avec laquelle les fédérés peuvent rejoindre ou quitter la fédération, la sauvegarde et la restauration des états au cours de l'exécution de l'exercice, la gestion des points de synchronisation et la destruction de la fédération.
2. **Gestion de déclarations :** regroupe les services et callbacks permettant aux fédérés de déclarer au RTI leurs intentions de publier ou de recevoir les informations des interactions et des états et des attributs des objets.
3. **Gestion des objets :** Ce service génère la plupart du trafic réseau au cours de la simulation. Il permet à un fédéré d'enregistrer et de créer les instances des objets d'une classe d'objet qu'il publie et d'être informé de la création d'une instance de cette classe par un autre fédéré. En plus, elle permet aux fédérés publiant une classe d'objet de mettre à jour, supprimer et de notifier les autres fédérés de ces opérations.
4. **Gestion de temps :** fournit les services permettant la configuration, la synchronisation et la modification des horloges de simulation. Chaque fédéré maintient deux horloges locales : une horloge physique utilisée pour la synchronisation entre les individus et les entités réelles et une horloge logique (temps simulé) pour assure la délivrance des messages et des événements dans le bon ordre.

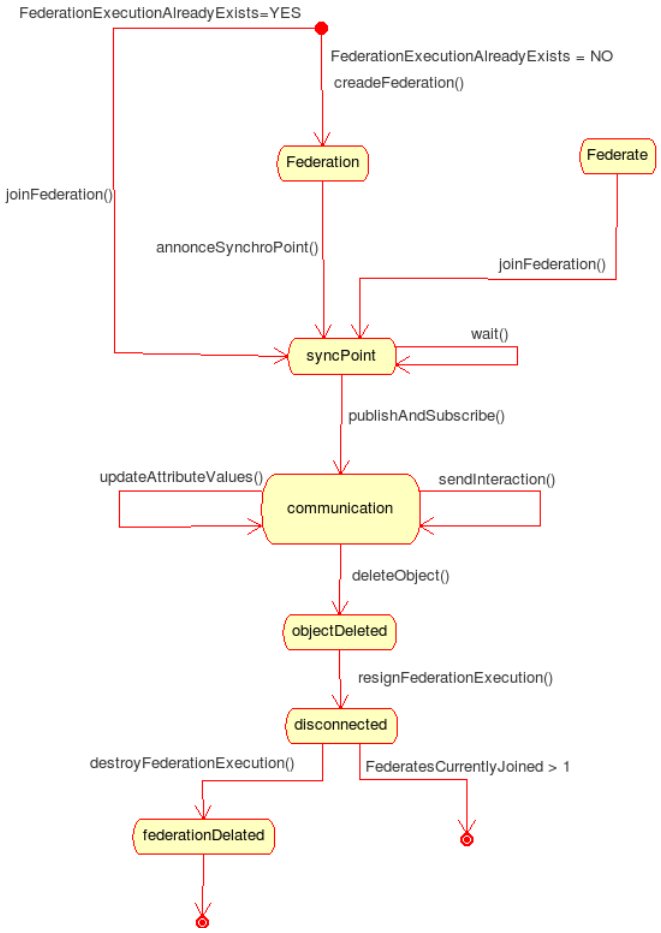


Figure 2.3: Diagramme d'état d'une simulation distribuée HLA

5. **Gestion de propriété :** ces services permettent aux fédérés d'acquérir ou de céder la propriété des attributs des objets aux autres participants de la simulation. Ceci facilite la création d'un modèle coopératif d'une entité à travers de multiples hôtes et fournit les moyens de migration des objets d'un hôte vers un autre. Un fédéré se dépossède des propriétés sur l'objet vers RTI qui les assigne à un autre fédéré intéressé par ces propriétés ou bien à un des fédérés qui peut se coordonner avec le premier pour échanger ces données avec lui.

6. **Gestion de distribution des données :** Ce service facilite le transfert et les mises à jour entre les fédérés en utilisant le modèle producteur-consommateur. Ce modèle étend les mécanismes utilisés dans le prédécesseur de HLA (DIS, ALSP [46]) pour les communications en multicast en fournissant un service de livraison à la demande. Les fédérés peuvent imposer des conditions strictes pour gouverner la distribution de leurs propres données ou pour interdire l'accès à certaines informations.

2.3.4 Fonctionnement de HLA sur les réseaux distants

L'enjeu pour permettre une simulation à grande envergure est de respecter les limites de la bande passante, du délai et de la variation du délai. Chen [26] propose une méthode de groupement des simulations HLA sous une forme de pont pour interconnecter les fédérations. Se basant sur l'avantage que présentent les *ambassadeurs des fédérés* pour gérer la communication avec *ambassadeur des RTI*, cette approche permet d'interconnecter les RTIs de différentes fédérations distantes pour assurer l'interopérabilité et la scalabilité des simulations.
Petty et Wood [134] ont proposé une passerelle externe pour supporter l'interopérabilité entre des fédérations HLA et DIS. Cette approche a l'avantage de ne pas nécessiter des changements sur les simulations existantes et de permettre à deux protocoles différents de communiquer entre eux. La passerelle convertit les PDUs DIS à des appels de services au niveau du RTI en utilisant les données définies dans le FOM.
Dambrogio [6] décrit une approche combinant HLA et CORBA pour des applications distribuées sur le web. Le composant central de la communication (CRC) ou se trouve le RTI est installé sur le serveur et les clients CORBA sont considérés comme des simulations. Les communications entre les fédérés sont gérés par RTI via un proxy CORBA. L'inconvénient de cette solution réside dans le fait que les requêtes et les réponses au sein de la simulation doivent passer par le bus CORBA. Cette approche ajoute des latences à la communication et exclut les communications en multicast. Afin de remédier à cet inconvénient, l'architecture est orientée vers les services Web distribués. Le management des simulations se fait via un navigateur web classique et les requêtes sont gérées par le protocole IIOP de CORBA.
Carlos O'Ryan and Douglas [103] décrivent une manière d'implémenter CORBA pour satisfaire la QoS requise pour les applications de la simulation distribuée. Ils présentent une extension à CORBA par la mise en place d'une communication distribuée via le middleware RTI-NG de DMSO. De même, [132] propose d'utiliser le protocole SIP pour assurer l'interopérabilité entre des RTI différents afin de pouvoir partager des applications de conférence sous HLA sur des réseaux grandes distances.
Wentong [131] discute l'utilité de développer des fédérations hiérarchiques pour la structuration de la simulation à grande distance. Il décrit l'utilisation d'une passerelle pour la structuration de la simulation et pour contrôler l'accès aux données de la simulation. Dans ce contexte, Jung [2] présente une autre approche pour implémenter des fédérations hiérarchiques. Benoît [17] présente l'utilisation d'un pont pour la communication inter-fédérations.

2.3.5 Techniques de réduction du trafic et d'optimisation de la bande passante

Les simulateurs distribués doivent échanger les états des l'entités qu'ils simulent de manière quasi-réelle pour maintenir une vision consistante de la simulation. Cependant, ceci n'est pas toujours réalisable car d'une part la QoS dans le réseau n'est pas mise en place, et d'autre part, la quantité de trafic échangé est assez grande pour être transmise sur le réseau qui est déjà contraint par d'autres types d'applications. Pour cela, des techniques de réduction de messages ont été proposées pour réduire la quantité de messages échangés.
Ces techniques avaient toujours la tendance à mettre en place une procédure de mise à jour

des états des entités simulées. Cette procédure repose sur la mise en oeuvre d'un même ensemble d'algorithmes dont l'objectif est tout d'abord, de conduire les simulateurs distants à diminuer le volume de trafic correspondant à la diffusion des états locaux des entités simulées localement, ensuite de permettre aux sites récepteurs de déterminer par extrapolation l'état d'une entité distante à chaque pas de la simulation. Toutefois, ces techniques diffèrent par les approches qu'elles utilisent pour contribuer à résoudre l'inconsistance de l'environnement virtuel. Par exemple, la technique de filtrage des pertinences permet d'adapter la communication entre des réseaux hétérogènes en éliminant les données non pertinentes. Bassiouni [89] propose d'utiliser la distance spatiale entre les entités de la simulation sous DIS pour filtrer le trafic envoyé par un réseau FDDI vers un réseau Ethernet.
Le service de gestion de la distribution de données de HLA (DDM ou Data Distribution Management) comporte aussi un mécanisme de filtrage par régions qui permet l'abonnement à certaines données via l'espace de routage de données [80]. Un espace de routage HLA est un système de coordonnées multidimensionnel selon lequel un fédéré peut implémenter son mécanisme de filtrage par région [109]. Le choix d'une région de routage se fait par un calcul des zones d'intersection entre les régions de producteurs de variables et les régions des consommateurs de variables. Ensuite, le moteur d'exécution (RTI) détermine quels fédérés peuvent accéder à quelle région. Les fédérés peuvent aussi choisir les variables auxquelles ils veulent accéder en précisant leurs seuils.
L'agrégation de paquets est une technique d'encapsulation des paquets qui permet de diminuer le temps de traitement des paquets dans les réseaux grande distance. Par exemple, dans des réseaux ATM, les unités IWU (Inter-Working Units) sont utilisées pour convertir le trafic DIS en cellule ATM [10]. Le flux de cellules est multiplexé et routé sur un back-bone ATM/ADSL pour lui permettre un lien de transmission haute vitesse et fiable. Comme un paquet ESPDU [1] est de 144 octets, il est fragmenté en 3 cellules ATM avant d'être mis sur l'interface réseau du commutateur ATM. Ensuite les cellules sont transmises sur des liens optiques fiables de haut débit. En arrivant à l'autre bout du réseau, les cellules sont rassemblées en niveau de l'interface IWU avant d'être adaptées à l'hôte.
La compression de données proposée par le protocole PICA [61] permet la diffusion de la différence des unités de données (mise à jour des états des objets simulés), qui dès sa réception sur les noeuds distants, permet de régénérer le paquet original . Toutefois, comme les entités ne changent pas rapidement leurs états, la taille des paquets de mise à jour est assez faible et sa différence entre deux paquets successifs est peu significative, particulièrement lorsqu'une entité devient totalement immobile, où elle cesse d'envoyer des informations sur son comportement (position, vitesse, accélération). Nous proposerons à la base de ces techniques le dead reckoning que nous étudierons sur les réseaux grande distance sans garantie de QoS dans la section 4.2 du chapitre 4.

2.4 Simulation distribuée à base de DDS

DDS (Data Distribution Service) [100] est un standard de l'OMG [2] qui cible les applications temps-réel distribuées centrées sur les données. Il intègre à la fois un modèle de données simple,

1. Entity State PDU
2. `http://portals.omg.org/dds/` (Object Management Group)

une distribution efficace et extrêmement configurable. Il présente une architecture producteur/-consommateur qui permet aux applications de communiquer tout simplement, par publication, l'information dont elles disposent et de souscrire à l'information dont elles ont besoin.
Le middleware DDS fournit les mécanismes nécessaires pour la prédictibilité et le contrôle des ressources tout en offrant la modularité, la scalabilité, l'évolutivité, la fiabilité et la robustesse de la communication. Sa structure est similaire à celle de HLA pour certains points [73] : une architecture de communication basée sur le modèle *producteur/consommateur*, qui supporte la modélisation par objets et la propagation des mises à jour des instances des objets, qui fournit un support de contenu basé sur les souscriptions, comparables au service de gestion de la distribution des données du standard HLA (espace de routage et régions du service Data Distribution Management).
La spécification DDS offre deux niveaux d'interface (Figure 2.4) : l'une de bas niveau, Data Centric Publish Subscribe (DCPS), hautement configurable, étroitement lié aux données et riche de nombreux paramètres QoS pour déterminer le comportement requis ; le niveau plus élevé, Data Local Reconstruction Layer (DLRL), qui offre une approche simplifiée du modèle OMT de HLA et qui permet une meilleure intégration au niveau applicatif.

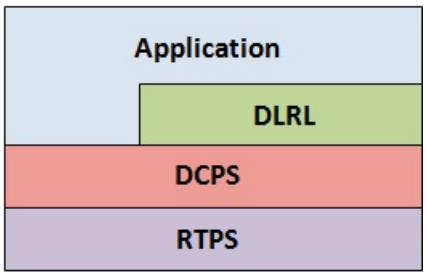


Figure 2.4: Les couches DDS

DDS fournit de nouveaux aspects non inclus dans HLA : un ensemble riche de politiques de qualité de service, un modèle de données fortement typé permettant la propagation d'état comprenant la distribution cohérente et ordonnée des données, ceci tout en omettant quelques autres aspects abordés par HLA, comme le management du temps et le management de fédération. En dépit de sa nouveauté, la technologie est bien démontrée, elle unifie la distribution de données avec succès sur des middlewares temps-réel tels que NDDS [112], OpenSpliceDDS [117] et OpenDDS [99].

2.4.1 Modèle de Communication

Le modèle conceptuel de la distribution de données dans DDS repose sur une abstraction d'un espace global de données typées partagé entre des processus applicatifs producteurs (publisher) qui produisent certaines de ces données et des processus consommateurs (subscriber) qui en consomment certaines. Un participant peut publier et s'abonner simultanément à des informations identifiées par nom de Topic (Figure 2.5). Le middleware DDS distribue les données de sorte que chaque participant pourra accéder aux valeurs les plus courantes. Ainsi, une application qui souhaite écrire une donnée dans l'espace partagé doit se déclarer comme producteur de ce topic et une application qui souhaite lire une donnée de l'espace partagé doit se

déclarer comme consommateur du topic qui lui est associé. DDS fournit également un modèle "de propagation d'état" qui permet par exemple, aux noeuds DDS la mise à jour efficace des valeurs des données uniquement lorsque celles-ci changent.

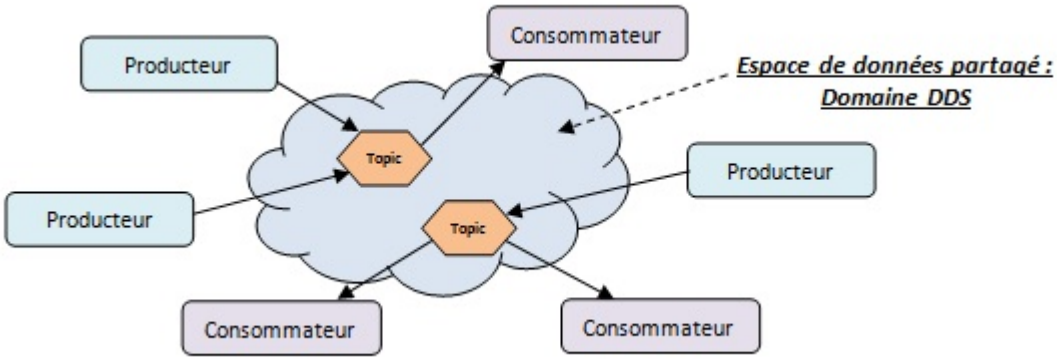


Figure 2.5: Espace de données partagé de DDS

Cette architecture permet d'assurer l'auto-configuration et la découverte automatique des différents noeuds distribués. DDS fournit la possibilité de spécifier pour chaque type de données plusieurs paramètres de QoS (débit, latence, durée de validité,...), qui permettent au concepteur de réaliser une application distribuée basée sur le besoin et la disponibilité de chaque type de données. Elle définit un ensemble d'interfaces permettant à des processus applicatifs d'échanger des informations en respectant des exigences de qualité de service prédéfinies.

Nous distinguons sur la Figure 2.6 des différentes entités intervenant pendant la distribution. En effet, à partir de ces déclarations d'intention (à produire ou à consommer un topic), le middleware DDS met automatiquement en relation les producteurs et consommateurs qui se partagent le même topic ; DDS se charge alors de livrer les différentes valeurs (également appelés échantillons) produites aux consommateurs qui se sont déclarés intéressés par le topic. Un topic correspond à un type de données qui peut être produit ou consommé dans l'espace global de données. Un topic associe un nom, un type de données et une spécification de QoS pour la distribution des valeurs de la donnée. Il permet donc de spécifier un flux de données, échangé entre processus applicatifs, identifié par un nom, conforme à un type et qui est distribué selon une certaine QoS. Un topic peut regrouper plusieurs instances de ce même topic qui peuvent être distinguées en utilisant une clef (désignée par "Key" dans DDS). Une instance est un sous flux du flux de données global d'un topic qui regroupe des échantillons de la donnée avec la même valeur de clef. Comme indiqué ci-après, un processus applicatif a la possibilité de souscrire à une instance de topic (un topic avec une valeur particulière de la clef).
Une application souhaitant produire un type de données doit attacher un DataWriter à un publisher. Le publisher est en charge de la distribution effective des données alors que le DataWriter est le moyen pour l'application d'indiquer au publisher la présence d'un nouvel échantillon. Un processus applicatif souhaitant transmettre un nouvel échantillon utilise donc un DataWriter pour activer au niveau du publisher la dissémination de la valeur de la donnée qui se fera en fonction de la QoS prévue. Contrairement à un publisher, un DataWriter est associé à un unique topic. Plusieurs DataWriters peuvent être attachés à un publisher. Des politiques de QoS y sont

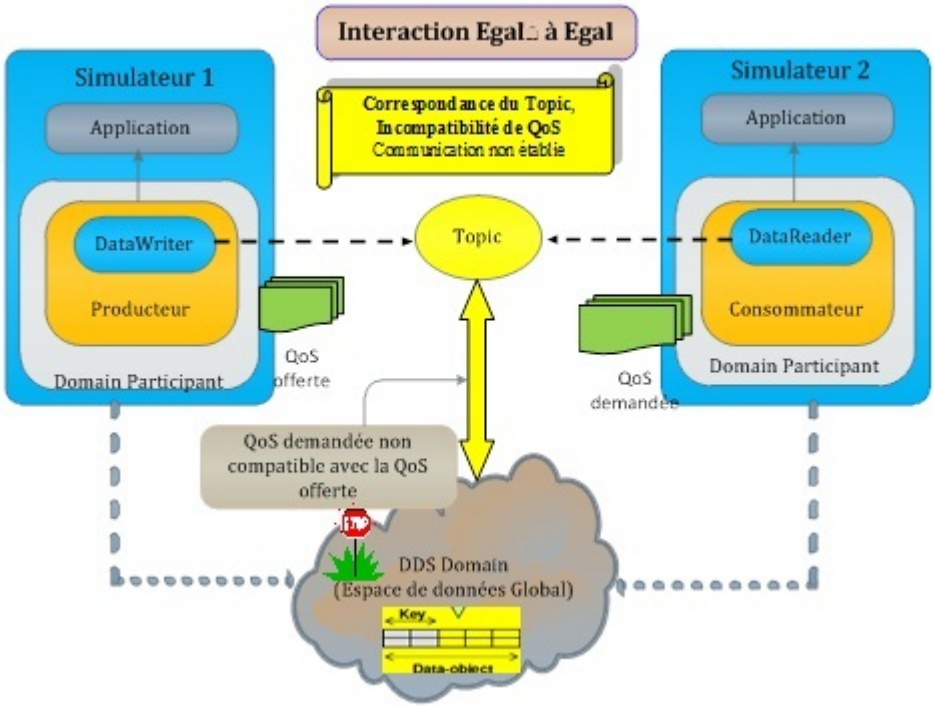


Figure 2.6: Principales entités de DDS

attachées pour paramétrer leur fonctionnement.
Du côté consommateur, un subscriber réceptionne les échantillons publiés des Topics auxquels il a souscrit. L'application utilise de son côté un DataReader (un par type de données) pour récupérer les échantillons reçus. Une application souhaitant consommer un type de données (spécifié par un topic) doit donc attacher un DataReader à un subscriber. Ce dernier engage, pour chaque type de données consommées, une phase de souscription qui lui permet de s'associer aux publishers respectifs. Cette souscription peut concerner tous ou partie des échantillons d'un topic en jouant sur les caractéristiques d'un topic, à savoir le nom, la clef éventuelle (souscription à une instance) ou le contenu/valeur des échantillons. La souscription vérifie que les paramètres de QoS associés aux publishers et subscribers sont compatibles (la QoS offerte par le publisher répond à celle requise par le subscriber). En cas d'incompatibilité la souscription échoue.

2.4.1.1 Politiques de Qualité de Service (QoS)

DDS permet un contrôle fin de la QoS basé sur les flux de données : chaque paire de producteur/consommateur peut établir un accord de QoS indépendant. Cet aspect, unique à DDS, permet aux applications de supporter les besoins extrêmement complexes et flexibles des flux de données. Les paramètres de la qualité de service contrôlent de façon virtuelle tous les aspects du modèle de DDS et des mécanismes de communications sous-jacents. DDS offre une granularité de politiques de QoS regroupée en cinq catégories :

– **Data Availability :** ce groupe de QoS contrôle la disponibilité et la durée de validité d'un échantillon produit.

- **Data Delivery :** ce groupe de QoS contrôle la manière dont les données sont délivrées et les entités qui ont le droit d'écrire des échantillons (en cas de présence de plusieurs producteurs possible).
- **Data Timeliness :** ce groupe de QoS contrôle l'aspect temporel associé à la distribution des échantillons d'un Topic.
- **Ressources :** ce groupe de QoS permet de contrôler et donc limiter les ressources nécessaires à la distribution des échantillons.
- **Configuration :** ce groupe de QoS correspond à des données qui peuvent être rattachées aux entités DDS pour les besoins de l'application. Ces données sont typiquement utilisées par l'application lors de la phase de découverte des participants à l'espace partagé de données. Le cas de Platsim sur DDS n'a pas besoin d'utiliser cette catégorie.

La Figure 2.7 montre un ensemble de paramètres de qualité de service adressé par DDS (nous citons uniquement celles qui nous servirons pour la suite).

QoS Policy	Applicability	RxO	Modifiable	
DURABILITY	T, DR, DW	Y	N	Data Availability
DURABILITY SERVICE	T, DW	N	N	
LIFESPAN	T, DW	-	Y	
HISTORY	T, DR, DW	N	N	
PRESENTATION	P, S	Y	N	Data Delivery
RELIABILITY	T, DR, DW	Y	N	
PARTITION	P, S	N	Y	
DESTINATION ORDER	T, DR, DW	Y	N	
OWNERSHIP	T, DR, DW	Y	N	
OWNERSHIP STRENGTH	DW	-	Y	
DEADLINE	T, DR, DW	Y	Y	Data Timeliness
LATENCY BUDGET	T, DR, DW	Y	Y	
TRANSPORT PRIORITY	T, DW	-	Y	
TIME BASED FILTER	DR	-	Y	Resources
RESOURCE LIMITS	T, DR, DW	N	N	
USER_DATA	DP, DR, DW	N	Y	Configuration
TOPIC_DATA	T	N	Y	
GROUP_DATA	P, S	N	Y	

Figure 2.7: Les politiques de Qualité de Service adressées par DDS

Pour la catégorie "Data Delivery", nous distinguons les paramètres de QoS suivants :

- **Reliability** : Elle permet de contrôler le niveau de fiabilité du service de distribution de données. Il peut prendre deux valeurs, à savoir Reliable ou Best-effort (Figure 2.8). Dans le premier cas, le service doit délivrer tous les échantillons à tous les DataReaders dans l'ordre et toute distribution d'échantillon qui échoue est retentée. Dans le second cas, l'application indique qu'elle peut tolérer des échecs de distribution d'échantillons.

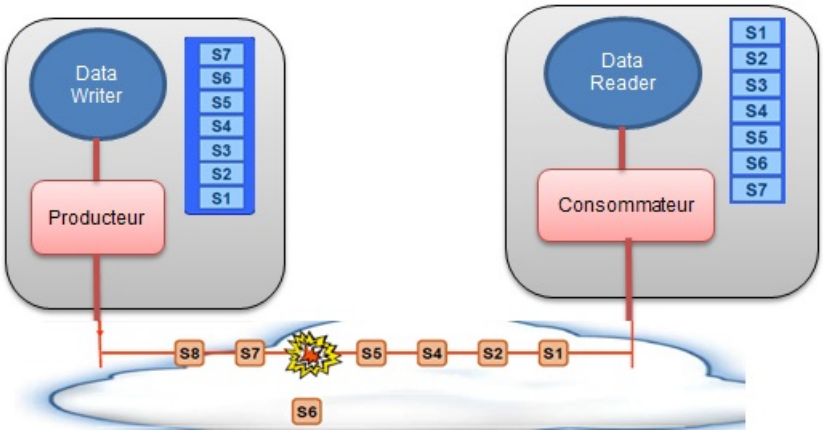


Figure 2.8: Les paramètres Reliability

– **Ownership :** Il permet d'identifier le ou les producteurs qui ont le droit d'écrire des échantillons d'un topic. Ce paramètre peut prendre deux valeurs : Shared ou Exclusive. Dans le premier cas, plusieurs DataWriters ont le droit d'écrire des échantillons et ces derniers seront distribués aux DataReaders concernés (Figure 2.9). Dans le second cas, un topic ne peut être mis à jour à un instant donné que par un seul DataWriter. Seuls les échantillons produits par ce dernier seront distribués. Le choix du DataWriter qui a l'exclusivité de la mise à jour est déterminé par une autre QoS Policy, à savoir "Ownership_Strength" qui correspondant à un niveau de priorité qui permet de départager les différents DataWriters.

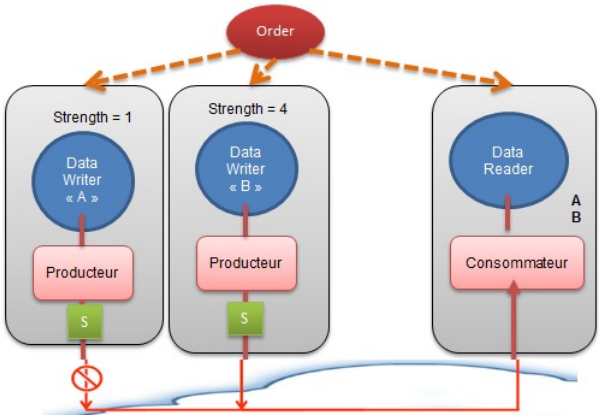


Figure 2.9: Les paramètres Ownership

La catégorie "Data Timeliness" regroupe les éléments suivants :

– **Deadline :** c'est une durée qui permet de spécifier, comme le montre la Figure 2.10 côté DataReader, la durée maximale entre deux arrivées successives d'échantillons. Elle permet d'identifier des exigences minimales sur le comportement du ou des publishers. Côté DataWriter, elle permet de spécifier un engagement de la part de l'application sur la durée maximale entre deux écritures successives (relatives à un topic).

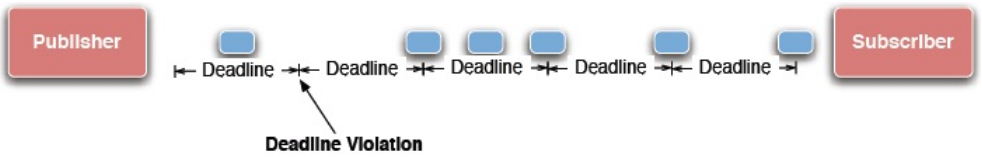


Figure 2.10: Les paramètres Deadline

– **Latency_Budget :** Il permet à l'application de spécifier une exigence sur la durée maximale entre le moment où un échantillon est écrit et le moment où il est placé dans la mémoire cache du DataReader à l'attention de l'application (Figure 2.11). En l'état actuel du standard DDS, ce paramètre est fourni à DDS pour lui permettre d'évaluer l'urgence ou la criticité de la distribution. Le middleware n'est pas obligé de garantir ce délai ni de vérifier s'il est respecté.

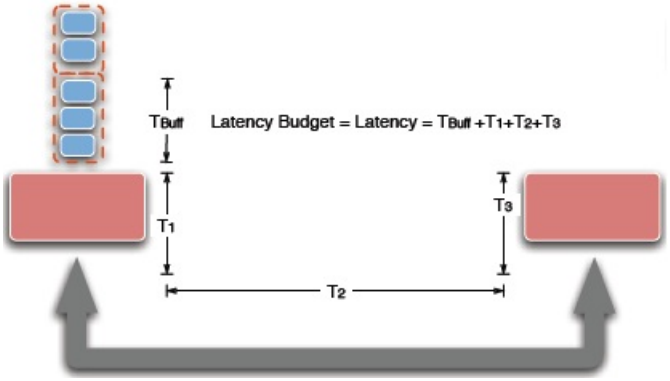


Figure 2.11: Les paramètres Latency_Budget

– **Transport_Priority :** Elle permet à l'application d'indiquer un niveau de priorité à prendre en compte lors de la distribution des échantillons. Cette indication de priorité est supposée être prise en compte par DDS et le réseau lors du traitement et la transmission des messages. "Transport_Priority" prend des valeurs entières. Plus la valeur est grande, plus la priorité est élevée.

La catégorie "Ressources" comporte quant à elle les paramètres suivants :

– **Time_Based_Filter :** Il permet à une application consommatrice de spécifier une durée minimale à respecter entre deux arrivées successives d'échantillons. Par le biais de ce paramètre, l'application indique qu'elle ne souhaite pas forcément consommer tous les échantillons (d'un topic ou instance de topic) mais plutôt se limiter à des mises à jour séparées par cette durée minimale (Figure 2.12). Cette politique permet de mieux contrôler (et limiter) les ressources réseau et machine nécessaires à la distribution des échantillons.

– **Resource_Limit :** Elle permet de contrôler les ressources nécessaires à la distribution des échantillons d'un topic. Trois paramètres sont définis (1) "max_Samples" qui spécifie le nombre maximal d'échantillons (toutes instances de topic confondues) qui peuvent, à un instant donné, être gérés par un DataReader (ou DataWriter) (2) "max_instances" qui spécifie le nombre maximal d'instances de Topics qui peuvent être gérées à un instant donné par un DataReader (ou DataWriter) (3) "max_samples_per_instance" qui spécifie le

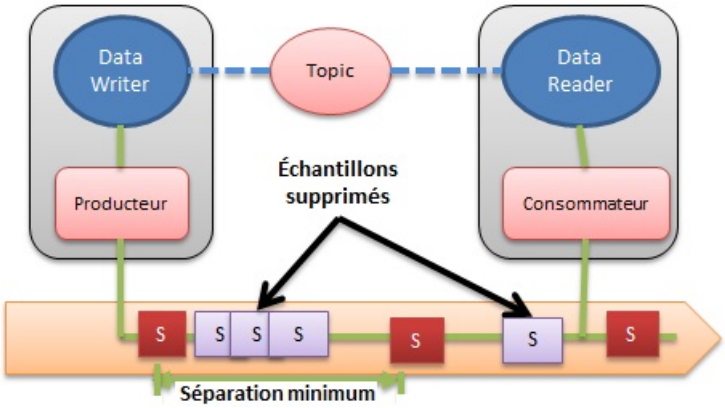


Figure 2.12: Les paramètres Time_Based_Filter

nombre maximal d'échantillons par instance qui peuvent être gérés à un instant donnée par un DataReader (ou DataWriter).

Finalement, la catégorie "Data availability" contient les paramètres suivants :

- **Durability :** Elle permet à l'application de contrôler si les échantillons écrits dans l'espace global de données à un instant donné méritent d'être maintenus à l'attention d'éventuels futurs consommateurs. Ce paramètre peut prendre les valeurs suivantes : Volatile, Transient et Persistent. Si configurée à volatile, la distribution d'un échantillon touchera les DataReaders connus (qui y ont souscrits préalablement) au moment de sa production. Aucun échantillon n'est maintenu. Si fixée à Transient, le service de distribution cherchera à maintenir (au-delà du moment de production) certains échantillons pour le compte d'éventuels futurs DataReaders. Les échantillons maintenus sont déduits d'autres QoS policies (History, Resource_limits, etc.). Souvent, ils sont maintenus tant que le DataWriter existe. Enfin, si configuré à Persistent, ces échantillons sont sauvés de manière permanente sur des supports appropriés (mémoire non volatile).
- **Lifespan :** Elle permet à l'application de contrôler la durée de validité d'un échantillon produit et d'éviter que des échantillons trop anciens (et donc non valides du point de vue de l'application) ne soient délivrés à l'application. Les échantillons périmés sont éliminés en les enlevant des mémoires cache des DataWriters et des éventuelles mémoires utilisées pour répondre à une durabilité temporaire (Transient) ou persistante.
- **History :** Cette QoS policy est décrite dans la Figure 2.13. Elle est importante dans la situation où la consommation (ou la distribution) des échantillons n'arrive pas à suivre le rythme de leur production. Dans cette situation, elle permet de contrôler si le service doit livrer tous les échantillons, ou les **n** derniers échantillons. La policy History prend alors respectivement les valeurs **KEEP_ALL, KEEP_LAST** avec un attribut supplémentaire Depth fixé à **n**. Il est à noter que cette QoS policy peut être fixée de manière indépendante du côté DataReader et DataWriter. Du côté DataWriter, elle permet de contrôler le comportement de ce dernier si le rythme des écritures est plus grand que celui de la distribution. Du Côté du DataReader, elle contrôle son comportement lorsque le rythme des arrivées des échantillons dépasse le rythme de consommation de la part de l'application. Enfin, si la "Durability" est fixée à "Transient" ou "persistent", le paramètre History

permet de délimiter les échantillons qui méritent d'être maintenus à l'attention de futurs DataReaders.

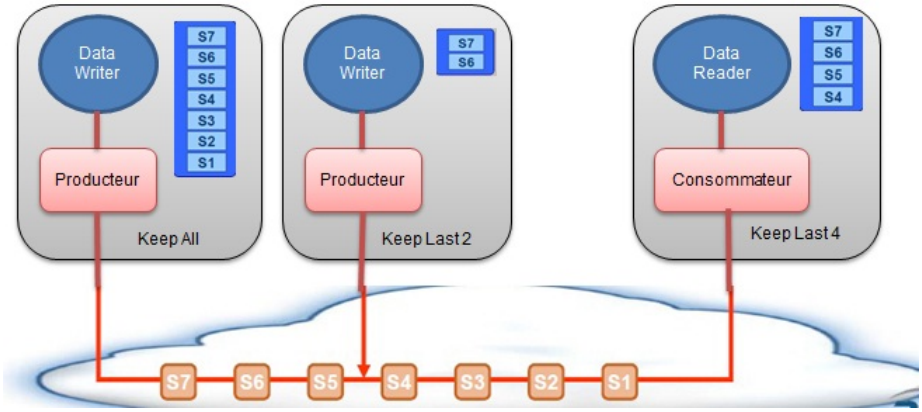


Figure 2.13: Le paramètre History

2.4.2 Spécificité de la distribution des données avec DDS

Pour cette étude nous avons utilisé deux implémentions du middleware DDS : La première est *OpenSplice* [117] vu qu'il est open source pour permettre de mieux comprendre le comportement de son service réseau, et la deuxième "RTI DDS" [112] qui est un des pionniers des solutions DDS. Il existe des différences notables liées aux implémentations dont nous tiendrons compte par la suite et qui justifient cette comparaison.
Les deux middlewares considérés s'intègrent dans une architecture TCP/IP. Au niveau transport, ils utilisent un service UDP. Il est à noter que certaines implémentations de DDS reposent sur le protocole RTPS [101] qui permet l'interopérabilité dans le cas où différentes implémentations de DDS sont utilisées. Ce protocole vient donc s'ajouter au-dessus du protocole UDP. À titre d'exemple, RTI DDS repose sur le protocole RTPS alors qu'OpenSplice DDS donne la possibilité de choisir entre RTPS et un protocole spécifique à OpenSplice DDS.

2.4.2.1 Les modes de communication des deux middlewares

Pour permettre la livraison des données via le réseau, les middlewares DDS peuvent reposer sur les trois modèles du service IP : unicast, multicast et broadcast (en utilisant respectivement des adresses IP unicast, multicast ou broadcast). Le choix du modèle de service n'est pas trivial et a un impact sur les ressources réseau utilisées. A titre d'exemple, si un Publisher utilise le service unicast pour distribuer une donnée, une copie de cette donnée sera individuellement envoyée (dans un paquet IP) à chaque Subscriber (noeud terminal). Si le service broadcast est utilisé, la donnée est diffusée dans un paquet IP vers tous les noeuds participant au domaine DDS et ce même si certains n'ont pas de Subscribers intéressés par cette donnée (dans ce cas, faute de consommateur, la donnée reçue sera éliminée au niveau de la couche DDS). L'utilisation du service multicast semble le plus approprié mais implique une gestion compliquée des groupes multicast (groupes dynamiques dont le nombre peut devenir élevé). Les deux middlewares considérés ne supportent pas exactement les mêmes modes. OpenSplice DDS supporte l'unicast, le

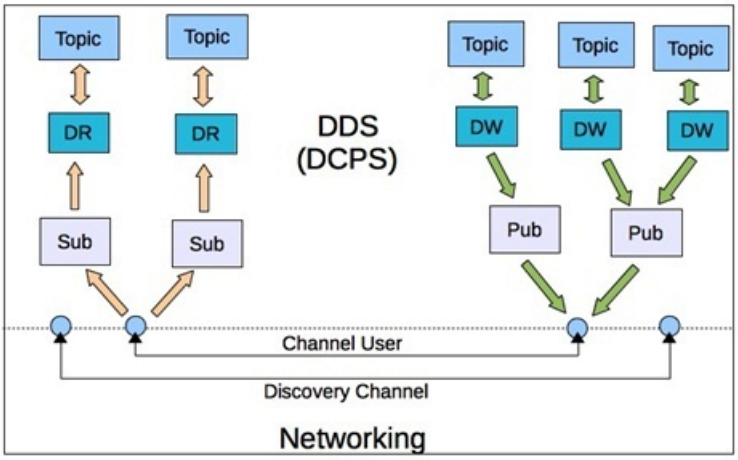


Figure 2.14: Service networking de OpenSplice-DDS

multicast et le broadcast (modèle par défaut) alors que RTI-DDS supporte le modèle unicast (par défaut) ainsi que le multicast.

2.4.2.2 Service Networking d'OpenSplice DDS.

Dans OpenSplice DDS [1], une couche s'intercale entre la couche transport et la couche DDS (DCPS) pour implémenter le service networking. Cette couche offre des services à la couche supérieure (DCPS) et utilise les services de la couche inférieure (la couche transport). Deux types de service sont offerts à la couche DCPS : "Channel User" et "Discovery Channel" (voir Figure 2.14). Le "Channel User" est utilisé par les Publishers et les Subscribers. Plusieurs Publishers peuvent se partager un "Channel User". De la même manière du côté réception, plusieurs Subscribers peuvent se partager un "Channel User". Les "Discovery Channel" quant à eux sont dédiés au service de découverte.
Un "Channel User" peut fonctionner en mode fiable ou en mode non fiable. Dans le cas non fiable, il n'y a aucune garantie que la donnée soit délivrée. Pour ce type de communication, deux sockets UDP sont utilisés, un pour l'envoi (côté producteur) et un pour la réception (côté subscriber). Dans le cas du mode fiable, le service networking met en place des mécanismes pour assurer la livraison des données. Pour ce type de communication, quatre sockets UDP sont utilisés, un socket pour l'envoi des données, un socket pour l'envoi des acquittements, un socket pour la réception des données et un socket pour la réception des acquittements.
La Figure 2.15 présente un exemple qui permet d'illustrer le fonctionnement des sockets dans OpenSplice DDS. Les sockets utilisés pour l'envoi des données possèdent des numéros de port alloués dynamiquement (noté dans la figure W, X, Y, Z).
Du coté réception, les numéros de port sont fixés préalablement. Ils sont indiqués dans la configuration du service networking. Dans cet exemple, on peut considérer que le port 53370 est utilisé pour recevoir des données non fiables, que le port 53380 est utilisé pour recevoir les données fiables, le port 53381 est utilisé pour recevoir les acquittements et que le port 53390 est utilisé pour recevoir les messages liés au service de découverte. Ici, nous avons 53380 comme

1. http ://www.prismtech.com/opensplice

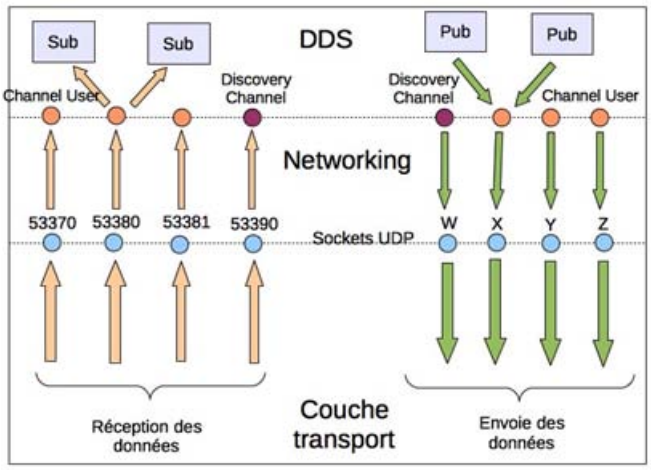


Figure 2.15: Service networking de OpenSplice-DDS

numéro de port pour le socket de réception des données fiables et 53381 pour la réception des acquittements correspondants. Cela vient du fait que le numéro de port du socket pour la réception des acquittements correspond au numéro de port du socket associé à la réception des données fiables incrémenté de 1. Donc pour une communication fiable, les sockets marchent "par paires".

2.4.2.3 Modèle de distribution dans RTI DDS

Montré dans la Figure 2.16, la distribution de données dans le middleware RTI DDS est gérée par un "Plugin" de transport qui caractérise le mode de transfert de données. Ce dernier utilise plusieurs canaux pour la transmission de données sur le réseau. En effet, plusieurs sockets doivent être créées pour l'envoi et la réception des données, groupées par nature de transmission en multicast et/ou en unicast. Les ports utilisés sont :

- Port 7400 : Supporte le trafic de découvert des participants (Built-in Multicast port Offset),
- Port 7401 :Supporte le trafic de découvert point à point (User MultiCast Port Offset),
- Port 7410 : Alloué pour l'échange du "Méta trafic utilisateur" (ex. :HEARTBEAT, ACK_NACK, INFO_DST, etc.), Built-in unicast port Offset
- Port 7411 : Alloué pour l'échange de données utilisateur "UserData" (User Unicast Port Offset),
- les ports d'envois sont alloués dynamiquement par le système (Meta Unicast Port 65660, Meta Multicast Port 65650, User Unicast Port 65661, User Multicast Port 65651)

2.4.3 Conclusion sur la simulation distribuée

Depuis leurs créations, les standards des applications de simulation distribuée interactive ont connu une grande évolution. Partant du modèle basique de SIMNET, les évolutions vers des architectures client/serveur comme le cas du protocole DIS ont connu un grand succès. Avec l'émergence des jeux multi-joueurs en ligne sur des réseaux comme l'internet, la communauté scientifique s'est orientée vers les architectures de type égal à égal avec les premières utilisations

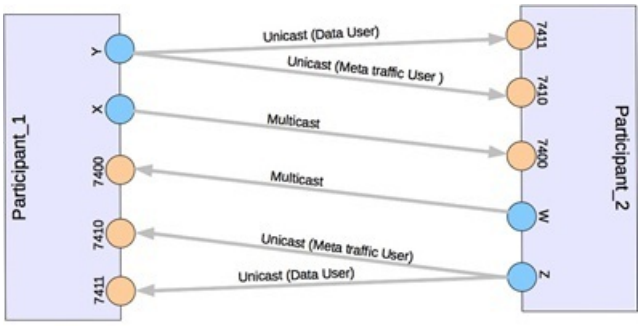


Figure 2.16: Schéma de distribution dans RTI DDS

Domaine d'étude	DIS	HLA	DDS
Domaine d'application	entraînement temps réel	Tout type de simulation	applications temps-réel
Interopérabilité	Oui	Oui	Oui
Réutilisation	Non	Oui	Oui
Multi sites et utilisateurs	Oui	Oui	Oui
Architecture	Egal à égal	Pub/Sub	Pub/Sub
Systèmes d'exécution	PDU de gestion	RTI	DDS
Interface informatique	Coupleur	API	API
Base de données	Dupliquée	Distribuée	Distribuée
Point d'accès au service de transport	socket	services et callbacks	services de notification
Filtrage messages	non	DDM	Content filter
Gestion de temps	Temps réel	Temps réel/coordonné	Temps réel

Table 2.1: Comparaison de différentes architectures

du protocole DIS. Les différentes architectures que nous venons de décrire dans ce chapitre concourent à l'interopérabilité d'applications hétérogènes sur des systèmes différents.
Le choix d'une architecture pose des contraintes sur la distribution des simulations et l'interopérabilité entre les simulateurs. Pour cela, nous avons présenté une comparaison de ces différentes architectures au Tableau 2.1 selon certains critères bien que leurs fonctionnalités soient différentes : Par exemple, DIS offre des simulations en temps réel, permettant à des unités militaires de travailler ensembles au combat. C'est une architecture de simulation pour le milieu militaire. Elle n'est malheureusement pas disponible pour le monde civil puisque les PDUs qui interagissent entre les entités sont toutes d'ordre militaire. HLA fournit une architecture de bas niveau d'abstraction afin d'assurer l'interopérabilité et la réutilisation du code des fédérés.
DDS propose une solution imitant la distribution des journaux via le paradigme *Production/-Consommation*, où la distribution des données est rendue beaucoup plus simple : le producteur produit ces données dans un espace de partage de données virtuel et les consommateurs viennent souscrire uniquement aux données dont ils ont besoin. Cette approche diminue considérable-

ment le trafic circulant sur les réseaux de communications qui sont déjà surchargés.
Même si ces architectures semblent être matures pour assurer la communication dans des environnements contrôlables dont les contraintes sont maîtrisées, le passage à des réseaux à large envergure nécessite des mécanismes plus sophistiqués pour assurer la qualité de service tout au long du chemin de communication entre les simulateurs distribués. Par conséquent, nous discuterons les architectures de gestion de la QoS pour les simulations distribuées en abordant successivement le contexte des réseaux locaux et des réseaux distants.

2.5 Gestion de la Qualité de Service

2.5.1 Métriques pour la QoS

La Qualité de Service (QoS) peut être définie par la mesure de l'aptitude du réseau et du système informatique à fournir différents niveau de services pour des applications ou des flux réseaux bien identifiés. La difficulté rencontrée lors de la mise en place d'une communication à garantie de QoS de bout-en-bout, est de décrire la manière de faire réussir l'intégration de la QoS sur les deux axes vertical et horizontal. Sur l'axe vertical, il s'agit de spécifier la méthodologie qui permet de descendre avec les besoins applicatifs de l'utilisateur vers les ressources physiques disponibles afin de répondre aux besoins de ces applications. L'intégration horizontale concerne la garantie de la QoS de bout-en-bout, en traversant des réseaux d'opérateurs différents supportant des technologies d'accès différentes et qui offrent des services très différents.
Dans le cadre des recommandations de la série E.800 [37], l'IUT décrit un ensemble de paramètres quantitatifs de la QoS pour évaluer de manière objective les services offerts. Les paramètres les plus utilisés aux réseaux et ayant un impact direct sur l'application sont : la disponibilité, la bande passante, la latence, la gigue, le taux de perte. Dans cette partie nous allons définir ces paramètres et identifier leur impact sur les applications et les éléments qui interviennent pour garantir ou détériorer la QoS.

La disponibilité : La disponibilité d'un service est le temps entre l'ouverture de la connexion et le temps global nécessaire pour l'accès à ce service.

Bande passante : Le débit de transmission d'un flux est défini comme le nombre de bits émis pendant un intervalle de temps divisé par la longueur de l'intervalle

Délai : Le délai de transit est définie dans le RFC2679 [3] par : "*Le délai de transit d'un paquet à travers un réseau complet est donné par le temps entre l'émission du premier bit du paquet par la source et la réception du dernier bit du paquet par le destinataire*". Pour une analyse plus fine, le délai de transit peut se décomposer en plusieurs composantes :

- **Le délai de traitement :** c'est le temps nécessaire par un élément du réseau (ex. routeur, machine hôte,...) pour le traitement de l'en-tête d'un paquet et la détermination de la liaison de sortie correspondant à sa destination. Il dépend de la vitesse de traitement des éléments matériels et de la complexité des fonctions de traitement qui lui sont associées.

- **Le délai d'attente :** ce délai correspond au temps passé dans les files d'attentes des éléments du réseau, et qui dépend de l'état instantané du réseau ; par exemple si le réseau est congestionné la taille des files d'attente devient plus grande et le temps de traitement devient plus long.
- **Le délai de transmission** : c'est le temps nécessaire pour l'acheminement d'un paquet à un certain débit : Délai de transmission = nombre de bit à transmettre/taux de transmission
- **Le délai de propagation :** c'est le temps nécessaire au trajet entre l'émetteur et le récepteur à travers le medium : Délai de propagation= distance physique/vitesse de propagation

Gigue : La variation maximale absolue du délai de transit définit la gigue [33]. Elle correspond à la variation de la latence, mesurant l'écart entre les délais d'acheminement de tous les paquets d'un même flux.

Taux de perte : Le taux de perte de paquets, définie dans le RFC2680 [4], indique le nombre d'octets ne pouvant pas rejoindre la destination en fonction du nombre total d'octets transmis par la source (pertes non détectées (cas d'UDP), mauvaise qualité du support de transmission, congestion du réseau).

2.5.2 Gestion de la QoS dans les réseaux locaux

2.5.2.1 Fourniture de QoS dans un Ethernet

Dans les réseaux locaux de type Ethernet, un ensemble de fonctionnalités ont été décrites par les normes complémentaires 802.1p [65] et 802.1q [66]. Ces deux standards sont utilisés pour l'identification des trames Ethernet dans les réseaux virtuels VLAN. Il s'agit de rajouter dans chaque trame Ethernet destinée à être transmise entre des commutateurs de niveau 2 (couche 2 du modèle OSI) des en-têtes supplémentaires contenant les informations suivantes :

- Le champ EtherType illustré dans la Figure 2.17 (dit aussi Tag Protocol ID) permet de préciser si la trame est signée selon le protocole 802.1Q.
- le champ "user priority" illustré dans la Figure 2.18 est défini par le standard IEEE 802.1p dans le cadre d'une interconnexion de niveau 2 de réseaux locaux. Ce champ de 3 bits est une partie de l'en-tête 802.1Q qui également spécifie le VLAN (Virtual LAN) auquel appartient la trame. l'IEEE 802.1D a défini 7 classes de trafic auquel le champ "user priority" peut se référer.
- Le champ CFI (Canonical Format Indicator) indique que le format de l'adresse MAC de l'interface réseau est standard.
- Le champ VLAN ID permet d'identifier le VLAN auquel appartient la trame.

Fournir la QoS au niveau MAC est défini par le standard IEEE 802.1p qui correspond à la définition de différentes classes de trafic. Le nombre de classes trafic qu'on peut assigner aux paquets dépend de la capacité du commutateur Ethernet et du nombre de ses files d'attentes qui permettent la sélection de ces paquets. De manière générale, le standard 802.1p définit 8 classes de services (la Figure 2.18) exprimé par le champ 'user priority'.

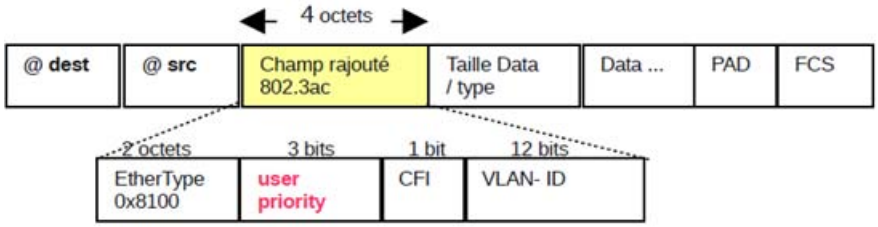


Figure 2.17: trame Ethernet sur VLAN

Type de service	802.1p
Administration réseau (*Network Control*)	7
Voix (*Voice*)	6
Vidéo (*Video*)	5
Avec charge contrôlée (*Controlled Load*)	4
A un excellent effort (*Excellent Effort*)	3
Avec économie (*Spare*)	2
Arrière-plan (background)	1
Best effort	0

Figure 2.18: Classe de services de l'IEEE 802.1p

2.5.3 Gestion de la QoS dans les réseaux grande distance

Les réseaux grande distance sont formés par des systèmes autonomes. Un système autonome est un ensemble de réseaux sous l'administration d'une seule entité ayant une politique de routage interne cohérente et commune. Le routage est un mécanisme par lequel les routes sont choisies sur le réseau pour l'acheminement des données vers une ou plusieurs destinations. La qualité de service dans le réseau dépend du traitement effectué sur les nœuds intermédiaires du réseau, du nombre de nœuds traversés et le temps de traitement sur chaque nœud. Chaque nœud ou routeur possède une table de routage (statique ou dynamique) qui est utilisée pour retrouver le meilleur chemin vers la destination. En effet, on distingue trois niveaux différents de routeurs qui fonctionnent avec deux types de protocole de routage :

- Les routeurs de cœurs sont les routeurs de base qui relient les différents réseaux.
- les routeurs externes utilisent des protocoles de routage capable d'échanger les informations entre des systèmes autonomes : Border Gateway Protocol(BGP)
- les routeurs internes se basent sur les protocoles de routage utilisé dans le système autonome : Interior Gateway Protocol (IGP)

2.5.3.1 Routage intra-domaine

Le routage intra-domaine se base sur le protocole de routage IGB [51] (Interior Gateway Protocol) qui se distingue par deux approches pour réaliser leurs algorithme de routage : les algorithmes basés sur les vecteurs de distance (Distance-Vector) et les algorithmes basés sur l'état des liens (Link-state) [108]. Le routage à vecteur de distance (Bellman-Ford) transmet d'un routeur à l'autre des copies périodiques d'une table de routage. Ces mises à jour permettent de communiquer les modifications de topologies entre les routeurs directement connectés. Le protocole RIP (Routing Information Protocol) se base sur l'algorithme de routage à vecteur de

distance dont la métrique est le nombre de saut qu'un paquet doit traverser pour atteindre sa destination finale [83]. RIP connaît des inconvénients (un lent temps convergence des informations de routage, le passage à l'échelle, et le nombre maximal de saut est limité, pouvant engendrer des boucles de routages qui conduit les routeurs à incrémenter infiniment la métrique de routage) qui font qu'il fut remplacé par les protocoles à état de lien comme OSPF [90][91] (Open Shortest Path First) [92][93] et IS-IS (Intermediate system to intermediate system) [102] [21].

2.5.3.2 Routage Inter-Domaines

Comme l'Internet est composé de réseaux indépendants et interconnectés entre eux, chaque domaine est administré par des responsables différents. Les protocoles de routage inter-domaine sont créés pour permettre l'échange d'informations de connectivité entre les systèmes autonomes. Un système autonome est une notion utilisée par le protocole BGP (Border Gateway Protocol) [81].
Contrairement aux protocoles classiques, BGP n'utilise pas les métriques classiques, mais dépend des routes, des attributs de préfixes, des politiques des opérateurs ou bien de diverses règles de sélection de réseaux.
BGP est constitué de deux parties : iBGP [11] (Interior BGP) et eBGP (Exterior BGP). iBGP est utilisé dans les routeurs de cœurs du réseau à l'intérieur des systèmes autonomes, alors que eBGP est utilisé dans les routeurs de bordure pour interconnecter les systèmes autonomes.

2.5.3.3 Routage à QoS

L'objectif retenu par l'IETF pour le routage à QoS est de trouver une route optimale qui satisfait les contraintes de la QoS pour un flux de données. Pour ce faire, des métriques de QoS et des critères de sélection de chemin ont été ajoutées dans les différents protocoles de routage existants pour le calcul de chemin à la suite de modifications topologiques. Un protocole de routage à QoS doit aussi indiquer les perturbations qui peuvent arriver et qui sont dues aux changements des routes, permettre l'acheminement du trafic Best Effort sans réservation de ressources et diminuer la surcharge induite par la consommation des ressources de calcul.
La RFC2386 [30] définissant les conditions auxquelles doit satisfaire ce modèle a été rédigée par le groupe de travail QoSR, avec pour objectifs de faire évoluer les protocoles de routages pour supporter la QoS (classes de services), favoriser les interactions simples, cohérentes et stables entre les domaines administratifs (AS) (éviter la sélection fréquente de routes) et utiliser les routes qui existent déjà. Les techniques de routages à QoS sont utilisées conjointement avec des mécanismes de réservation de ressources, de contrôle d'admission et d'ingénierie de trafic. On peut trouver les mécanismes suivants :

- classification : la classification de trafic a pour but d'identifier le flux auquel chaque paquet appartient. Un flux (également appelé classe de trafic) peut avoir une granularité très fine et correspondre à un flux de paquets échangés entre deux processus applicatifs ou large et correspondre à une famille ou classe d'applications ayant des exigences et caractéristiques similaires. Elle intervient au niveau de tous les nœuds du réseau (du moins ceux qui cherchent à distinguer les ressources associées à différents flux) et peut prendre des formes plus ou moins complexes ;

- contrôle (policing) : Il intervient sur les interfaces d'entrée d'un nœud intermédiaire et cherche à vérifier qu'un flux de communication identifié respecte un profil de trafic prédéfini (typiquement un débit à ne pas dépasser sur un intervalle de temps donné) et prend des actions (sanctions) en conséquence. Ces sanctions peuvent être le retardement du paquet, son marquage à un niveau de priorité faible voire son élimination.
- gestion des files d'attentes : permet de contrôler l'utilisation d'un buffer et décide les paquets qui peuvent être mémorisés dans le buffer en attente de transmission et ceux qui doivent être éliminés faute de place.
- ordonnancement : Il intervient au niveau d'une interface de sortie et permet de choisir parmi les paquets en attente dans le buffer, le prochain paquet à transmettre.

2.5.4 MPLS et ingénierie du trafic

Le protocole MPLS [8] (Multi-Protocol Label Switching) est un protocole à commutation de paquets normalisé par L'IETF qui utilise des étiquettes (labels) permettant l'intégration de plusieurs protocoles de routage (multi-protocoles) et de commutation de paquets (label Switching) à différents niveaux dans les réseaux (ATM, Ethernet, IP,...) [94][45].

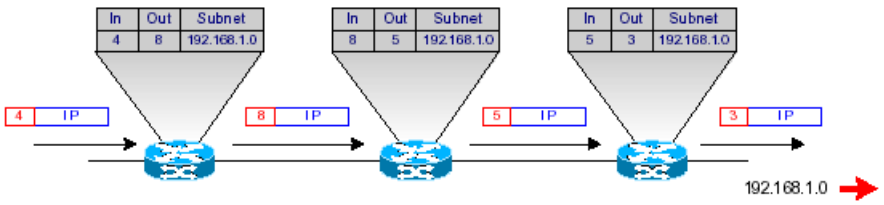


Figure 2.19: MPLS ingénierie de trafic

Comme montré dans la Figure 2.19, les étiquettes (label de 20 octets), simples identifiants numériques, identifient le trafic spécifique aux couches basses (adresse MAC Ethernet, champs VPI/VCI des cellules ATM [95]) à l'intérieur d'un réseau MPLS. Les routeurs permutent ces labels associés à un groupe de paquets encapsulés et transportés dans les en-têtes de niveau 2 et 3, acheminés de la même manière dans le même chemin et subissant le même traitement, sans consulter l'entête IP et leur table de routage, formant ainsi une classe spécifique de trafic dite FEC (Forwarding Equivalent Classes).
Typiquement, les FEC sont des préfixes IP correspondant à une plage d'adresse destination pouvant aussi être définis par des informations de QoS comme le type de trafic, les priorités,... On trouve aujourd'hui des réseaux VPN (Virtual Private Network) de niveau 3, dits VPN IP qui permettent de rassembler les propriétés que l'on trouve sur les réseaux Intranet ou Extranet, et qui permettent à la fois la gestion des communications fixes et mobiles. Ces VPN peuvent se classer en deux catégories : les VPN de niveau IP et les VPN réalisé à l'aide le protocole MPLS.

2.5.5 Modèles existants pour la gestion la QoS

Plusieurs mécanismes de QoS ont été développés par l'IETF selon leur complexité de réalisation, la granularité de l'allocation de ressources aux utilisateurs, la scalabilité et leurs coûts.

Ces critères sont très critiques pour juger le domaine d'applications d'une architecture de QoS. Par exemple, les mécanismes de contrôle de flux et de congestion nécessitent des délais d'aller-retour ne dépassant pas les 100ms. Les mécanismes de réservation de ressources et de contrôle d'admission nécessitent des délais de l'ordre de 1 seconde vu qu'ils opèrent à l'échelle de sessions utilisateurs. D'autres mécanismes considèrent le contrôle de QoS par flux individuels, mais leur réalisation reste complexe et leur coût est très élevé. L'agrégation de flux permet de réduire la quantité de données de contrôle et le temps de traitement nécessaire sur les noeuds du réseau. En outre, les mécanismes de contrôle peuvent être localisés sur des terminaux utilisateurs, à la bordure du réseau ou au coeur du réseau. Ces mécanismes nécessitent des protocoles de signalisation qui installe des informations le long du chemin de données.
Deux grandes catégories de services, qui se composent de sous-services dotés de différentes QoS ont été proposés : les Services Intégrés et les services différenciés. Les services intégrés sont gérés indépendamment les uns des autres (généralement sur les réseaux d'accès) alors que les services différenciés rassemblent plusieurs applications ensemble (généralement sur les réseaux de coeur).

2.5.5.1 Service Intégré

Le modèle IntServ a été développés par l'IETF [13] afin de faire évoluer l'architecture de l'Internet dédiée au transfert de fichiers vers un réseau multi-service offrant, en plus des services classiques, des services adaptés aux applications temps-réel et multimédia. IntServ est capable de prendre en charge la QoS en définissant des mécanismes de contrôle supplémentaires sans toucher au protocole IP. L'objectif de cette architecture est de fournir une QoS couplée au chemin entre l'émetteur et un ou plusieurs récepteurs (Figure 2.20). La réservation de ressources est faite de manière explicite avant la transmission de données. Si le réseau ne possède pas les ressources suffisantes, la réservation est rejetée. Dans ce cas, la source est amenée à adapter la demande des ressources afin de satisfaire la QoS requise. Les services intégrés ne peuvent pas contrôler de manière stricte le délai de transit des différentes applications. Pour cela, deux autres services ont été définis : le service " QoS Garantie " et le service "Charge Contrôlée" :

- **Service Garantie (GS)** : le service de QoS Garantie [120] fournit un niveau de QoS strictement contrôlé, similaire à la QoS d'un flux sur une liaison dédiée entre l'émetteur et le récepteur. Un flux utilisant ce service est complètement isolé des autres flux du réseau. Les paquets utilisant le service GS sont transmis avec un débit de transmission garanti, un délai d'attente total strictement borné sans perte de paquets liées au débordement des files d'attente dans les noeuds du réseau (si le flux respecte les paramètres réservés).
- **Charge Contrôlé (CL)** : le service Charge Contrôlée [135] propose un service de bout-en-bout exprimable de façon quantitative en termes de bande passante : il assure que le transfert de données sera fait de façon équivalente à un réseau peu chargé, non congestionné. Cependant, le service de QoS CL n'offre pas une garantie stricte quant au délai de transit ou à la perte de paquets. Cette QoS doit être fournit indépendamment de la charge réelle du réseau, elle peut être utilisée pour le trafic multimédia qui peut tolérer des pertes occasionnelles ou qui est capable d'adapter la taille du tampon de lissage en fonction de mesures de la variation du délai.

Le cadre de fonctionnent proposé par le groupe IntServ repose sur quatre fonctions principales suivantes : l'ordonnancement des paquets, le contrôle d'admission, la classification des paquets et la signalisation.

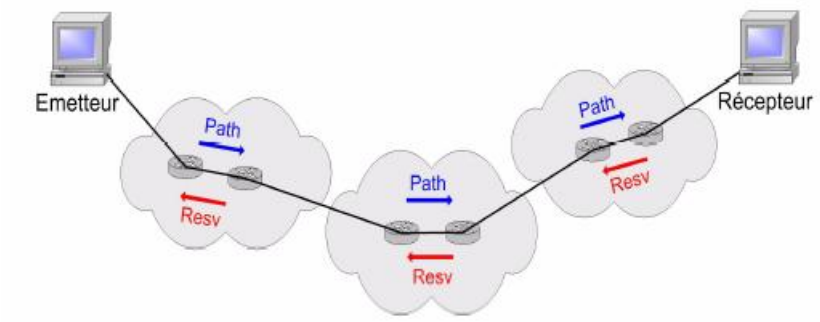


Figure 2.20: Modèle de service Intégré

Problèmes associées à IntServ Toutefois, le protocole IntServ présente plusieurs inconvénients :

- tous les éléments du réseau (routeurs, hôtes,. . .) doivent supporter le protocole RSVP pour fournir la QoS. RSVP nécessite le maintien des états du flux sur noeud chaque tout au long du chemin reliant l'émetteur au récepteur. Si le nombre de client augmente, le nombre d'états augmente en conséquence, et trafic devient saturant lorsque le rafraîchissement des états entre routeurs devient plus important ; ainsi le contrôle d'admission devient de plus en plus difficile à faire, ce qui dégrade les performances du système dans son ensemble.
- IntServ ignore les réseaux multi-domaines comme l'internet. L'administration par domaines indépendants impose des gestions locales pour assurer l'autonomie du système, chose qui est incompatible avec l'architecture pour laquelle ce protocole a été créé.

2.5.5.2 Service Différencié

Le modèle DiffServ a été conçu pour remédier aux limites d'IntServ. Il est définit pour assurer la QoS par classe de trafic IP en repoussant le traitement vers les routeurs de bordure des domaines pour ne pas surcharger le réseau de coeur. Les classes DiffServ sont associées aux groupes de paquets IP (agrégat) nécessitant un traitement semblable pour le marquage des paquets. Cette solution évite les problèmes de passage à l'échelle déjà rencontrés pour IntServ. En effet, les classes DiffServ sont identifiées par le marquage de l'entête IP du champ DSCP (DiffServ Code Point) qui remplace le champ TOS dans l'entête de l'IPv4 ou la classe de trafic de l'IPv6. Six bits sont choisis sur les 8 utilisés (les autres 2 bits sont réservés) pour cette raison. Un exemple de routeur DiffServ est décrit dans la Figure 2.21. Il doit être capable de classifier le trafic et de faire une politique de contrôle de flux rentrants. Les paquets arrivant à l'entrée du routeur doivent être classifiés pour leur attribuer le comportement adéquat en se basant sur le marquage DSCP. Le conditionnement le trafic réalise le comptage (meter), la mise en forme (shaping), le polissage (policing) pour s'assurer que le trafic entrant dans le domaine DiffServ est conforme à une politique de service bien connue.

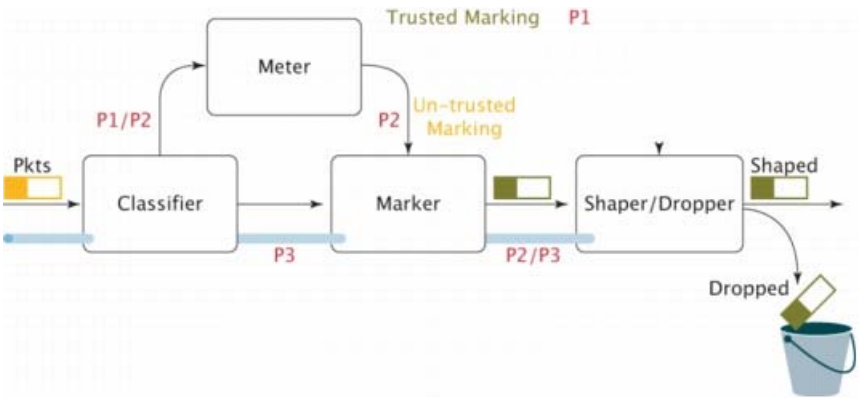


Figure 2.21: le modèle DiffServ

La Figure 2.22 décrit l'architecture DiffServ qui se base sur les concepts de domaine DiffServ, de contrat de service (SLA), de comportement, et de gestionnaire de bande passante :

1. Domaine DiffServ : un domaine DiffServ est une zone administrative avec un ensemble commun de politiques d'approvisionnement de QoS et de définitions de PHB.

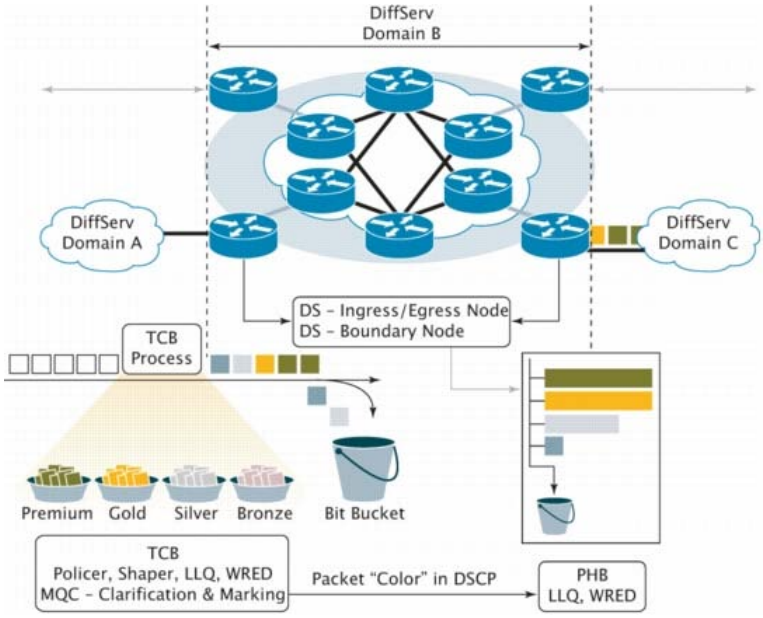


Figure 2.22: Comportement ou PHB dans le modèle DiffServ

2. Notion de comportement (PHB) : DiffServ permet de distinguer différents types de classes de trafic. Chacune de ces classes est traitée de manière différente sur les routeurs DiffServ. Cette politique de service est réalisée selon un comportement bien spécifique appelé PHB (Per-Hop Behaviour) où les paquets appartenant à un certain type de trafic sont routés selon une politique de routage bien spécifique à chaque noeud. Plusieurs PHB peuvent être définis pour répondre aux différents types de classes de service ayant des besoins

spécifiques de la QoS. Les services différenciés disposent aussi de deux PHB définis par les classes suivantes :

- service garanti (Expected Forwarding ou EF) : ou premium service [71] (RFC 2598) ; il a pour but de garantir une bande passante avec des taux de perte, de délai et de gigue faible ;
- service contrôlé (Assured Forwarding ou AF) [59] : regroupant plusieurs PHB garantissant un acheminement de paquets IP avec une haute probabilité ; Cette famille de PHB est scindée en 4 classes garantissant de fournir une bande passante et un délai minimum, chaque classe comprenant 3 niveaux de priorité (Drop Precedence).

3. Contrat de service (SLA) : Pour déterminer le comportement adéquat que les paquets doivent recevoir, une forme de négociation de service doit être réalisée avant la retransmission des paquets sur la sortie du routeur DiffServ. L'architecture DiffServ définit un contrat de service ou SLA (Service Level Agreement) entre le client et le fournisseur de service qui spécifie la politique de traitement que les paquets de ce client doivent recevoir. Toutefois, le SLA qui est un document informel, ne peut pas être utilisé lors de l'implémentation de la politique de service. Le SLS (Service Level Specification) contient exclusivement les détails techniques spécifié par le SLA (traduction du contenu de SLA en information nécessaires à la configuration des noeuds du réseau), comportant un ensemble de paramètres et de leurs valeurs, qui définissent le service offert à un flux de trafic par un domaine DiffServ (spécifier comment le trafic sera traité, le comportement d'expédition à la sortie du routeur,...). Une fois le SLS est accepté par le domaine, il doit s'engager à respecter la garanti du service pour le trafic spécifié dans le SLS durant la durée de l'accord (le SLS n'est pas une réservation, mais plutôt un engagement à autoriser des réserves sur la base des conditions énoncées dans le SLS).

PHB	Signification	Comportement	DSCP	
			Binaire	Décimal
EF	PHB EF	Service premium	101110	46
AF11	Groupe AF1	Préférence de suppression basse	001010	10
AF12		Préférence de suppression moyenne	001100	12
AF13		Préférence de suppression élevée	001110	14
AF21	Groupe AF2	Préférence de suppression basse	010010	18
AF22		Préférence de suppression moyenne	010100	20
AF23		Préférence de suppression élevée	010110	22
AF31	Groupe AF3	Préférence de suppression basse	011010	26
AF32		Préférence de suppression moyenne	011100	28
AF33		Préférence de suppression élevée	011110	30
AF41	Groupe AF4	Préférence de suppression basse	100010	34
AF42		Préférence de suppression moyenne	100100	36
AF43		Préférence de suppression élevée	100110	38

Table 2.2: Les PHB normalisés et leurs valeurs DSCP associés

2.5.5.3 Notion de gestionnaire de bande passante (Bandwidth Broker)

Pour que les routeurs puissent accorder le traitement adéquat à chaque classe de trafic une entité logique dite gestionnaire de bande passante (Bandwidth Broker, ou BB) a été proposée

pour permettre la configuration automatisée des accords contractuels entre le fournisseur de service et le client. L'idée du BB a été proposée par l'IETF dans le RFC2638 [96] à la suite d'une constatation faite par le groupe QBone [106] et qui dit qu'il n'était pas pratique et même inefficient pour les utilisateurs de connaître la topologie de bout en bout du réseau et toutes les politiques de services pour lesquelles il faut marquer les paquets.
Le BB est complètement conscient de toutes les ressources de son domaine. Il doit aussi avoir accès aux tables de routage des routeurs pour s'assurer que les paquets sont bien acheminés vers leurs destinations. En plus de cette fonctionnalité qu'il assure vis-à-vis du routeur, le BB doit en outre être capable de négocier les paramètres de la QoS avec les autres BB adjacents dans les domaines voisins afin de gérer les flux en inter-domaine. Plusieurs types d'architectures de déploiement ont été proposés pour [29] les BBs pouvant être classifiées comme suit :

- une seule entité est en charge de la gestion du domaine
- un BB distribué entre plusieurs entités
- une structure hybride qui combinent les avantages des deux architectures précédentes.

L'architecture de BB proposée par le groupe QBone, décrit à la Figure 2.23, comportent les éléments suivants :

- Interface d'accès aux utilisateurs, administrateurs et aux applications
- module de contrôle d'admission
- composant de communication en intra et inter domaine
- base de données qui contient toutes les informations concernant la topologie du réseau, les contrats établis avec les clients et les domaines adjacents, les politiques, les ressources disponibles, les allocations actuelles, les informations relatives à l'authentification, l'autorisation, la comptabilité et la tarification.
- les informations liées au routage
- un système de gestion de politiques et du réseau

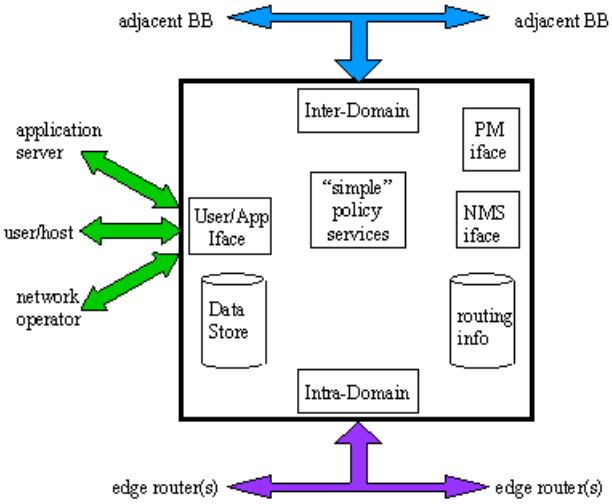


Figure 2.23: Architecture du Bandwidth Broker proposée par le groupe QBone

Problèmes ouvert associés à DiffServ Le modèle DiffServ était proposé pour surmonter le problème de passage à l'échelle rencontré dans le modèle IntServ. Même si DiffServ est le modèle le plus adapté pour internet, cependant il laisse ouvert un certain nombre de problèmes qu'il est nécessaire de résoudre pour obtenir la garantie de la QoS de bout en bout, à savoir :

- la complexité du provisionnement de la QoS qui nécessite le dimensionnement des ressources de chaque domaine par rapport au contrat de service établit par les clients dans les domaines adjacents.
- le contrôle d'admission des requêtes de services doit être propagé sur les liens inter-domaine afin d'évaluer la disponibilité du réseau (surdimensionnement en bande passante)
- la gestion d'internet multi-domaines et multi-technologies devient plus complexe à cause de l'hétérogénéité des domaines et du choix des classes de services entre domaines. Par conséquent, il devient nécessaire de mettre en place un modèle de signalisation de bout-en-bout pour assurer la disponibilité et la continuité du service.

Face à ces difficultés associées à IntServ et DiffServ, de nouvelles propositions visant le provisionnement de la QoS sur des architectures multi-domaines, notamment celles qui s'intéressent à la gestion de la QoS par contrôle de politique sont apparues. Nous décrivons ces contributions dans la suite de ce chapitre.

2.5.5.4 Provisionnement des ressources pour la QoS

Le provisionnement de service consiste à accorder le dimensionnement des ressources du réseau en fonction des contrats établis avec les divers clients et du trafic véhiculé. On considère un service bien provisionné si le réseau est capable d'assurer le transfert de la communication d'un point d'entrée (souscription de contrat) vers un (ou plusieurs) point(s) de sortie dans les conditions spécifiées dans la souscription tout en respectant les performances annoncées. Le provisionnement conduit à trois possibilités :

- Le surprovisionnement : une politique de surprovisionnement consiste à s'assurer que les capacités des liens sont supérieures au volume de trafic à écouler sur les liens. Dans le cas du mono-domaine, cette opération se traduit par un ajustement adopté par le fournisseur sur le surdimensionnement des ressources qui engendre la disponibilité des ressources. Pour le multi-domaine, le problème est plus complexe car il s'avère que le surdimensionnement n'est pas envisageable à cette échelle. Outre le fait que cette solution est très coûteuse pour les opérateurs, elle engendrerait une explosion des accords de pairs.
  Cette approche ignore non seulement la volonté d'optimiser les ressources réseaux mais aussi n'offre pas de solution pour le développement des services futurs. Ces remarques nous amènent à conclure que le surdimensionnement n'apporte pas une réponse aux problèmes de garantie de QoS. Avec les nouvelles applications et les nouveaux usages qui se développent, tels que les applications interactives, la visioconférence ou la voix sur IP, les très gros transferts de fichiers, il devient important de ne plus seulement provisionner les ressources, mais aussi d'adapter la manière dont les données sont transportées dans le réseau. Ceci est de plus vrai dans les réseaux sans fil ou ad-hoc déployés dans des situations d'urgence (médicales, militaires) où la qualité de service est primordiale.
- Dans le cas intra-domaine, le provisionnement consiste pour l'administrateur à définir l'infrastructure (les ressources) nécessaire pour la mise en oeuvre des contrats de services avec les clients. Ces clients peuvent être des particuliers, des entreprises/universités ou

bien d'autres fournisseurs. Nous précisons que nos travaux n'abordent pas cet aspect, le choix des mécanismes étant libre pour les administrateurs de domaines.
– La troisième découle de la nature multi-domaine de l'Internet. Il est à noter que les domaines traversés par une communication n'offrent ni les même capacités, ni les même performances. Il est donc nécessaire de connaître les caractéristiques des services fournis par les domaines impliqués le long du chemin de données pour faire un choix de concaténation de services (un par domaine) sur le chemin, suivi par les données afin de répondre aux besoins en QoS.

2.5.5.5 Contrôle d'admission

Le provisionnement ne suffit pas à garantir la disponibilité des ressources à tout instant sur le chemin de données. Pour garantir que le trafic reçoit le service adéquat un mécanisme de contrôle d'admission (CAC) de connexion doit être mis en place [5]. Les performances se détériorent rapidement si les demandes dépassent la capacité du réseau : pour éviter ce problème, le CAC assure qu'un nouveau flux de données admis ne provoque pas une dégradation de la QoS [138]. En d'autres termes, chaque nouveau flux susceptible de recevoir une QoS est soumis à un contrôle d'admission qui décide si celui-ci est accepté ou rejeté.
Ce mécanisme assure la limitation de la charge et la vérification que les ressources sont disponibles pour satisfaire les besoins d'un nouveaux flux sans pénaliser les communications existantes. Le CAC doit être complété par un mécanisme de protection du trafic conforme qui garantit qu'une nouvelle connexion ne pénalise les performances des communications existantes [137]. Le CAC concerne les équipements réseau impliqués sur le chemin de données (bande passante, buffers) et distingue deux aspects : le contrôle de la disponibilité à l'intérieur d'un domaine ainsi que sur les liens inter-domaines dans le cas du multi-domaine ou même en multicast [116].

2.5.6 Signalisation pour La QoS

La signalisation désigne l'ensemble des informations de services pour assurer le provisionnement des ressources nécessaires à l'établissement de la communication entre hôtes distants. Il existe plusieurs niveaux de signalisation sur l'Internet. Les mécanismes de signalisation basés sur des protocoles au niveau applicatif, des protocoles de signalisation au niveau réseau et la signalisation générique. Nous présentons ici ces trois types, tout en donnant plus de détails sur ceux que nous utilisons dans le cadre de cette thèse.

2.5.6.1 Signalisation au niveau applicatif

Au niveau applicatif, une signalisation est nécessaire entre les clients pour négocier une série de paramètres pour la mise en place d'une communication : les capacités des terminaux (en particulier les codecs), la localisation des terminaux (adresses IP, numéros de port), les protocoles transport utilisés, etc.

Le protocole H323 Le protocole H.323 définie un ensemble de protocoles de signalisation utilisé pour l'interactivité en temps-réel, notamment la voix, la vidéo, de l'image et de données sur les réseaux à commutation de paquets qui fait partie d'un ensemble de recommandations standardisées par l'IUT-T [70].

Le protocole SIP L'objet du protocole de signalisation SIP [58] (Session Initiation Protocol) est l'établissement, la modification, et la terminaison de sessions multimédia sur les réseaux IP. SIP peut fonctionner sur plusieurs protocoles de transport : UDP (le plus utilisé), TCP et même d'autres protocoles comme SCTP [127]. SIP a été conçu pour être évolutif : seules les fonctions de base sont obligatoires, il y a des extensions qui peuvent être supportées en option par les entités SIP. En effet, SIP fonctionne en mode client-serveur en échangeant des messages en format texte. Deux modes de communications sont possibles entre les différentes entités SIP.

- mode direct : les deux entités SIP sont des terminaux qui communiquent directement.
- mode indirect : des entités intermédiaires faisant partie du réseau relaient les messages échangés.

Le protocole SDP [56] (Session Description Protocol) a été publié par le groupe IETF sous RFC4566 [57]. SDP est une syntaxe de description de sessions media pour l'annonce, l'initialisation et l'invitation de sessions. Dans la cadre des travaux présentés dans ce mémoire, c'est le protocole SIP qui est utilisé, et il sera donné plus de détails sur le fonctionnement du protocole dans l'architecture développée.

2.5.6.2 Signalisation au niveau réseau

L'intérêt principal de la signalisation au niveau du réseau est de configurer les équipements réseaux pour assurer la QoS de bout en bout. Toutefois il existe deux catégories de protocole pour assurer la signalisation : la signalisation couplée au chemin de données "On-Path" et la signalisation découplée du chemin de données "Off-PATH". L'approche "On-Path" permet l'acheminement des informations de réservation et des données sur la même route. L'approche "Off-Path" ne nécessite pas le couplage entre le chemin de données et le chemin de réservation des ressources. Le protocole RSVP [84][14] issu des travaux du groupe IntServ est l'une des solutions qui sont proposées pour la signalisation couplée au chemin de données. Le protocole COPS représente une des solutions qui permettent la signalisation découplée du chemin de données.

RSVP Dans le modèle IntServ, la signalisation est basée sur le protocole RSVP (Resource ReSerVation Protocol). RSVP a été développé par l'IETF dans le RFC2205 [15] en parallèle à IntServ [136], et il est en conséquence particulièrement adapté aux services GS et CL. RSVP permet d'établir des réservations de ressources pour les flux individuels, en mode unidirectionnel ou bidirectionnel : les messages PATH sont envoyés des sources aux destinations suivant le chemin déterminé par le routage IP. Ces messages définissent le chemin qui va être utilisé pour le flux de données et établissent les réservations le long de ce chemin (ON-PATH). Ils permettent aux routeurs de mémoriser le ou les noeuds précédents pour renvoyer les messages RESV, annoncer leurs caractéristiques (via un élément ADSPEC) et informer les récepteurs des caractéristiques du flux et du service qui peut être réservé. Les messages RESV sont envoyés aux sources pour établir la réservation de ressources, empruntant exactement le chemin inverse du flux de données.

COPS COPS (Common Open Policy Service) est un protocole d'échange de politiques issu des travaux qui ont été fait dans le contexte de la réservation des ressources dans le groupe de travail sur la "RAP : Resource Allocation Protocol" de l'IETF [36]. COPS est basé sur une architecture client-serveur, pour le contrôle des réseaux par politiques [125]. Il est composé de deux entités de base décrites dans la Figure 2.24 : le PDP (Policy Decision Points) et le PEP (Policy Enforcement Points). Le PDP est une entité logique qui représente un équipement de management et de prise de décision dans le réseau. Le PEP est aussi une entité logique qui représente le point où les politiques seront appliquées. Le mode d'échange COPS est de type

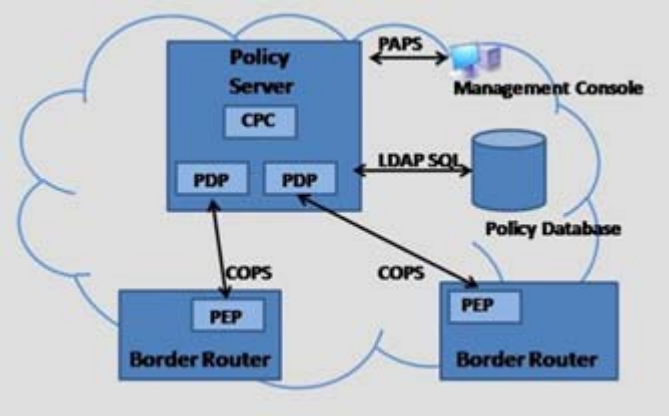


Figure 2.24: Le protocole d'échange de politiques COPS

requête-réponse en mode maître-esclave : le PDP est le maitre et le PEP est l'esclave. Le PEP interroge son PDP sur les actions à prendre suite à l'apparition d'un évènement (par exemple message de signalisation).

2.5.7 Contrôle par politiques

Le contrôle par politique est une approche qui a été standardisée par l'IETF pour permettre aux administrateurs de domaine de définir des règles de politiques pour gérer le comportement du réseau afin d'éviter toute manipulation directe sur les équipements du réseau. L'IETF a spécifié plusieurs modes de fonctionnement de contrôle par politique, parmi lesquels nous décrivons les suivants :

- **COPS-Outsourcing :** dans ce mode les évènements qui déclenchent les échanges COPS sont des messages arrivant au PEP. Le PDP est sollicité pour prendre la décision sur la politique à appliquer. Ce mode était défini pour les réseaux où la signalisation existe déjà, et a été spécifié par l'IETF dans la RFC2749 [60] 'COPS usage for RSVP'.
- **COPS-Provisioning :** dans ce mode de configuration COPS est intégré au réseau où les politiques sont transmises par le PDP au PEP pour le configurer et lui permettre d'appliquer la politique. COPS-Provisioning a été spécifié par l'IETF dans la RFC3084 [24] 'COPS usage for Policy Provisioning'.
- **COPS-SLS :** l'objet de cette extension est de permettre la négociation de niveau du service (SLS) en intra et inter-domaine en proposant un approvisionnement de QoS automatique et dynamique basé sur la négociation de niveau du service. Ce mode apparu en 2002 dans les travaux de Nguyen [130] sous le nom 'COPS usage for SLS Negotiation'.

- **COPS-DRA :** COPS-DRA combine les deux modes COPS-Outsourcing et COPS-Provisioning pour le support de l'allocation dynamique de ressources DiffServ. Ce mode qui a été décrit en 2001 par Salsano [114] peut être employé entre :(1) un routeur de périphérie agissant à la fois comme PEP et un Bandwidth Broker agissant comme un serveur ; (2) un client réseau agissant comme PEP et un point d'accès au réseau agissant comme PDP (routeur d'accès).

2.5.8 Intégration de COPS et SIP dans la réservation de ressources

Pour assurer la QoS au niveau réseau, différentes approches ont été proposées pour permettre l'intégration des mécanismes de réservation de ressources au sein de l'établissement de session. Dans cette partie nous décrivons les différentes techniques utilisées pour l'intégration du contrôle par politique avec l'établissement de sessions media via le protocole SIP.

- La première approche est définie par le groupe de travail SIP WG et présente l'intégration de la réservation des ressources pendant l'établissement de la session. Cette solution décrite dans [22], se base sur l'introduction de préconditions de type de réservation de ressources et suppose l'existence d'un mécanisme de type RSVP sur lequel elle s'appuie.
- La deuxième approche est proposée pour permettre la communication inter-domaine basée sur SIP avec QoS. Dans cette architecture des proxis SIP des domaines sont utilisés pour la négociation de la QoS au niveau application. Du côté COPS, ce proxy joue le rôle d'un PEP pour les sessions de communication IP en utilisant SIP qui nécessite l'utilisation de l'extension de COPS dans le mode outsourcing dite 'COPS usage for SIP'. Le serveur de politique (PDP implémenté au sein d'un serveur AAA and Policy Server) interagit avec les autres entités selon plusieurs modes de fonctionnement décrits en 2001 dans les travaux de GROS [50].
- La troisième approche connue sous le nom "QoS Control by means of COPS to support SIP-based application" [115] et "SIP Extensions For QoS support in DiffServ Networks" [133] [47] est une proposition qui vise à étendre SIP pour l'établissement de session et l'intégration dans les réseaux à QoS basés sur les architectures DiffServ s'appuyant sur l'extension COPS-DRA. L'avantage de cette solution est qu'elle préserve la compatibilité avec les applications SIP existantes. La solution introduit un serveur proxy-SIP enrichi pour supporter la QoS appelé Q-SIP Server qui combine les fonctions classiques des proxies SIP et des fonctions de réservation de ressources.
- La quatrième approche consiste à étendre SIP pour l'intégration d'un contrôle d'admission et l'autorisation de média au sein de l'établissement de session. Connu sous le nom "SIP Extensions for Media Autorisation" [38], elle se base sur le concept de Proxy SIP utilisé au sein des domaines administratifs ayant des accords de politiques.
- Le draft "Framework for session setup with media authorization" [54] décrit les mécanismes pour la création de liens nécessaires à la vérification et l'utilisation de ressources QoS conformément à la demande établie et autorisée préalablement pour la session. Le draft décrit les mécanismes qui permettent la coordination des actions de contrôle de politique de QoS pour un flux media afin d'accéder au service à l'intérieur des limites du profil de service établi par l'hôte demandeur.

2.5.9 Signalisation générique NSIS

Vu les difficultés rencontrées avec RSVP (signalisation, mobilité, etc.), le groupe de travail NSIS de l'IETF s'est focalisé sur le développement d'une nouvelle suite de protocoles de signalisation pour la manipulation de l'état le long du chemin de données. Suite à ces efforts est né le protocole de signalisation Next Steps in Signaling (NSIS) [55] comme étant un protocole générique, dans la mesure où il permet a priori tout échange de signalisation entre pairs NSIS. NSIS est scindé en deux couches, NSIS Transport Layer Protocol (NTLP) [119] et divers NSIS Signaling Layer Protocols (NSLP).

2.5.10 Architecture des réseaux pour la gestion de la QoS sur Internet

Du fait de la convergence vers les réseaux tout IP qui permet l'interfonctionnement d'infrastructures hétérogènes, l'objectif est de fournir des services de QoS pour faire évoluer l'architecture de l'Internet. Dans cette optique, plusieurs projets ont été menés dans les travaux de recherches pour trouver des réponses aux problématiques de provisionnement des ressources, de contrôle d'admission et de la signalisation. Par exemple, Le groupe Internet2 QoS a proposé une solution pour le test et le déploient de mécanismes de QoS scalables supportant des applications et technologies avancées dans un système de différentiation de trafic sans rendre complexe l'Internet. Également, d'autres projets Européens ont été menés pour pousser l'Internet actuel vers un réseau nouvelle génération qui offrira des services évolués avec un niveau élevé de qualité de service, de performance et de sécurité. Dans ce contexte, nous proposons un aperçu de certains projets de recherche qui ont eu pour but d'élaborer des propositions sur la QoS, principalement dans un environnement DiffServ.

2.5.10.1 Le IP Multimedia Subsystem (IMS)

L'IMS [107] est une architecture réseau standardisée conçue pour permettre de délivrer des services multimédia par le biais de réseaux fixes et mobiles (services de transfert de voix, de messagerie instantanée, de jeux interactifs, de partage de contenus, etc.). L'IMS repose sur la technologie VoIP basée sur une implémentation 3GPP standardisée de SIP fonctionnant sur un protocole standard IP pour supporter une sur-couche réseau indépendante du type d'accès (UMTS, WLAN, GPRS, DSL, etc.). Dans cette optique, l'IMS gère à la fois les flux voix et données [78]. Les travaux communs avec ETSI-TISPAN [43][39] ont conduit à l'ouverture de l'IMS pour fournir la standardisation des réseaux de nouvelle génération (NGN) [40] en rajoutant des composants spécifiques pour supporter l'interopérabilité entre différents réseaux et leur interfonctionnement avec les réseaux et services téléphoniques existants [42][41]. Les architectures NGN fournissent un cadre pour la mise en oeuvre de services évolués avec des caractéristiques innovantes : QoS, souplesse de la taxation, intégration et convergence fixe/mobile.

2.5.10.2 QBone

Le projet QBone [106] (qbone.internet2.edu), lancé en 1999 aux USA est un pionnier qui avait pour objectif le déploiement et le test des mécanismes de QoS évolutifs dans un environnement Internet multi-domaine. QBone visait la mise en oeuvre d'un service Premium (équivalent à une ligne louée) à l'échelle de l'Internet. Le projet n'a pas abouti aux résultats initialement ciblés

et a ensuite visé un objectif de service moins performant ne nécessitant pas de mécanismes complexes.

2.5.10.3 AQUILA

Les objectifs principaux du projet *AQUILA* [49] visaient la conception et l'implémentation d'une architecture de QoS dans un environnement IP. La solution AQUILA est basée sur la proposition d'un niveau de contrôle des ressources en privilégiant une architecture de type DiffServ. De plus, AQUILA propose l'utilisation d'une boite à outils pour faciliter la mise en place de la QoS par les utilisateurs finaux. Pour répondre au problème de provisionnement, AQUILA adopte une approche orientée agent qui configure automatiquement les équipements à partir d'une base de données. AQUILA propose une gestion globale d'un domaine via un agent qui a pour rôle de surveiller et contrôler les ressources. Pour le contrôle d'admission, AQUILA propose une vue qui implique principalement les routeurs de bordure d'un domaine, le routeur de coeur ne gardant que des états par agrégat.

2.5.10.4 TEQUILA

Le projet *TEQUILA* [48] visait à obtenir des garanties quantitatives de QoS en étudiant la définition des services et des outils d'ingénierie de trafic. Les axes principaux de recherche de Tequila comprenaient la spécification des SLS et des schémas d'ingénierie de trafic intra et inter-domaine. Le provisionnement des ressources dans TEQUILA repose sur une approche basée sur des politiques qui visent à l'ingénierie de trafic pour gérer la mise en place des solutions Diff-Serv sur MPLS. Concernant le contrôle d'admission, TEQUILA adopte une approche orientée mesure, basée sur une estimation et une prédiction de la charge qui conditionne l'acceptation d'une nouvelle requête.

2.5.10.5 NETQOS

Le projet *NETQOS* [52] avait pour objectif le développement d'une solution automatique de gestion de la QoS basée sur des politiques qui visent un environnement hétérogène mono-domaine. Le but est de proposer une architecture autonome, auto-configurable qui repose sur des politiques de gestion, capables de répondre aux besoins dynamiques de QoS des utilisateurs qui peuvent évoluer pendant une communication.

2.5.10.6 EuQoS

Sur la base de plusieurs contributions, le projet européen *EuQoS* [16], dans le cadre duquel ont été menés les travaux décrits dans ce mémoire, a proposé, développé et testé une architecture globale de gestion de la QoS de bout-en-bout dans un environnent Internet multi-domaine hétérogène. EuQoS concentre l'effort sur la signalisation intra et inter-domaine, et sur le contrôle d'admission spécifique aux différentes technologies sous-adjacentes. L'architecture d'EuQoS fait partie de notre proposition pour le support à la mise en réseaux distant de la simulation distribuée ; pour cela, nous détaillerons l'architecture plus loin dans cette thèse.

2.6 Conclusion

Les travaux décris dans cette thèse s'articulent autour des architectures de communication en réseaux locaux et réseaux distants pour les applications de simulation distribuée interactive. Nous avons traité alors les aspects gestion de QoS dans le contexte local et distant et des solutions permettant de mettre en place des mécanismes de gestion de la QoS, comprenant la signalisation, l'approvisionnement, le contrôle d'admission et la réservation des ressources réseau. La première solution possible, très utilisée, consisterait à enrichir PlatSim par HLA pour combler les manques de QoS dont souffre l'architecture. De plus, l'utilisation de DDS pour l'interconnexion des simulateurs distribués semble intéressante surtout vue la simplicité de son implémentation et ses performances de communication déjà prouvées sur des systèmes complexes. DDS a été déployée dans de nombreux systèmes, pour satisfaire les contraintes de systèmes temps-réel critiques, comme le transport aérien, la finance, la défense et les télécommunications. Pour cela, nous étudierons dans le prochain chapitre les deux solutions, et évaluerons les performances de chacune d'entre elles afin de choisir celle qui conviendra dans le reste de nous travaux.
Dans notre deuxième contribution, nous proposons un modèle de navigation à l'estime (Dead Reckoning) pour l'anticipation du comportement des entités simulées. Ensuite, nous proposons, pour notre troisième contribution, deux approches pour la gestion du trafic et l'optimisation de la bande passante pour les applications DDS. Nous n'adressons pas les applications HLA dans les réseaux grande distance, parce que ce standard est bien mature et plusieurs contributions ont été apportés dans ce contexte, et même adoptés par l'industrie.
Notre dernière contribution propose d'étendre l'approche précédente aux réseaux grande distance. De ce fait, une approche de gestion de la QoS dans les réseaux étendus est nécessaire et nous proposons deux techniques :

- SIP/DDS qui vise à mettre en place une architecture de communication basée sur le protocole SIP pour la gestion de QoS DDS sur des réseaux à grande distance.
- EuQoS/DDS dans le cadre du projet EuQoS, en particulier sur l'approche de signalisation et gestion de la QoS. Nous avons pu modifier des briques logicielles qui ont été proposées dans le cadre d'EuQoS, et nous les avons adaptées à la plateforme de simulation PlatSim_QoS que nous décrivons dans le prochain chapitre.

Chapitre 3

Interconnexion sur des réseaux locaux

3.1 Introduction

Ce chapitre décrit la problématique associée à la mise en réseau de la simulation distribuée réalisée en réseau local dans le cadre du projet PlatSim qui se base sur les services offert par les deux d'architecture middleware HLA et DDS. L'établissement du lien entre la spécification fonctionnelle et formelle des flux applicatifs, la correspondance (mapping) entre ces besoins et les services du middleware et la description du modèle du trafic appropriés feront l'objectif de ce chapitre.
L'objectif de ce chapitre est donc de décrire dans la section 3.2 l'architecture existante de simulation distribuée et les principales contraintes qui devront être supportées par le projet PlatSim. Ensuite, nous proposons à la section 3.3, une architecture de communication à base de HLA et nous montrons comment il est possible de réaliser une application coopérative synchronisée pour combler le manque de mécanismes de QoS de HLA.
Puis, nous décrirons dans la section 3.4, une deuxième architecture de communication basée sur le middleware DDS et nous montrerons qu'il est possible d'utiliser les mécanismes de gestion de la QoS DDS pour réaliser cette application sans avoir recours à la mise en place de mécanismes spécifiques pour gérer le problème de désynchronisme. La Section 3.5 focalise sur les différents scénarios de tests et les résultats obtenus avec les deux solutions. Enfin, quelques conclusions seront données à la fin de ce chapitre.

3.2 Le projet PlatSim

Le projet PLATSIM, consistait à développer une plateforme qui permette de mettre des simulateurs terrestres en réseau et de partager le même champ d'opérations virtuel (par exemple une ville, un aéroport, un abord de stade, etc.) et ainsi répondre à une demande de formation collective dans un environnement virtuel le plus proche possible de la réalité. Le partenariat est constitué de deux laboratoires publics (LAAS-CNRS à Toulouse et LISYC-CNRS à Brest), de 2 PME (DIXID à Lannion, SIRADEL à Rennes), 1 groupe leader du projet (ECA Faros à Lannion) et de la Gendarmerie Nationale Française.

3.2.1 Description fonctionnelle

Pour mieux comprendre l'architecture de la plateforme, nous présenterons un exemple d'utilisation au travers d'un exercice d'entraînement pour un groupe de policiers. L'exercice consiste à simuler une situation de cortège présidentiel sur une autoroute ; dans ce cas, tous les véhicules de police sont remplacés par des simulateurs de véhicules et la voiture du président par un autre simulateur (soit np simulateurs réels, environ 20). Dans l'exercice de simulation, il y a aussi des véhicules automates, soit na (une valeur inférieure à 50 unités est considérée) gérés par un serveur de simulation.
La Figure 3.1 décrit la plateforme matérielle visée par le projet PlatSim. En plus des simulateurs de véhicule, elle comporte un serveur de simulation dont la principale vocation est de constituer une vue globale de l'ensemble des simulateurs et de l'environnement simulé dans lequel chacun évolue. Cette vue est exploitée afin de ne relayer que les informations les plus pertinentes à chaque simulateur. Elle est également utilisée pour des fins pédagogiques.
La plateforme intègre également des postes instructeurs chargés de suivre un ensemble d'élèves (placés sur les simulateurs). Ce suivi est assuré conjointement par la réception d'une vidéo de chaque élève et la réception des interactions vocales impliquant ces élèves. L'instructeur peut également exploiter ces canaux audio afin d'interagir directement avec ses élèves, et la réception des données de simulation et/ou des données pédagogiques émanant du serveur de simulation.

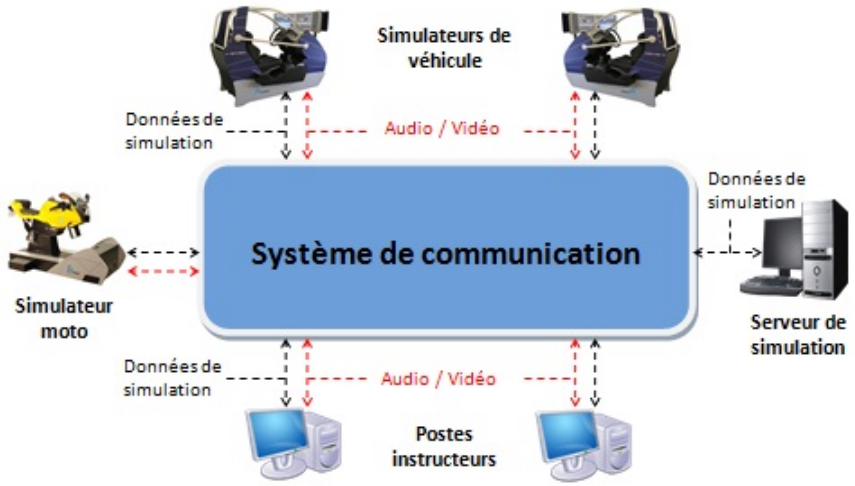


Figure 3.1: L'architecture fonctionnelle globale de Platsim

3.2.2 Description des données échangées

La Figure 3.2 décrit les quatre types de flux de données échangés entre les différents éléments de la plateforme PlatSim : des flux de données de simulation, des flux de données pédagogiques, des flux audio et des flux vidéo.

Flux de données de simulation : Ces flux regroupent toutes les données de simulation qui sont émises des simulateurs (les PCs centraux des simulateurs) au serveur de simulation et

Nom du flux	Structures de données	Nombre de flux	direction	Service	Profil	BP moyenne Requise (bps)	Exigences QoS
DS_VS_à_SS	PhotovhleSimule	np	Si -> SS	UDP (diffusion)	Périodique ($1/f$)	$1936 \times f$ ≈ 116 kbps (f=60hz)	- petite tolérance perte -délai < $1/f$
DS_SS_à_VS	*PhotoScene, PhotoObjet3D PhotoRoue3D*	1	SS->Si	UDP (diffusion)	Régulier: chaque $1/f$, rafale de messages applicatifs de 512 octets	$BP_{DS\_SS\_à\_VS}$ ≈ 5Mbps (n=54, m=216, f=60Hz)	- petite tolérance perte délai < $1/f$
DS_SS_à_PS	*PhotoScene PhotoObjet3D PhotoRoue3D*	1	SS->PI	UDP (diffusion)	Régulier: chaque $1/f$, rafale de messages applicatifs de 512 octets		-petite tolérance perte -délai < $1/f$
DS_PI_à_SS	N.D.	k	PI->SS	N.D.	apériodique	N.D.	-pas de perte -tolérance moyenne au délai (1s)
DP_SS_à_PI	N.D.	> k	SS->PI	N.D.	apériodique	N.D.	- pas de perte - délai: fonction du type d'erreur
Audio	-	np	Si <-> PI	dédié	-	-	-
Vidéo	-	np	Si <-> PI	dédié	-	-	-

DS : données de simulation ; DP : données pédagogiques ; Si : Simulateur physique ; SS : Serveur de Simulation, PI: Poste Instructeur; N.D. : non défini

La bande passante requise est calculé pour une architecture à 32 bits. $BP_{DS\_SS\_à\_VS} \cong \left(\left\lceil \frac{376+48\times n+30\times m}{512} \right\rceil\right) \times 4464 \times f$

Figure 3.2: Flux de données PlatSim

celles qui sont émises par le serveur de simulation à destination des simulateurs et des postes instructeurs. Elles permettent de décrire l'état dans lequel se trouvent les simulateurs (ou des éléments des simulateurs, e.g. roues, etc.) à un instant donnée. Ces flux incluent également les données de contrôle de la simulation qui sont envoyés par les postes instructeurs à destination du serveur de simulation afin de paramétrer en-ligne (durant la simulation) l'exercice (climat, position des véhicules simulés, contrôle du trafic routier, rajout d'obstacles, etc.).
Pour la suite, nous considérons une plateforme PlatSim constituée des éléments suivants :

- 1 serveur de simulation ;
- np véhicules simulés impliqués dans l'exercice ; une valeur np inférieure à 16 unités semble répondre aux besoins de ECA-Faros ;
- na véhicules automates gérés par le serveur de simulation. Sous les conditions liées au point précédent, une valeur de na inférieure à 50 unités est considérée par ECA-Faros ;
- k le nombre de postes instructeurs.

Nous supposons que les simulateurs physiques, automates et postes instructeurs sont numérotés et notons par Sp (resp. Sa) l'ensemble des simulateurs physiques (resp. automates). np =Cardinal(Sp) et na=cardinal(Sa). Nous noterons par mi le nombre de roues du simulateur i (mi=2 (resp. 4) si le simulateur est une moto (resp. auto)). Nous noterons par n le nombre de simulateurs impliqués dans l'exercice ; $n = na + np$. Nous noterons par m le nombre total de roues présentes dans l'exercice ($m = \sum_{i \in S_p \cup S_a} m_i$). Enfin, nous noterons par f la fréquence avec laquelle sont cadencés les simulateurs. f est un paramètre des simulateurs qui peut être fixé (eg. $30Hz$, $40Hz$ ou $60Hz$). Le flux de données émis par le serveur de simulation est une

succession de messages concaténant à chaque fois :

- une instance de la structure PhotoScene (de taille 376 octets)
- n instances de PhotoObjet3D (une de taille 48 octets par véhicule participant à l'exercice)
- m instances de PhotoRoue3D (une de taille 30 octets par roue de véhicule), qui sont envoyés périodiquement à la fréquence f du simulateur en utilisant le service UDP en mode diffusion. Étant de grande taille, ce message est découpé en fragments de 512 octets. C'est donc une suite de fragments du message suscité qui est diffusée périodiquement.

La bande passante requise est calculé pour une architecture à 32 bits est calculée par la relation 3.1.

$$BP = \left(\left\lceil \frac{376 + 48 \times n + 30 \times m}{512} \right\rceil \times 4464 \times f \right) \tag{3.1}$$

Flux de données de contrôle : Pour ce qui concerne les données de contrôle de simulation (envoyées par le poste instructeur au serveur de simulation), le flux de données est de nature apériodique car déclenché à l'initiative de l'instructeur.

Flux de données de pédagogie : Les données pédagogiques correspondent à des erreurs individuelles ou collectives détectées par le serveur de simulation et transmises aux postes instructeurs concernés. Ces flux sont de nature apériodique et devraient contenir en plus de l'identificateur de l'erreur et de la liste des élèves concernés, des informations additionnelles déduites des informations d'état des simulateurs aidant l'instructeur à mieux juger l'erreur.

Flux multimedia : Les données multimedia sont composées de deux flux distincts : les flux audio, bidirectionnels, supportent des échanges vocaux entre chaque élève et son instructeur dont le but est de permettre l'échange de certaines consignes données par l'instructeur. Ces flux sont supposés être ponctuels à la demande de l'instructeur ; ils ne sont pas supposés durer pendant toute la simulation ; les flux Vidéo, unidirectionnels, allant de chaque poste élève vers le poste de son instructeur, transportent des vues vidéo du comportement des élèves. Ces flux ont une durée de vie égale à la durée de la simulation. Dans PlatSim, ces échanges se font sur un réseau dédié différent de celui emprunté par les données de simulation et pédagogiques.

3.2.3 Fourniture de la QoS pour PlatSim

L'approche préconisée pour la fourniture de la QoS dans la définition de l'architecture de PlatSim à QoS, ici Platsim_QoS, est montrée à la Figure 3.3. Elle consiste donc au découpage du trafic applicatif en plusieurs flux ayant des besoins et des exigences spécifiques en termes de QoS. Ces derniers sont obtenus par redécoupage des flux de données PlatSim en séparant les données qui sont réellement périodiques de celles qui ne le sont pas (apériodiques ou événementielles). Nous avons fait le choix de faire transiter sur le même réseau aussi bien les données de simulation et pédagogiques que les flux audio et vidéo. Ils résultent d'un compromis entre, d'une part, la réduction du volume de trafic qui est échangé entre les éléments de la plateforme, et d'autre part l'effort de développement pour intégrer cette recomposition dans les simulateurs.

3.2.4 Lien entre la QoS IP et les QoS applications :

Établir le lien entre l'application et la caractérisation de son trafic ne pourrait avoir lieu que si l'application fournit un ensemble de paramètres bien identifié et qui a une influence directe sur le trafic. Pour cela, afin de fournir la QoS requise par ces applications, nous avons utilisé les services définis par le groupe de travail DiffServ de l'IETF. Ce modèle est répandu d'une part

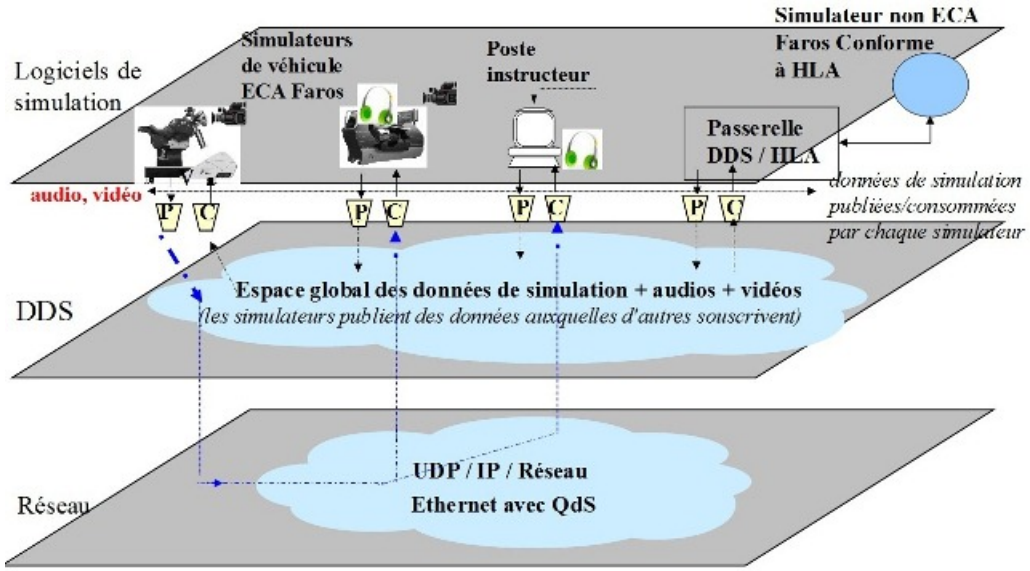


Figure 3.3: Architecture de la solution PlatSim à QoS

par sa simplicité à mettre en place et d'autre part parce qu'il permet de prendre en compte les besoins des applications distribuées dans les contextes des réseaux locaux et étendus.
Notre solution tient compte des services de transports déjà offerts par les middlewares et propose l'intégration de mécanisme de gestion et de fourniture de la QoS au niveau IP. Cette approche permet de rendre transparents les mécanismes de gestion de QoS à l'application et de masquer la complexité des mécanismes de communication au réseau sous-jacent. Les paquets sont marqués selon la classe de service qu'elle souhaite utiliser (champs 802.1p et DSCP), de manière à satisfaire son besoin en QoS en tenant en compte des caractéristiques du flux global injecté dans le réseau.
Dans ce chapitre l'accent est mis sur la classification des données au travers du middleware et partant de l'hypothèse que le réseau local est surdimensionné. Il ne sera donc pas traité de problématique de QoS de niveau 2.

3.3 Implémentation de la Simulation distribuée sur HLA

L'objectif de cette première partie consiste à proposer une architecture de communication satisfaisant les contraintes de QoS décrites dans le chapitre précédent et à la mettre en place sur le middleware HLA. Comme HLA ne propose aucun mécanisme de gestion de la QoS, nous avons alors renforcé les mécanismes natifs de ce middleware pour répondre à ces exigences.

3.3.1 Définition des flux applicatifs

Nous avons décrit dans la Section 3.2 des différents flux de simulation à manipuler par PlatSim_QoS. L'approche que nous préconisons consiste à faire le lien entre les paramètres généraux de l'application et les composants du trafic pour déterminer le trafic résultant sous forme d'éléments applicatifs. Les éléments applicatifs peuvent se rapporter à des flux autonomes de

l'application.

Objets	Attributs	Taille	Débit Moyen	Débit Crete	Transport	DSCP	Onep
VehiculeSimule	voitureVitesse (PS) voitureCle (PS) voitureFeux (PS)	128	a [1]	a	R	18	3
Obj3D	Obj3DAttrs (PS)	55	a	a	RO	18	3
PhoRo3D	PhotoRo3DAtt (PS)	6	a	a	RO	18	3
AudioFormat	Format (P) FormatLength(P)	4	a a	a a	R R	18 18	3 3

Table 3.1: Table des attributs de PlatSim QoS selon le modèle HLA OMT

Nous avons déjà décrit dans la Figure 2.2 (section 2.3) les attributs et paramètres qui caractérisent quelques éléments applicatifs de Platsim_QoS sur HLA. L'utilisation du fichier XML pour la spécification des différents attributs et paramètres respecte la règle 1 de HLA concernant les fédérations ([2]) et les fédérés ([3]), qui exige que toutes ces informations soient documentées en respectant la spécification. Nous décrirons ces éléments dans le Tableau 3.1 et le Tableau 3.2 pour montrer la signification des différents champs.
En effet, La colonne 1 décrit le nom de chaque élément applicatif qui correspond aux objets et aux interactions HLA. La colonne 2 inclut les noms des attributs des différents objets et les paramètres des différentes interactions. Un Fédéré peut soit produire (P), soit souscrire (S), soit produire et souscrire (PS) aux échantillons de ces attributs (resp. paramètres). Pour déterminer le flux global de l'application, chaque objet (resp. interaction) est caractérisé par la taille de chaque attribut (resp. paramètre), qui permet de déterminer la taille globale des données (en bits) générées par l'application. Cette information, présente dans la colonne 3, permet de déterminer le trafic global de PlatSim_QoS qui comporte 10 classes d'objets et 7 classes d'interactions ayant chacune plusieurs attributs et paramètres.

Interactions	paramètres	Taille	Débit Moyen	Débit Crete	Transport	DSCP	Onep
Observateur	obsTypeVue obsReculX	24	a	a	RO	46	4
Exo	ExoParam	08	a	a	RO	46	4
AudioPacket	Data	320	a	a	RO	34	6
VideoPacket	Data	512	a	a	RO	36	5

Table 3.2: Table des interactions de PlatSim à QoS selon le modèle HLA OMT

Les colonnes 4 et 5 montrent respectivement le débit moyen et le débit crête (en bit/s) nécessaire pour l'envoi du trafic global de l'objet en fonction de leur fréquence de production (il est aussi possible d'agréger plusieurs objets dans un même paquet et les envoyer sous forme d'une rafale à débit crête). Ces paquets vont être envoyés à travers les services de transport fournis par HLA. Ces services permettent deux types de transport : fiables et non fiable. Le

2. Federations shall have an HLA federation object model (FOM), documented in accordance with the HLA object model template (OMT)
3. Federates shall have an HLA simulation object model (SOM), documented in accordance with the HLA object model template (OMT).

transport fiable, décrit par "R" (HLAReliable) dans la colonne 6, utilise le mode de transport TCP, alors que le transport non fiable, décrit par "RO" (ReceiveOrder), utilise le protocole de transport UDP. Les données envoyées en mode non fiable sont des données périodiques, comme par exemples les données audio et vidéo. La mise à jour de leur état permet d'écraser les anciennes instances par les nouvelles valeurs. Les informations transmises en mode fiable, sont des données apériodiques. Elles contiennent les données de contrôle des simulateurs. Finalement, les deux dernières colonnes représentent respectivement le marquage des champs DSCP et 802.1p des paquets. Ces derniers champs ne sont pas spécifiés par le formalisme HLA-OMT (Object Management Template), mais ils nous serviront plus avant pour spécifier la QoS à mettre en place sur le switch en réseau local.

3.3.2 Définition des paramètres de la QoS

Nous avons spécifié dans la description de Platsim_QoS (Section 3.2) que la QoS au niveau réseau prendra en compte le modèle DiffServ. Les caractéristiques du modèle DiffServ sont définies en termes quantitatifs (débit, délai, gigue, perte) et termes de priorité d'accès aux ressources du réseau. Nous décrivons dans ce paragraphe la configuration de la QoS en fonction du flux applicatif de Platsim_QoS.
En effet, la taille d'un paquet est déterminée par la somme des tailles des objets et des interactions (que nous appelons ici unité de données). Par exemple, la taille de l'unité de données (colonne 3 du Tableau 3.2, interaction HLA comme élément applicatif) se traduit en débit moyen de l'unité. Après le découpage du trafic, nous avons obtenu une taille moyenne des paquets de 554 octets (notée p) envoyés à une fréquence de production de 10ms, le débit résultant (noté D) est alors de $443200 bits/s$.
Rappelons que les deux dernières colonnes (7 et 8) du Tableau 3.1 et du Tableau 3.2 décrivent les niveaux de priorités des objets et des interactions (champ DSCP et champ "User Priority", respectivement). Par exemple, si une application désire mettre en place une politique de QoS pour l'objet '**VehiculeSimule**' et pour ces attributs, le marquage des paquets est alors 18 pour le champ DSCP et 3 pour le champ User Priority. Si les différents objets nécessitent des traitements différents, il faut associer à chaque attribut les valeurs de priorité adéquates.
La mise en place de la QoS est faite par le commutateur, qui classifie ou reclassifie ces flux à l'aide du champ DSCP ou du champ CoS 802.1p. En l'entrée, le commutateur contrôle également le trafic pour déterminer si un paquet correspond ou non au profil défini, marque le trafic pour modifier l'étiquette de classification, accepte ou refuse le profil des paquets et les place dans une file d'attente selon leur classification.

3.3.3 Modélisation de l'application sous HLA

Les contraintes de QoS qui peuvent exister dans une application de simulation distribuée interactive, et qui devront être contrôlées pour permettre la manipulation d'une simulation cohérente, sont variables. Ces contraintes, qui sont essentiellement temporelles, sont relativement complexes à exprimer, et l'un des problèmes à résoudre consiste à pouvoir représenter de façon simple et complète les contraintes de synchronisation qui peuvent exister entre les différents flux de la simulation.

Pour pouvoir atteindre cet objectif avec les hypothèses fixées, il est nécessaire de tenir compte de la variabilité temporelle des temps de traitement (délai, gigue) et des propriétés intrinsèques des flux manipulés. Ainsi, pour représenter les propriétés de synchronisation des flux manipulés, il est essentiel de disposer d'un modèle permettant de spécifier les contraintes de synchronisation qui devront être satisfaites par l'application. Ce modèle formel est particulièrement intéressant, car d'une part il permet de spécifier sans ambiguïté, des scénarios de présentation multimédia, et d'autre part de les simuler et les valider. Ainsi, afin d'évaluer l'adéquation du modèle formel à la spécification des flux, il est nécessaire de le confronter à trois critères fondamentaux :

- le pouvoir d'expression du modèle : pouvoir de modéliser de façon complète la problématique de synchronisation de flux, en particulier exprimer les contraintes temporelles de synchronisation ;
- le pouvoir d'analyse : pouvoir de mise en oeuvre des techniques de vérification des propriétés du système ;
- le pouvoir de modélisation : pouvoir d'abstraire le système et de cacher les informations non pertinentes pour maîtriser sa complexité.

L'analyse du pouvoir d'expression et de modélisation des systèmes distribués interactifs milite en faveur du formalisme des réseaux de Petri à flux temporel, qui permettent d'expliciter de façon complète et précise les potentialités de synchronisation (et de désynchronisation) intra-flux et surtout inter-flux. Seuls les modèles temporels peuvent exprimer les contraintes d'ordre temporel des simulations et apportent par leur pouvoir d'expression, d'analyse et de modélisation des moyens pour effectuer une analyse quantitative et qualitative de ces contraintes.

3.3.3.1 Les réseaux de Petri à Flux Hiérarchiques (HTSPN)

Un HTSPN est un réseau de Petri étendu qui utilise des arcs temporels pour la modélisation du traitement de flux (communication, présentation, etc.) pour réaliser une spécification de façon descendante par raffinage successif. La gigue temporelle qui apparaît dans le modèle du système distribué interactif est modélisée par un arc contenant l'interval de validité temporelle (TVI). Ces arcs sont des tuples [x,n,y] où x, n et y sont respectivement les durées minimales, nominales et maximale admissibles du traitement correspondant. Ces durées sont modélisées au sein des couches du modèle HTSPN [1] illustré à la Figure 3.4. Il est composé par une couche de synchronisation de lien, une couche de synchronisation composite et une couche de synchronisation atomique :

- Les liens dans la couche de synchronisation de liens, représentent le niveau supérieur de la Figure 3.4. Ils sont modélisés par arcs temporels (L, t), où L est la place lien. Le TVI associé à chaque lien est géré selon une taxonomie de synchronisation de tir associés à un réseau de Petri Temporel à Flux (Time Stream Petri Net ou TSPN) [34] ;
- les composants de la couche de synchronisation composite se situent dans le niveau intermédiaire de la Figure 3.4. Ils fournissent un mécanisme hiérarchique et structuré basé sur la composition récursive des composants atomique et composite à travers la notion de sous-réseau. La couche de synchronisation composite permet de décrire les contraintes structurelles et temporelles de synchronisation inter-flux en utilisant les places de type composite qui sont équivalentes aux réseaux atomiques sous-jacents ;

1. Hierarchical Time Stream Petri Net

– la couche de synchronisation atomique, présentée dans la couche basse de la Figure 3.4, contient les composants atomiques. Chaque composant est modélisé dans le HTSPN par un arc avec un TVI et une place associée à une ressource atomique. Ainsi, la couche de synchronisation atomique permet de décrire les contraintes de synchronisation intra-flux au sein des composants atomiques.

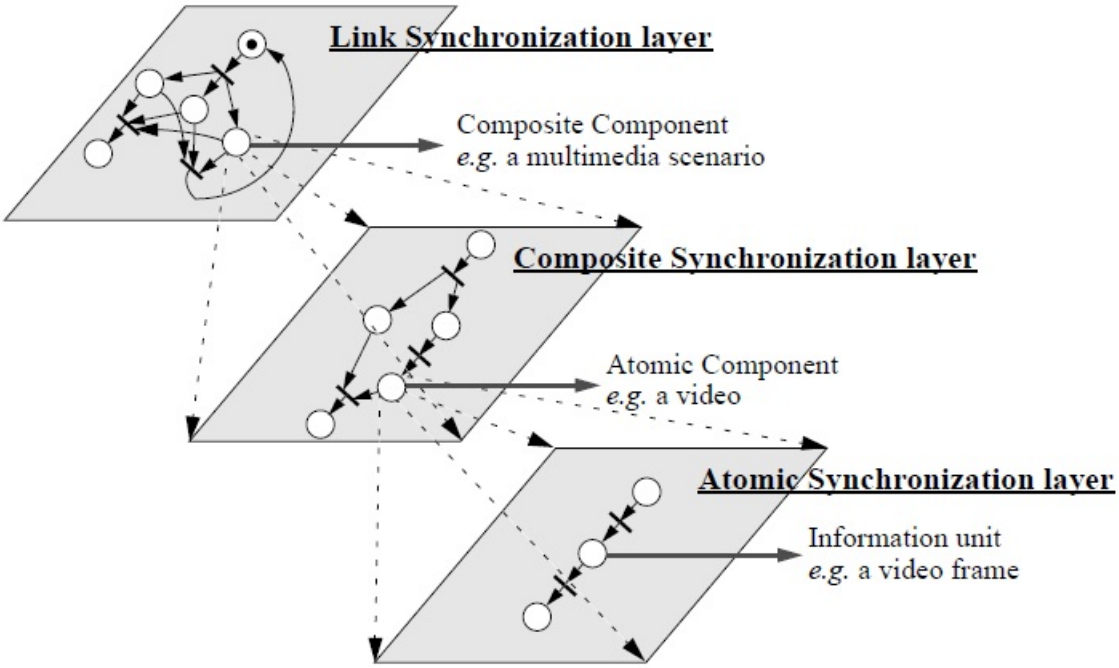


Figure 3.4: Réseau de Pétri Hiérarchisé à Flux Temporel

Le modèle de la Figure 3.4 permet d'associer des informations qualitatives et quantitatives aux unités temporelles des flux de l'application. Cette approche de modélisation va nous permettre à la fois d'exprimer en plus de l'indéterminisme temporel des unités de synchronisation dans un système de simulation distribuée faiblement synchrone, la variabilité temporelle admissible des objets de simulation.

3.3.3.2 Spécification formelle des contraintes de synchronisation sur les flux

Diaz et Senac [35] ont montré que le modèle TSPN, qu'on appelle ici modèle de présentation, permet de décrire des scénarios de présentation multimédia. Cependant, l'utilisation d'un seul modèle s'avère insuffisante dans notre cas pour décrire complètement l'ensemble de l'application. Pour cela, nous avons utilisé le modèle HTSPN à trois niveau hiérarchiques, que nous avons illustré à la Figure 3.5 pour décrire le modèle de présentation de haut niveau et les différents niveaux hiérarchiques : les TSPN de haut niveau modélisent le moteur de synchronisation de l'application au niveau de l'interface de présentation permettant de créer des points rendez-vous décalés assurant la re-synchronisation des flux. Ces point de synchronisation rendez-vous s'appliquent seulement sur l'application réceptrice ("application de synchronisation" dans la Figure 3.5).

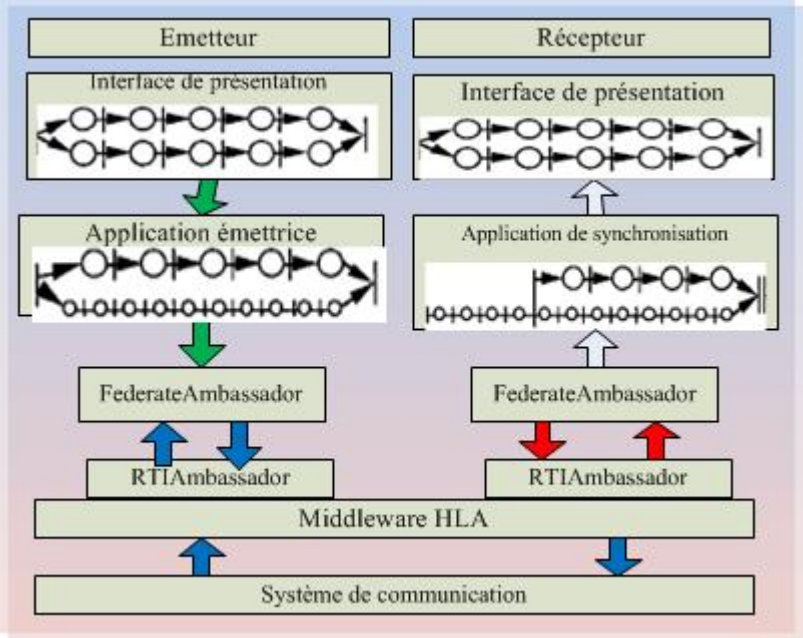


Figure 3.5: Modélisation Hiérarchique de l'application

En plus, le middleware HLA-RTI fournit des mécanismes de communication entre les fédérés permettant de transmettre des données en assurant une vision globale de la simulation [87] [122]. Pour cela, deux proposition ont été faites : la première considère la transmission du flux de simulation à travers le middleware HLA et l'utilisation d'un autre protocole comme RTP [118] pour envoyer le flux multimédia [88]. La deuxième solution consiste à transmettre le flux de simulation, l'audio et la vidéo à travers le middleware HLA-RTI [79] [139].
Nous avons alors retenu cette dernière approche qui a l'avantage de faciliter la cohérence entre les différents flux issues de l'application et de respecter les règles de HLA que nous avons introduit dans la section 2.3.1, en particulièrement la règle 3. Cette règle exige que la transmission de données entre fédérés doit passer obligatoirement à travers les services et les CallBack fournis par la couche "RTI" (During a federation execution, all exchange of FOM data among federates shall occur via the RTI). Sur la Figure 3.5, HLA-RTI reçoit les données sous forme d'interactions et d'objets HLA de la part de chaque **RTIAmbassador**, et les envoie vers chaque Fédéré via l'interface **FederateAmbassador**. Toutefois, cette solution ne suppose aucune garantie de QoS de la part du middleware. Pour cela, nous avons enrichi l'application pour améliorer la gestion de la synchronisation des flux applicatifs.

3.3.3.3 Modèle de Synchronisation avec les HTSPN sous HLA

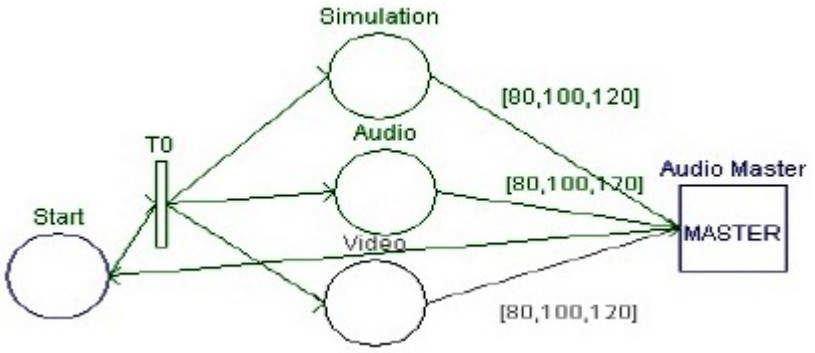
Le cas de la simulation étudié, plus simple que le cas Platsim, comporte trois flux : un flux audio, un flux vidéo et un flux de simulation interactive. La synchronisation inter-flux est relative à un ensemble de flux considérés et elle spécifie les dérives inter-flux admissibles. Ces contraintes de synchronisation devant être vérifiées par chaque flux sont les suivantes :

1. Les unités de synchronisations audio ont une durée nominale de 25ms et elles admettent une gigue maximale de ±5 ms. Ceci signifie que l'intervalle de validité temporelle de chaque unité de synchronisation audio est [20, 25, 30].
2. Les unités de synchronisation vidéo ont une durée nominale de 40 ms et admettent une gigue intra-flux de ±10 ms. L'intervalle de validité temporelle du flux vidéo est [30, 40, 50].
3. Les unités de synchronisation du flux de la simulation distribuée interactive ont une durée nominale de 20 ms et admettent une gigue de ±4 ms. L'intervalle de validité temporelle de ce flux est [16, 20, 24].

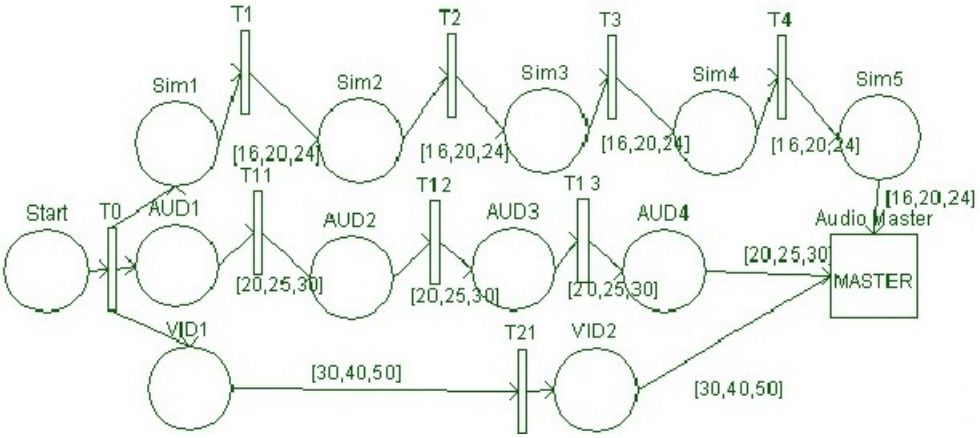
Les contraintes de synchronisation inter-flux qui doivent être vérifiées par l'ensemble des flux sont : (i) le flux vidéo peut être en avance/retard sur le flux audio de 15ms, (ii) et le flux de la simulation distribuée peut être en avance/retard sur l'ensemble des flux vidéo/audio de 4ms.
Pour pouvoir associer des intervalles de validité temporelle aux arcs des HTSPN, représentant les contraintes de synchronisation inter-flux, nous avons cherché la transition de rendez-vous (RDV) de synchronisation. Cette transition permettra de sélectionner un point de synchronisation : (1) qui définit une période adéquate représentative du fonctionnement de la simulation, et (2) qui définit et explicite les besoins de QoS de l'application interactive considérée.
La modélisation formelle de cette approche sera formée par trois niveaux hiérarchiques et 5 réseaux HTSPN. Nous avons présenté à la Figure 3.6 le modèle de synchronisation entre les trois flux. La simulation est faite par l'outil HTSPN Toolkit [110]. Les HTSPN considérés dans cette approche sont formés de trois composants :

Les liens : définissent à la fois des relations sémantiques entre composants multimédia et des capacités de navigation entre ces composants offertes à l'utilisateur. Ce niveau, illustré à la Figure 3.6a, représente la couche supérieure du modèle de synchronisation, et modélise le niveau de l'application (interface utilisateur).

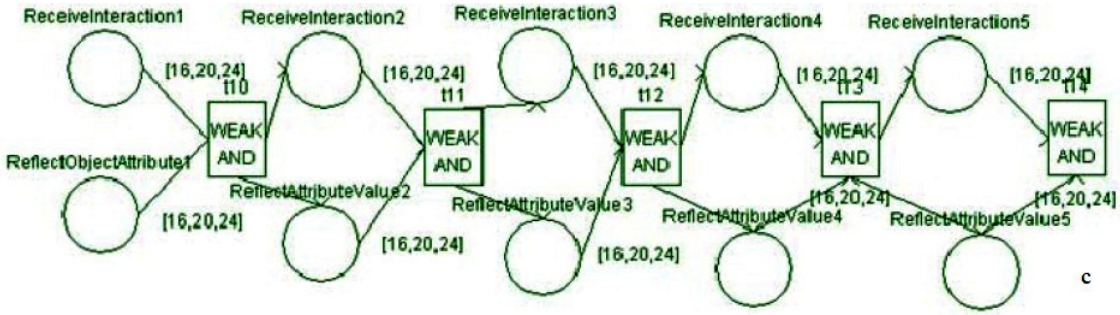
Les Composant composites : illustré à la Figure 3.6b, ce niveau spécifie les contraintes de synchronisation inter-flux entre le flux audio modélisé par les places Aud_i, le flux de la simulation interactive modélisé par les places Sim_i et le flux vidéo modélisé par les places VID_i. Ce réseau spécifie en particulier le contrôle qui doit être mis en oeuvre pour assurer la synchronisation entre ces trois flux et assurer que le flux vidéo n'est plus en retard de plus de 30ms par rapport aux autres flux. Ce contrôle doit être appliqué avec une granularité maximale de 20ms correspondant à deux unités de synchronisation vidéo, ou 4 unités de synchronisation audio et 5 unités de synchronisation du flux interactif.
En effet, la gigue admissible par unité de synchronisation vidéo est de 10ms, celle de la voix à 5ms et celle du flux interactif à 4ms, une dérive maximale de 30ms est autorisée pour le flux vidéo. Le premier point de synchronisation inter-flux de type "MASTER", avec le flux audio comme "MASTER", est placé au point de rendez-vous après 100ms au maximum. Cette synchronisation est susceptible d'induire une accélération du flux interactifs et des flux audio après 5 unités de synchronisation pour le premier et pour le second et peut éventuellement engendrer du retard pour le flux vidéo.



(a) couche lien



(b) couche composite



(c) couche atomique

Figure 3.6: Modélisation du schéma de synchronisation d'une application de simulation distribuée

Les Composants atomiques : ces composants sont modélisés par des TSPN dans la Figure 3.6c. Ils représentent chaque flux pris individuellement. Ils sont assujettis à des contraintes temporelles intrinsèques aux medias. Un élément atomique est modélisé par un arc avec un intervalle de validité temporelle auquel s'associe une place atomique associée à une ressource mono média. Les composants atomiques visent à modéliser les contraintes de synchronisation intra-flux.

Les places abstraites Sim_i sont spécifiées par le sous-réseau illustré à la Figure 3.6b. Ce HTSPN

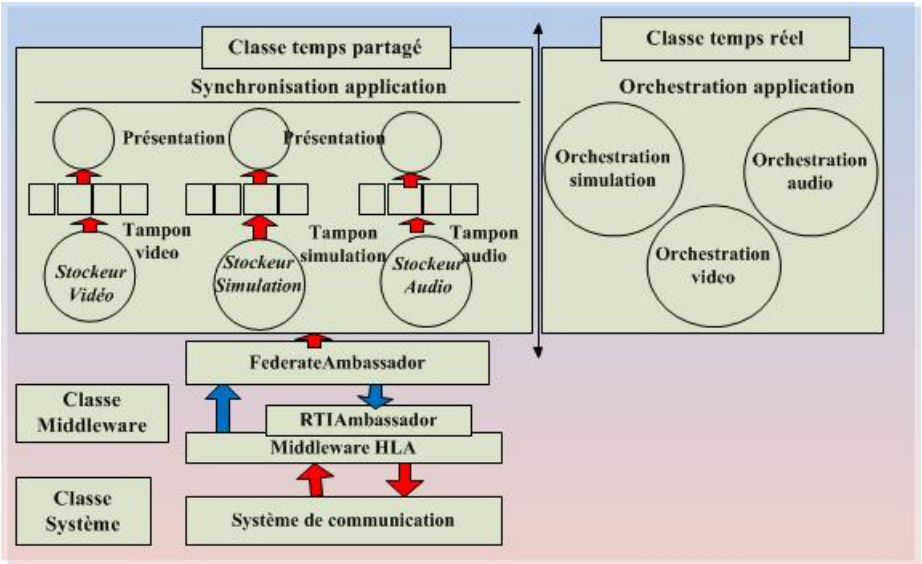


Figure 3.7: Architecture de synchronisation pour PlatSim sous HLA

explicite le contrôle de l'avance du flux interactif par rapport aux autres flux. Étant donnée la gigue des unités de synchronisation vidéo à 10ms et celle de flux audio à 5ms, au bout de 5 unités de synchronisation, le flux interactive se trouve en avance de 20ms sur les autres flux. Le contrôle de la dérive de ce flux doit être contrôlé par le middleware HLA-RTI de façon à garantir toutes les contraintes de synchronisation avec les autres flux. Cet auto contrôle devra être fait par les objets et les interactions définis dans le modèle objet de la simulation [67].

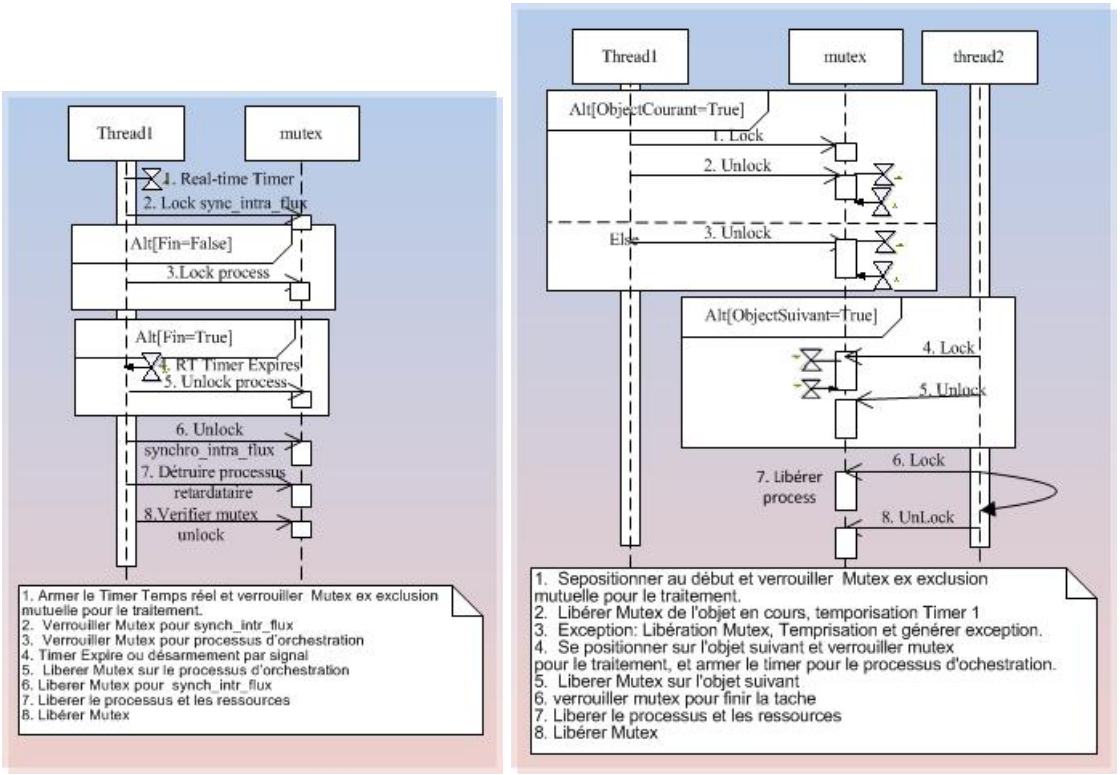
3.3.4 Implémentation des mécanismes de synchronisation

Le fait que les données échangées lors de l'exercice de simulation doivent passer par le middleware, par le noyau du système d'exploitation et par les cartes multimedia respectives, il existe inévitablement un risque de désynchronisation. Il n'est toutefois, pas possible de contrôler les données dans le noyau et dans les cartes multimedia (l'audio et la vidéo sont reçus sur des cartes différentes). La solution idéale serait de mettre en place des mécanismes de gestion de priorités au niveau du noyau. Toutefois, ceci n'est pas permis sur des systèmes d'exploitation Windows. Ce système étant asynchrone par nature, aucun temps de traitement maximum n'est garanti pour pouvoir assurer des présentations ne dépassant pas les bornes maximales.
La tâche de synchronisation se réduit alors à des opérations réalisées au niveau de l'application, en introduisant une classe de traitement avec les priorités proche du temps-réel, supérieures à la classe de priorité du système opératoire. Comme le montre la Figure 3.7, à cette tache de synchronisation nous associons des tâches moins prioritaires identique à la classe de priorité du middleware et du système opératoire pour la production des données.

3.3.4.1 Algorithme de synchronisation intra-flux

L'implémentation du schéma de synchronisation intra-flux met en concurrence des processus implémentés à l'aide de threads : un processus de présentation illustré à la Figure 3.8b et un pro-

cessus d'orchestration illustré à la Figure 3.8a. Toutefois, pour éviter des problèmes de conflits d'accès aux ressources, des mutex permettent d'accéder à ces variables en exclusion mutuelle. Cette approche permet de définir une borne minimale d'exécution que les différents processus



(a) **(b)**

Figure 3.8: Scénario d'orchestration et de présentation en intra flux

doivent respecter. Nous avons associé conjointement aux threads [72], des Timers Multimedia de Windows SDK [1] de façon que les durées d'exécution soient respectées, et contrôlées par des timeout qui doivent avoir une priorité haute pour interrompre l'exécution à l'épuisement du Timer.

3.3.4.2 Algorithme de synchronisation inter-flux

L'algorithme de synchronisation inter-flux est réalisé à l'aide de processus d'orchestrations des trois flux. Le principe consiste à l'implémentation d'un point de synchronisation rendez-vous entre les différents processus correspondant aux trois flux vidéo, audio et simulation comme le décrit la Figure 3.9.
Ce rendez-vous respectant les contraintes temporelles définies par la règle "Master" avec le flux "audio" comme flux maitre. Les "Timers" temps-réel sont réalisés à l'aide de l'horloge de haute résolution fournit dans Windows SDK, comme pour la synchronisation Intra-flux.

1. `http://msdn.microsoft.com/en-us/library/windows/desktop/dd743609(v=VS.85).aspx`

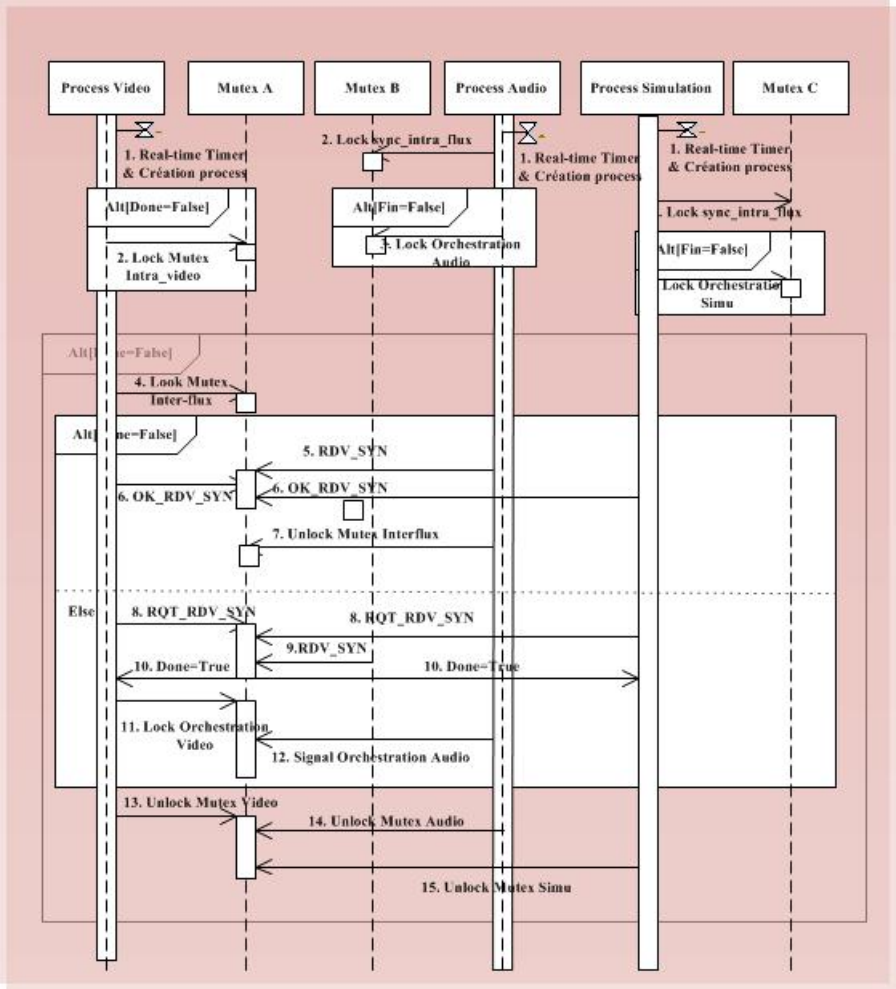


Figure 3.9: Scénario de synchronisation en inter flux

3.3.5 Les interfaces de programmation

Rappelons ici que, l'implémentation du schéma de synchronisation évoqué dans la Figure 3.6 du paragraphe précédent, est décrite par les places atomiques "$ReceiveInteraction_i$" et "$ReflectAttributeValue_i$", décrivant les composantes atomiques de la Figure 3.6c.
A partir des Tableaux 3.2 et 3.1, les paramètres que nous utilisons pour l'envoi des données multimédia sont décrites par "AudioPacket" et "VideoPacket". Les attributs qui contrôlent ces paramètres sont décrits par "AudiFormat", "VideoFormat" et "Operations". Ainsi, le moteur d'exécution "HLA-RTI" peut transmettre le flux audio, le vidéo et le flux interactif vers les autres simulations, tout en assurant le contrôle de chaque flux indépendamment l'un de l'autre.

L'utilisation des APIs de l'interface de spécification de HLA pour définir les scénarios de communication entre les fédérés permet de respecter la règle 4 de HLA pour les fédérations ([1]).

1. During a federation execution, federates shall interact with the run-time infrastructure (RTI) in accordance with the HLA interface specification

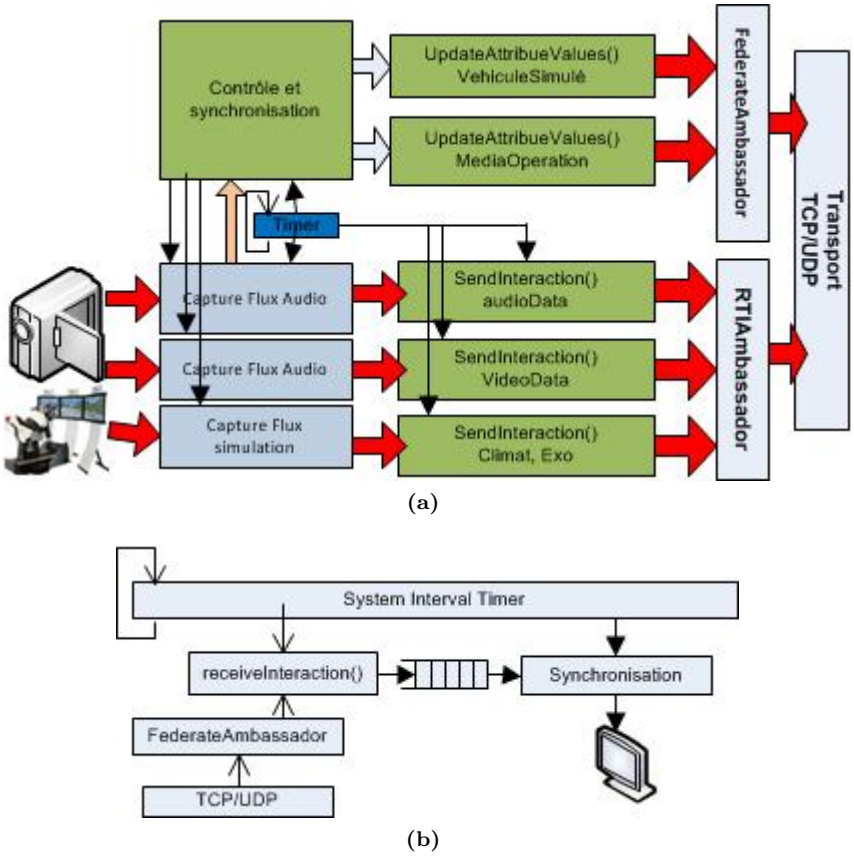


Figure 3.10: Transfert et de contrôle de données sous HLA

La Figure 3.10 illustre le scénario de transfert et de contrôle de données sous HLA. Selon l'interface de spécification de HLA (HLA-IS), **sendInteraction()** est une fonction de la classe RTIAmbassador, qu'un fédéré utilise lorsqu'il souhaite envoyer les données sur le réseau. La Figure 3.10a illustre un fédéré HLA produisant le flux qu'il souhaite envoyer vers les autres simulations au sein de la même fédération. La Figure 3.10a illustre ces mécanismes de contrôle et de synchronisation des différents flux. En effet, les APIs **UpdateAttributeValues()** et **reflectAttributeValue()** sont les APIs non seulement pour contrôler l'avance d'un flux par rapport aux autres mais aussi pour assurer la synchronisation des flux. La Figure 3.10b décrit un fédéré consommateur qui utilise la méthode **receiveInteraction()**, qui est une fonction de la classe FederateAmbassador, pour recevoir des informations du réseau. Le fédéré doit donc implémenter les fonctionnalités déclarées dans le FederateAmbassador.

3.3.5.1 Scénario de communication

Ce paragraphe illustre les différents scénarios de mise en réseau de deux simulateurs Platsim_QoS conforme avec HLA. En effet, un simulateur producteur initie la création de la fédération avec le serveur de simulation (le composant RTI). Une fois la fédération créée et que les simulateurs ont pu la rejoindre, le scénario de communication commence par l'échange de messages entre les participants. Enfin, lorsque l'un des simulateurs veut quitter la simulation,

il notifie le RTI et demande quitter la simulation.

Scénario d'initialisation de la Fédération :

La Figure 3.11 présente le scénario d'initialisation de la fédération Platsim_QoS. Lorsque un simulateur Platsim_QoS désire joindre la simulation, il envoie le message "`createFederationExecution ("Platsim", "platsim.fed")` ". Ce message contient le nom de la fédération à laquelle le simulateur veut participer et le nom du fichier décrivant les différents flux à échanger avec les autres simulateurs (les objets et les interactions). Ainsi, si la fédération a déjà été créée, une exception `(RTI::FederationExecutionAlreadyExists)` sera levée pour assurer le simulateur de l'existence de la fédération. Le simulateur peut alors joindre cette fédération en envoyant le message "`joinFederationExecution( federateName, "Platsim", platsim_fedamb)` ".

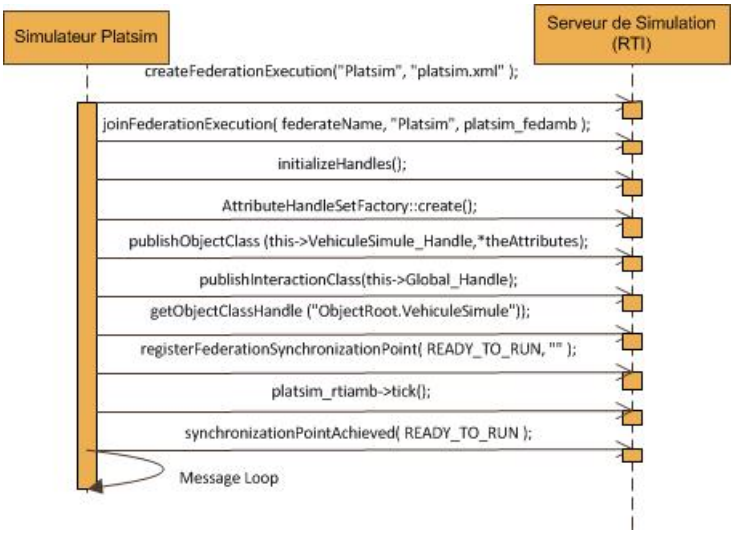


Figure 3.11: initialisation de la Fédération

Ensuite, chaque simulateur associe aux différents éléments du flux applicatif des identifiants uniques (les Handlers des objets et interactions) pendant toute la simulation et les publie vers le serveur de simulation (RTI) en envoyant le message "`publishObjectClass(this→VehiculeSimule_Handle, ∗theAttributes)`". Comme le fichier OMT suit la même organisation logique dans tous les simulateurs, les mêmes noms d'objets et d'interactions ont les mêmes Handlers.

Une fois ces objets et interactions identifiés, le simulateur procède à la préparation de la synchronisation avec les autres : création des différents points de synchronisation rendez-vous `synchronizationPointAchieved(READY_TO_RUN)`, l'appel des "Timers" temps-réel et l'exécution de mécanismes de synchronisation décrit dans le paragraphe précédent.

Scénario de communication au sein de la Fédération :

La Figure 3.12 présente un scénario de l'interaction entre deux fédérés (simulateurs Platsim_QoS) au sein d'un exercice de simulation distribuée. Pour simplifier l'illustration de la Figure, un objet ("VehiculeSimule") et deux interactions ("Global" et "AudioPacket") sont produits par le premier simulateur, envoyés vers le serveur de simulation (RTI) et réfléchis vers le simulateur qui va les consommer.

Le scénario débute par l'arrivée du fédéré consommateur dans la fédération. Ce fédéré déclare son souscription de l'objet "VehiculeSimule" et tous les attributs qu'il contient. Le serveur de

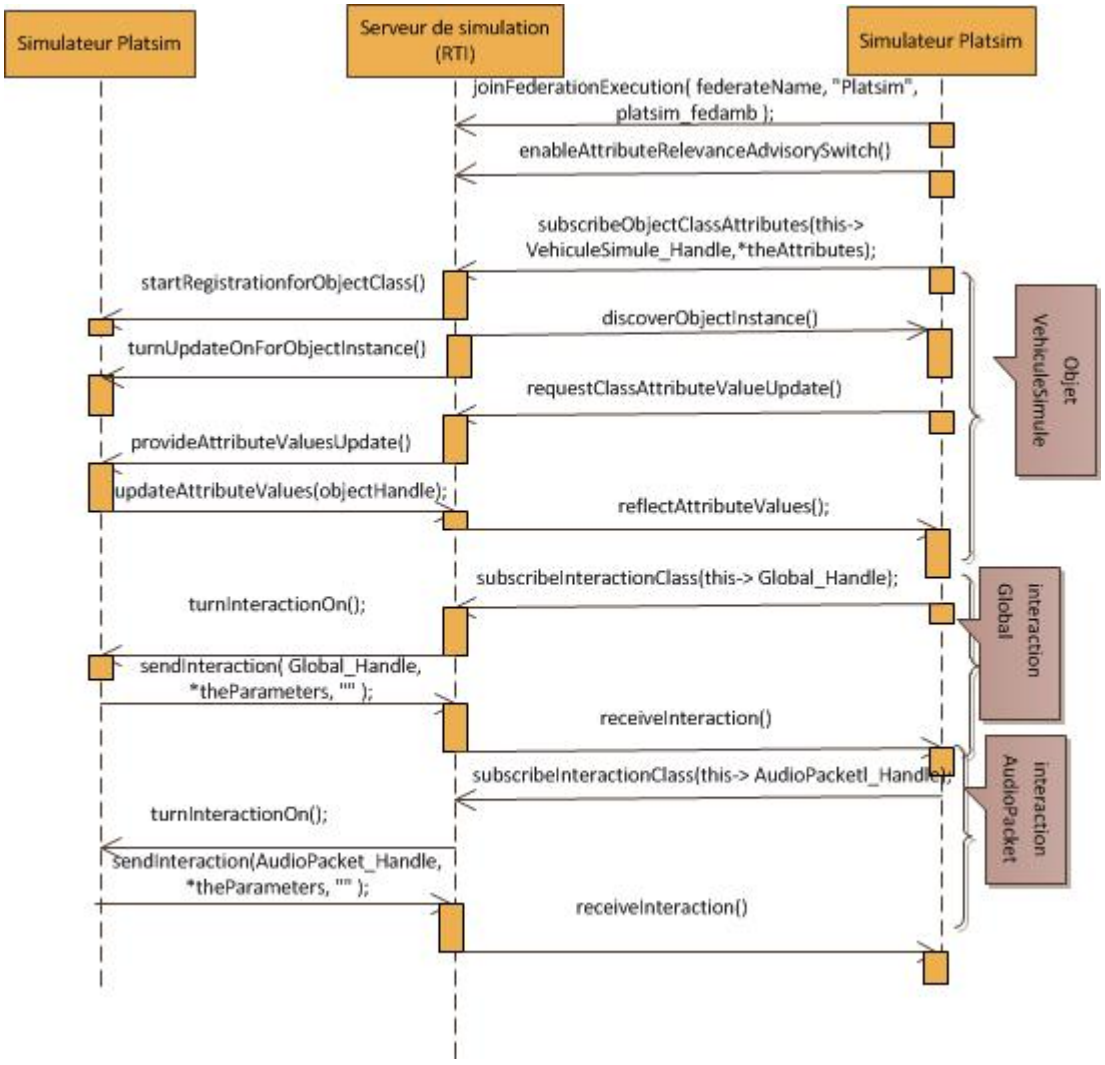


Figure 3.12: Publication et souscription entre deux simulateurs Platsim_QoS

simulation notifie le producteur (envoi des "Handles" des objets) pour qu'il commence l'enregistrement des instances des objets. Ces instances sont par la suite découvertes par le simulateur consommateur par le RTI. Si RTI envoie le message `turnUpdateOnForObjectInstance()` au producteur, c'est pour s'assurer que les valeurs des attributs spécifiés de l'instance de l'objet sont nécessaires à la simulation et qu'il devra maintenant envoyer les mises à jours des valeurs de ces attributs (`provideAttributeValuesUpdate()`) ; dès lors, ces nouvelles valeurs sont envoyées au serveur de simulation (RTI) par le service `updateAttributeValues(objectHandle)`, qui les distribue aux différents fédérés via la méthode (callback) `reflectAttributeValues()`. Le flux multimedia "audio" sur la Figure est aussi envoyé comme interaction. En effet, le simulateur dispose d'une interface qui lui permet de lancer la réception de l'audio et de la vidéo. Il envoie le message `subscribeInteractionClass(this→AudioPacketl_Handle)` pour spécifier l'instance des interactions auxquelles il s'abonne. Les identificateurs de ces instances seront alors réfléchis par le RTI vers le simulateur producteur. Dès lors, celui-ci commence à mettre à

jour les valeurs des paramètres de l'interaction "AudioPacket".

Scénario de terminaison de la Fédération :

La Figure 3.13 présente le scénario de terminaison de l'exercice de simulation. Le simulateur informe le serveur de la simulation qu'il souhaite se désinscrire de la simulation et arrêter de recevoir les instances des objets et des interactions (`unsubscribeObjectClass()` et `unsubscribeInteractionClass()`).

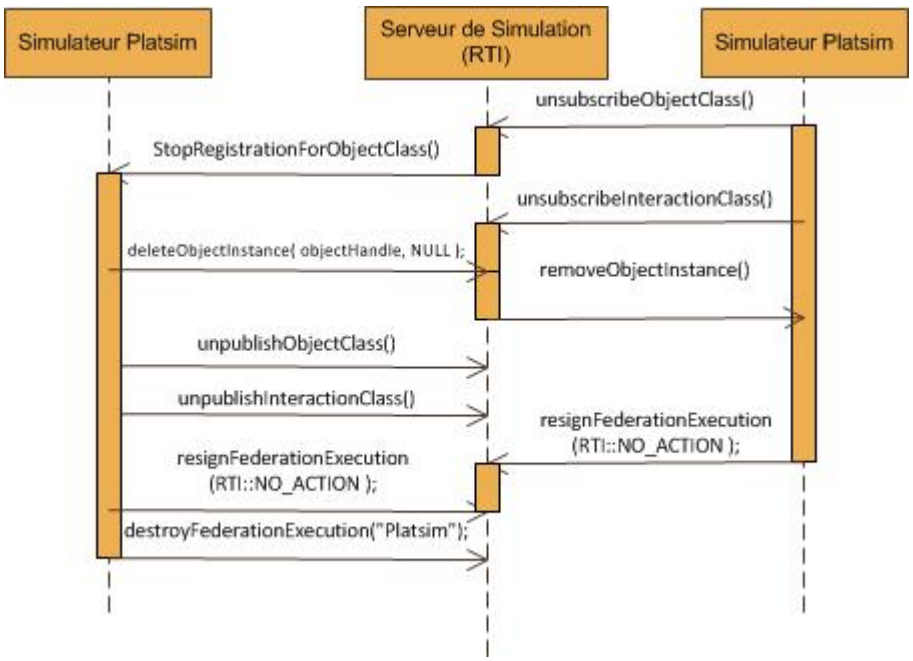


Figure 3.13: Terminaison et Libération de la Fédération entre deux simulateurs Platsim_QoS

Le fédéré producteur est notifié : il supprime les instances de son cache (`deleteObjectInstance (objectHandle, NULL)`) et suspend la publication des mises à jour des valeurs des attributs et des paramètres vers ce consommateur. Dès lors, ce dernier peut sortir de la fédération : il doit alors notifier le serveur de simulation qu'il quitte la simulation via le message `resignFederationExecution (RTI::NO_ACTION)`. La Figure 3.13 montre aussi le cas où les deux simulateurs désirent quitter la simulation. Dans ce cas-là, le dernier simulateur doit demander la destruction de l'exercice de simulation au serveur (`destroyFederationExecution("Platsim")`).

3.3.6 Conclusion sur La Simulation avec HLA

Dans cette section, nous avons présenté un modèle formel basé sur les réseaux de Pétri à Flux Temporel Hiérarchique (HTSPN) pour redéfinir les flux de données générés par l'application et élaborer un schéma de synchronisation entre eux. Ces flux découlaient directement des spécifications issues de la décomposition que nous avons proposée en spécifiant les différents "sous-flux". Chaque sous-flux a des caractéristiques intrinsèques exprimées par leur profil de trafic applicatif et par les exigences applicatives. Le modèle HTSPN offre à la fois de bons pouvoirs de modélisation, d'expression et d'analyse des contraintes de synchronisation. Il nous a permis

de spécifier précisément, complètement et de façon unifiée les contraintes de synchronisation logiques, temporelles et sémantiques. Nous avons défini, ensuite, les interfaces de programmation (APIs) que avons implémentées pour assurer les contraintes de la QoS sous HLA.
Dans la section suivante, nous proposerons les différentes possibilités de définition des données par rapport à DDS, ainsi que notre architecture de communication pour le support de la QoS par rapport au contexte de réseau local dans lequel nous nous plaçons.

3.4 Implémentation de la Simulation distribuée basée sur DDS

Dans cette section, nous allons spécifier l'application Platsim avec le middleware DDS, en focalisant sur l'expression de contraintes de QoS utilisant DDS/DiffServ. Pour cela, nous commençons d'abord par l'analyse des différents flux de Platsim sous forme d'éléments applicatifs conformes à DDS. Ensuite, nous décrivons leurs contraintes de QoS avec les concepts de DDS, les regroupons sous forme de classes de service conforme à DiffServ et définissons pour chaque classe ses exigences de QoS et ses paramètres QoS DDS. Enfin, nous présenterons l'architecture de mise en réseau local à QoS de Plasim_QoS sur DDS.

3.4.1 Analyse, Specification et caractérisation du trafic

La difficulté de l'analyse et de la caractérisation du trafic sur DDS réside dans l'agrégation des éléments applicatifs en classes ayant chacune leurs paramètres de QoS DDS et leurs exigences de QoS réseau. Pour cela, nous décrivons les critères de classifications ainsi que les difficultés d'association de la politique de QoS DDS avec la politique de QoS réseau.

3.4.1.1 Les critères de classification

La classification des flux porte sur deux axes principaux : la politique de QoS DDS, celle qui permet de paramétrer le service de distribution des données associé aux Topics PlatSim_QoS, et les classes de trafic DiffServ permettant de définir leurs comportements. Ces critères sont les suivants :

- **Fiabilité :** Porte sur la retransmission des données en cas d'erreur pour assurer que tous les paquets transmis doivent être délivrés au récepteur. Dans notre cas la fiabilité est gérée au niveau middleware DDS par le biais du paramètre Reliability.
- **Priorité :** Ce paramètre permet d'assurer la différenciation des flux. Elle est gérée au niveau middleware par la politique de QoS Transport_priority pour définir des priorités différentes aux canaux de transmission et marquer les paquets à l'aide du champ DSCP. Elle est définie aussi au niveau réseau par les routeurs de bordure pour assurer une différenciation entre les CoS du modèle DiffServ. Alors, dans ce cas, on doit garder une cohérence entre le middleware et le réseau pour être sûr que les différents types de trafic subissent le traitement souhaité.
- **Latence :** On distingue deux types de données : celles qui ne sont pas tolérantes aux délais et d'autres qui posent moins de contraintes de temps mais par contre exigent la fiabilité.

Figure 3.14: Topics DDS de la plateforme PlatSim

– **Profil de trafic :** Ce critère aide à identifier le trafic selon son rythme de transmission. Dans ce cas, on distingue deux types de données : des données périodiques ou régulièrement transmises avec une fréquence (f=60Hz) et d'autres apériodiques ou événementielles.

3.4.1.2 Caractérisation du trafic

Afin de réduire le volume de trafic échangé via le réseau, nous avons procédé à une recomposition basique des flux de données PlatSim de sorte que les flux soient constitués d'un agrégat de données élémentaires homogènes (en termes de régularité des mises à jour). Cette procédure a alors abouti à dix-sept topics (Figure 3.14) qui représentent tous les flux de données échangés entre les différentes entités du système de communication. Ces topics ont des profils de production différents ce qui nous permet de dégager 4 groupes qui sont : les données périodiques, les données multimedia, les états apériodiques et les données de contrôle apériodiques.

– **les topics périodiques :** regroupent les topics contenant les données d'état de chaque entité du système. Ils sont échangés périodiquement tout au long de la simulation afin de maintenir un état global du système et pouvoir ainsi garantir le même environnement simulé vu par chaque participant. Ces données permettent aussi au serveur de simulation de détecter les erreurs et aux instructeurs de suivre les apprenants (à travers les données des véhicules simulés correspondants, exemple vitesse moyenne) ;
– **les états apériodiques :** ce sont des topics événementiels dont la transmission est déclenchée par un événement. Ceci engendre des dates de transmission imprévues et irrégulières (comme les données pédagogiques déclenchées par une erreur commise par un apprenant, exemple collision, changements d'états d'un capteur dans un simulateur, ou le déclenchement d'une alarme). Nous allons les appeler simplement les états.
– **les données de contrôle apériodiques :** ce sont les topics de l'initialisation dont la transmission se fait une seule fois à l'initialisation de l'exercice. Nous allons nous référer à ces Topics simplement par contrôle.

– **les données multimedia :** ce sont les topics qui gèrent l'envoi de l'audio et de la vidéo entre les simulateurs et le poste instructeur.

3.4.2 Caractérisation de la communication entre l'application et le middleware

3.4.2.1 Les QoS policies des différents Topics

L'une des caractéristiques de DDS est de permettre de paramétrer très finement l'opération de distribution des données d'un topic en utilisant les paramètres de QoS DDS. Ces paramètres concernent les différentes entités DDS intervenant dans le processus de distribution de données (Topics, DataReaders, DataWriters, Publishers, etc.). Nous nous limitons dans le Tableau 3.3 (et le Tableau 3.4) à la présentation des QoS policies, que nous avons jugées suffisantes, associées aux topics DDS présentés ci-avant. En effet, du paramétrage de QoS d'un topic, DDS déduit un paramétrage par défaut pour les autres entités DDS impliquées par la distribution des échantillons de ce topic.

Type du Topic	DDS QoS policies				
	Fiabilité	Historique	Durabilité et persistance	Deadline	Transport Priority
Topics périodiques	UNRELIABLE	KEEP_LEAST	Volatile	{0s, 0ns}	32
États	RELIABLE	KEEP_ALL	Transient Local	{0s, 0ns}	31
Contrôle	RELIABLE	KEEP_ALL	Transient Local	{0s, 0ns}	31
Multimedia	UNRELIABLE	KEEP_LEAST	Volatile	{0s, 0ns}	26

Table 3.3: Paramètres de la QoS DDS pour les Topics de Platsim

Type du Topic	DDS QoS policies				
	Ownership	Latency_Budget	Vivacité	Lifespan	Time_based_Filter
périodiques	Exclusive	{0s,1000ns}	infini	0s,0ns	0s,0ns
États	Exclusive	{0s,0ns}	infini	{10s,0ns}	{0s,0ns}
Contrôle	Exclusive	{0s,0ns}	infini	{10s,0ns}	{0s,0ns}
Multimedia	Exclusive	{0s,300000ns}	infini	{1s,0ns}	{0s,25000000ns}

Table 3.4: Suite des paramètres de la QoS DDS pour les Topics de Platsim

Topics périodiques : Pour les Topics périodiques, les significations des différentes valeurs sont données comme suit :

– la persistance dite "volatile" correspond tout à fait au besoin de l'application, qui peut, si le message est incorrect utiliser le message suivant afin d'obtenir une valeur plus proche de la valeur actuelle de la donnée.
– La fiabilité quant à elle doit être positionnée à "UNRELIABLE", puisqu'une donnée perdue doit être remplacée par la future valeur et non une valeur ancienne retransmise.
– Le '*deadline*' et un 'latency budget' peuvent être spécifiés pour indiquer respectivement au middleware la fréquence de production des Topics et le délai nécessaire à son envoi à l'application distante.

- La priorité peut aussi être spécifiée pour aider le middleware à différentier les priorités entre données périodiques.
- La QoS policy "HISTORY" contrôle le fait que DDS envoie les instances les plus récentes des données. Dans cette optique, les échantillons sont envoyés de manière permanente entre les simulateurs ; ce paramètre est choisi *KEEP_LEAST*.
- La durabilité : Concerne la QoS DURABILITY qui contrôle la disponibilité des données dans l'espace global des données. Elle est utilisée si un nouveau simulateur joint la simulation pour récupérer les données qui ont été envoyées en son absence. La durabilité est choisi Volatile, car on n'a pas besoin des instances des données pour un simulateur qui rejoint la simulation plus tard.
- La vivacité : permet à DDS de vérifier si un DataWriter est en encore actif et d'aviser le DataReader correspondant. Elle permet de maintenir le service du système et d'informer les participants de l'entrée et de la sortie de la simulation. Par défaut, cette valeur est maintenue à l'infini.

États et données de contrôle : Pour les états et les données de contrôle, la configuration des QoS policies DDS a les significations suivantes :

- La fiabilité doit-être spécifiée comme fiable afin d'assurer la retransmission en cas de perte. Il n'est de fait pas judicieux de spécifier des paramètres temporels du fait de la reprise sur erreur. Toutefois, dans des cas particuliers, ces paramètres peuvent permettre d'implanter des fonctions proches d'un TimeOut.
- La durabilité est spécifiée comme "Transient Local" car l'entité productrice conservera quand même les données dans le cache local des DataWriters associés à ces Topics pour les envoyer au DataReaders qui vont rejoindre la simulation tardivement.
- La priorité peut elle aussi être spécifiée en fonction de l'importance de la donnée.

Les Topics multimédia : Finalement, les significations des paramètres de QoS DDS pour le flux multimédia sont les suivantes :

- La fiabilité du flux multimedia ne doit pas retransmettre les échantillons perdus pour optimiser l'utilisation de la bande passante, c'est pour cela que la QoS RELIABILITY est alors choisie UNRELIABLE.
- les échantillons sont envoyés de manière permanente entre les simulateurs, la QoS HISTORY est sélectionné *KEEP_LEAST*.
- La durabilité est choisi *volatile*, car nous n'avons pas besoin des instances des données pour un simulateur qui rejoint la simulation plus tard.
- la vivacité est maintenue à sa valeur par défaut, à savoir l'infini.
- Le Deadline : ce paramètre fixe le deadline limite entre deux mises à jours consécutifs des instances des objets. Si cette date limite expire, DDS informe l'application, permettant de détecter les éventuels problèmes dans l'échange entre les DataWriters et les DataReaders.
- Le TIME_BASED_FILTER permet de limiter le nombre d'échantillons à envoyer par seconde, en établissant un temps minimum de séparation entre eux. En cas de problème détecté par le DataWriter sur la vidéo, le système réduit le taux de transmission en réduisant la qualité du flux vidéo.

- La Durée de vie (LIFESPAN) : ce paramètre permet d'empêcher l'envoi d'échantillons avec des retards excessifs. Ainsi, chaque échantillon de données écrit par DataWriter a un délai d'expiration associé, après quoi il ne faut pas l'envoyer. Une fois la durée de vie des échantillons expirée, ils seront retirés du cache du DataReader. Cette stratégie permet de rejeter les paquets qui ont un retard excessif, et rend particulièrement utile l'application interactive. Elle s'assure que les paquets qui dépassent un certain délai ne seront plus valables et doivent être retirés du tampon.
- La propriété (OWNERSHIP) : la politique propriété détermine si un DataWriter pourrait mettre à jour (actualiser) les données, ou au contraire empêche des DataWriter bien particuliers de le faire. La propriété *OWNERSHIP_STRENGTH* permet d'attribuer à un DataWriter l'obligation de mettre à jour ou de l'interdire, et de s'assurer que les données qu'il actualise seront éventuellement remises au DataReader correspondant. Cette politique est à utiliser sur le canal Audio pour s'assurer que la voix d'un élève sur un simulateur est entendue sur l'instructeur sur son poste.

Enfin, nous avons utilisé un Topic spécifique qui n'est pas inclut dans la spécification des flux de Platsim_QoS. Ce Topic s'appelle "Built-in Topic" et permet la supervision de la simulation et l'enregistrement des erreurs et des avertissements que peut subir l'application Platsim_QoS ; par exemple, le Built-in Topic permet de détecter l'arrivée d'un nouveau simulateur dans la simulation. Ce Topic est semblable au service "Dicovery" de DDS (le même principe d'implémentation), il a des paramètres de QoS spécifiques et qui ne nécessitent pas un traitement spécifique au niveau des noeuds du réseau.

3.4.3 Caractérisation de la communication entre le middleware et le réseau

3.4.3.1 Lien entre la QoS du middleware DDS et la QoS réseau

Une politique de QoS est appliquée au niveau du réseau afin de prendre en compte et répondre aux exigences de QoS de PlatSim. Ces exigences étant exprimées par le biais des QoS policies DDS, le middleware DDS est amené à traiter localement, au niveau des noeuds terminaux, les échantillons à distribuer selon la QoS associée à leur service de distribution DDS (en jouant entre autres sur la priorité des threads de transmission, priorité de traitement dans les files de transmission, taille des files, etc.) ; mais également à reposer sur un service d'acheminement de niveau réseau capable de garantir une qualité de service qui permette de satisfaire la QoS de niveau DDS. Cela suppose, d'une part, la définition d'une politique de QoS appropriée permettant au réseau d'offrir les services adéquats et d'autre part, de permettre au middleware DDS de choisir et éventuellement paramétrer le service réseau à utiliser.
En effet, la politique de QoS doit répondre aux exigences de débit des différents topics et aux exigences sur les délais de distribution. Quant à la fiabilité, le réseau n'a pas à la garantir puisqu'elle est à la charge du middleware DDS (d'ailleurs, comme indiqué dans l'état de l'art, les deux middlewares considérés reposent sur un service UDP). En plus, le moyen à disposition des middlewares pour contrôler le service de niveau réseau est d'indiquer via le paramètre "Transport_Priority" le pré-marquage des champs DSCP ou TOS par la classe de trafic à utiliser.
La prise en compte des exigences de débit se fait assez aisément. En effet, pour un scénario de simulation donné, tous les flux de données et leurs caractéristiques (profil de production) sont

connus à l'avance. Une réservation de ressources peut être effectuée avant le démarrage de la simulation.
Pour ce qui concerne les exigences de délai l'approche suivie pour le contexte local consiste à favoriser le traitement des données ayant les exigences les plus strictes au niveau des éléments du réseau. Cela est suffisant puisque le délai de transfert des données les plus exigeantes devrait avoisiner les temps d'émission. En effet, elles seront traitées de manière prioritaire sous des conditions de charge du réseau contrôlées.
La politique de QoS définie pour PlatSim distingue quatre classes de trafic :

- Classe N°1 : cette classe regroupe le trafic des Topics possédant une périodicité de type périodique et nécessitant des délais très courts.
- Classe N°2 : cette classe regroupe le trafic des Topics des données de contrôle apériodiques qui exigent aussi des délais très courts.
- Classe N°3 : cette classe regroupe le trafic généré par les flux multimedia. Les exigences sur le délai sont comparables aux applications de streaming, i.e. un délai maximum de l'ordre de la demi-seconde ;
- Classe N°4 : cette classe regroupe les Topics des états et qui exigent aussi des délais très courts.

Ces classes sont identifiées dans le réseau par la valeur du champ DSCP des datagrammes IP. Le marquage DSCP que nous préconisons essaye de se rapprocher des recommandations de l'IETF de la RFC 4594 [9].
Nous avons fait correspondre la classe N°1 à la classe "Real-Time Interactive" de l'IETF car toutes deux se partagent les mêmes exigences : très peu de tolérance sur les délais et peu ou moyennement tolérance aux pertes. Le trafic de la classe N°1 hérite donc du marquage de la classe "Real-Time Interactive", à savoir un marquage DSCP à 32. De la même manière, nous avons fait correspondre la classe N°3 à la classe de service "Multimedia Streaming" de l'IETF. Elle hérite donc d'un des marquages DSCP, à savoir 26. Pour la classe N°2, nous fixons le marquage DSCP à 31 qui ne correspond à aucune classe de service promue par l'IETF. En effet, à cause du fonctionnement de RTI-DDS qui effectue une correspondance directe entre la valeur de la QoS policy "Transport Priority" et le marquage DCSP, ce dernier doit prendre une valeur comprise entre celles utilisées par les deux autres classes pour maintenir le niveau de priorité relatif entre les classes (dans DDS, plus la valeur "Transport Priority" est grande plus la données est considérée prioritaire). Afin d'éviter de faire correspondre la classe N°2 à une classe de service IETF ayant des exigences différentes, nous faisons le choix d'affecter à celle-ci un marquage DSCP non utilisé par l'IETF, à savoir 31. Enfin, la classe N°4 correspond à la classe "standard" de l'IETF et hérite d'un marquage DSCP à 0. Le Tableau 3.5 résume le marquage DSCP par topic.

Type du Topic	Numéro de la classe	Valeur DSCP
Topics périodiques	1	32
Topics États	2	31
Topics Contrôle	2	31
Topics Multimedia	3	26
Topics Discovery	4	0

Table 3.5: Marquage DSCP des flux de données par type de Topic

3.4.4 Implémentation du schéma de communication avec DDS

L'implémentation de l'exemple de Platsim_QoS sur DDS est réalisée avec les deux middlewares cités ci-avant. L'application contient deux participants : un producteur et un consommateur de données de platsim. Nous présentons ici les interfaces de programmation que nous avons développées et utilisées.

3.4.4.1 Les interfaces de programmation

Rappelons que nous avons présenté, au début de cette section, les différents topics que nous avons décrits pour Platsim_QoS. Les interfaces de programmation se ressemblent pour les producteurs et pour les consommateurs au niveau implémentation, mais elles diffèrent par leurs fonctionnalités :

Coté Publisher : Le producteur gère la création des différents Topics périodiques, des Topics états, des Topics media et des Topics de contrôle. En effet, ces différents topics diffèrent par leurs DDS QoS policies et par les besoins de traitement au niveau réseau. La Figure 3.15 montre la création d'un DataWriter DDS associé au Topic Media pour le contrôle du flux multimedia. Les QoS policies utilisées pour ce Topic sont sauvegardées dans un fichier "XML" dans un profil "MediaProfile". Comme les flux sont identifiés par des clés qui définissent le simulateur originaire de ces flux, il est alors nécessaire d'enregistrer les différentes instances avant leur envoi.

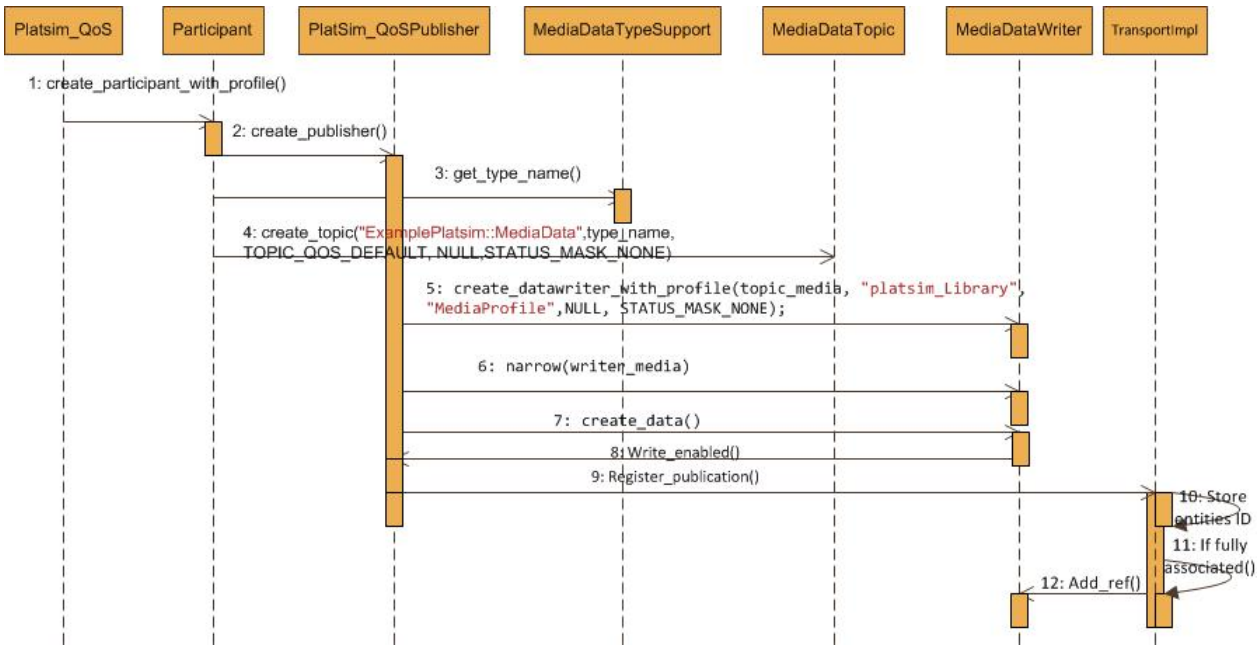


Figure 3.15: Scénario de la préparation à la publication

La Figure 3.15 montre aussi un exemple de scénario complet de création du DataWriter Multimedia et de préparation des données avant leur publication. La première étape de la simulation est de créer le participant (publisher et/ou subscriber) et de lui associer les paramètres de QoS DDS correspondants.

Comme nous l'avons dit dans l'état de l'art, les middlewares DDS offrent aux concepteurs plusieurs méthodes pour spécifier les QoS policies aux différentes entités de la simulation (participant, topic, DataReader, etc.). Il est possible de spécifier ces paramètres de QoS directement dans le code, ou bien dans le fichier de configuration en XML. Cette dernière méthode permet plus de flexibilité car elle donne la possibilité de changer la QoS DDS au cours de l'exécution de l'application tandis que l'implémentation dans le code source nécessite la compilation du code source. Par exemple, en chargeant les QoS policies du **MediaDataWriter** à partir du profil "MediaProfile" comme le montre la Figure 3.15, nous avons pu tester plusieurs possibilités des différents valeurs de ces paramètres pour choisir celles les mieux adaptées à Platsim_QoS.

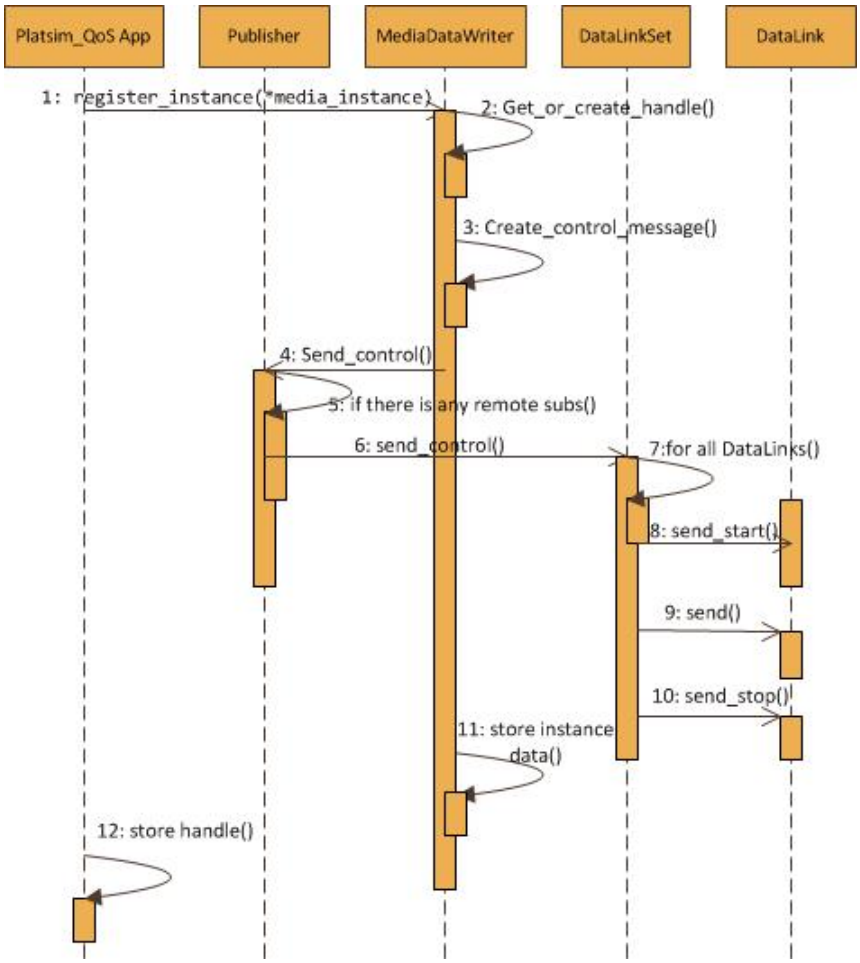


Figure 3.16: Procédure d'enregistrement des instances du Topic Media avant leur publication

La Figure 3.16 montre le mécanisme d'enregistrement des instances du flux multimedia avant son envoi vers les autres simulateurs distants. Notons que le "MediaDataWriter" prépare les instances des données qui vont être envoyées vers les subscribers distants de la manière suivante : il prépare les allocations des espaces mémoires pour les données par la fonction "*create_data*()", rend le DataWriter disponible au publisher et enregistre les différentes instances pour être accessibles à la publication. Enfin, il communique les Identifiants des entités et ceux des QoS policies pour les stocker par le plug-in de transport (DataLinkSet et DataLink dans La Figure 3.16)

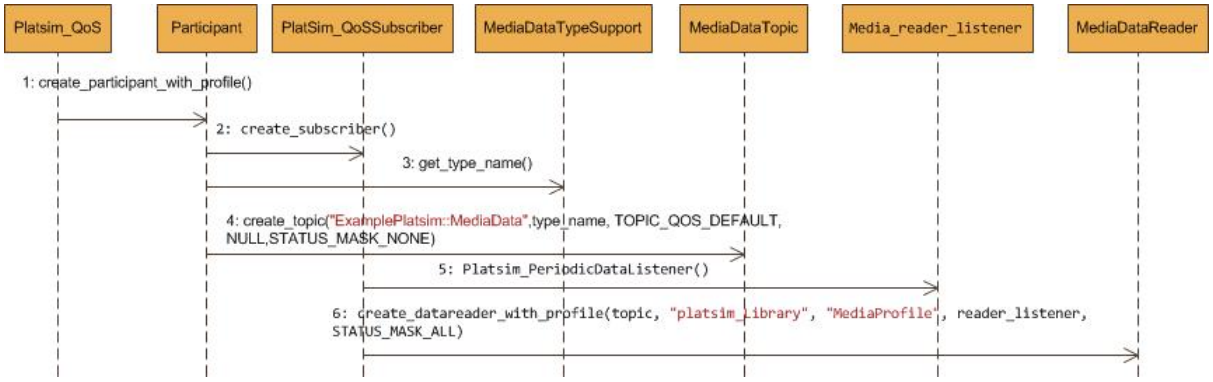


Figure 3.17: Scénario de la préparation à la souscription

afin de les rendre accessibles au protocole "Discovery".

Coté Subscriber : Les simulateurs qui souhaitent consommer les instances des topics envoyées par les publishers utilisent les "DataReaders" correspondant pour les recevoir. La Figure 3.17 montre le scénario de préparation du DataReader pour la réception du flux multimedia. En effet, chaque simulateur prépare son subscriber pour recevoir les données de la part du producteur. Il compare les politiques de QoS par leurs valeurs et leurs identifiants à ceux déjà décrits dans le profil de QoS du DataReader. Si ces QoS policies ne présentent pas de problème de compatibilité pour le contrat Offre/Réponse de DDS, la communication peut s'établir, sinon une erreur sera générée.

Scénario de communication : La Figure 3.18 illustre le scénario complet pour l'envoi et la réception des données. Ce scénario suppose que les différentes étapes décrites dans les paragraphes précédents se sont déroulées avec succès : une fois le MediaDataWriter et le MediaDataReader créés, les instances sont enregistrées au niveau du DataLinkSet, le publisher regarde la fonction "$Writer\_enabled()$" pour vérifier si le DataWriter est près pour lui envoyer les données. Il notifie l'application pour qu'elle associe les différents DataWriters à leurs publishers (respectivement, une association entre les différents DataReaders avec leurs subscribers respectifs) qui se chargent d'acheminer les données vers le protocole de transport sous-jacent.
Le plug-in de transport réserve les ressources dans les tampons d'émission et crée les associations entre le publisher et le subscriber dans une carte d'association (MapID) qu'il rend disponible dans l'espace de partage global de DDS. Ensuite, le subscriber s'abonne à cet espace global pour associer les Topics avec leurs QoS policies respectifs aux DataReaders. Cette association permet aux "Listeners" de distinguer les différents Topics par leurs identifiants et d'enregistrer leurs instances dans le cache de chaque DataReader. Ces derniers utilisent les fonctions de notification pour récupérer les instances et les acheminer vers l'application. Par exemple, la fonction "$on\_data\_available(DataReader \ *reader)$" attache un "listener" au DataReader pour qu'il puisse être notifié de la présente des données (Topics) dans le cache du Subscriber.

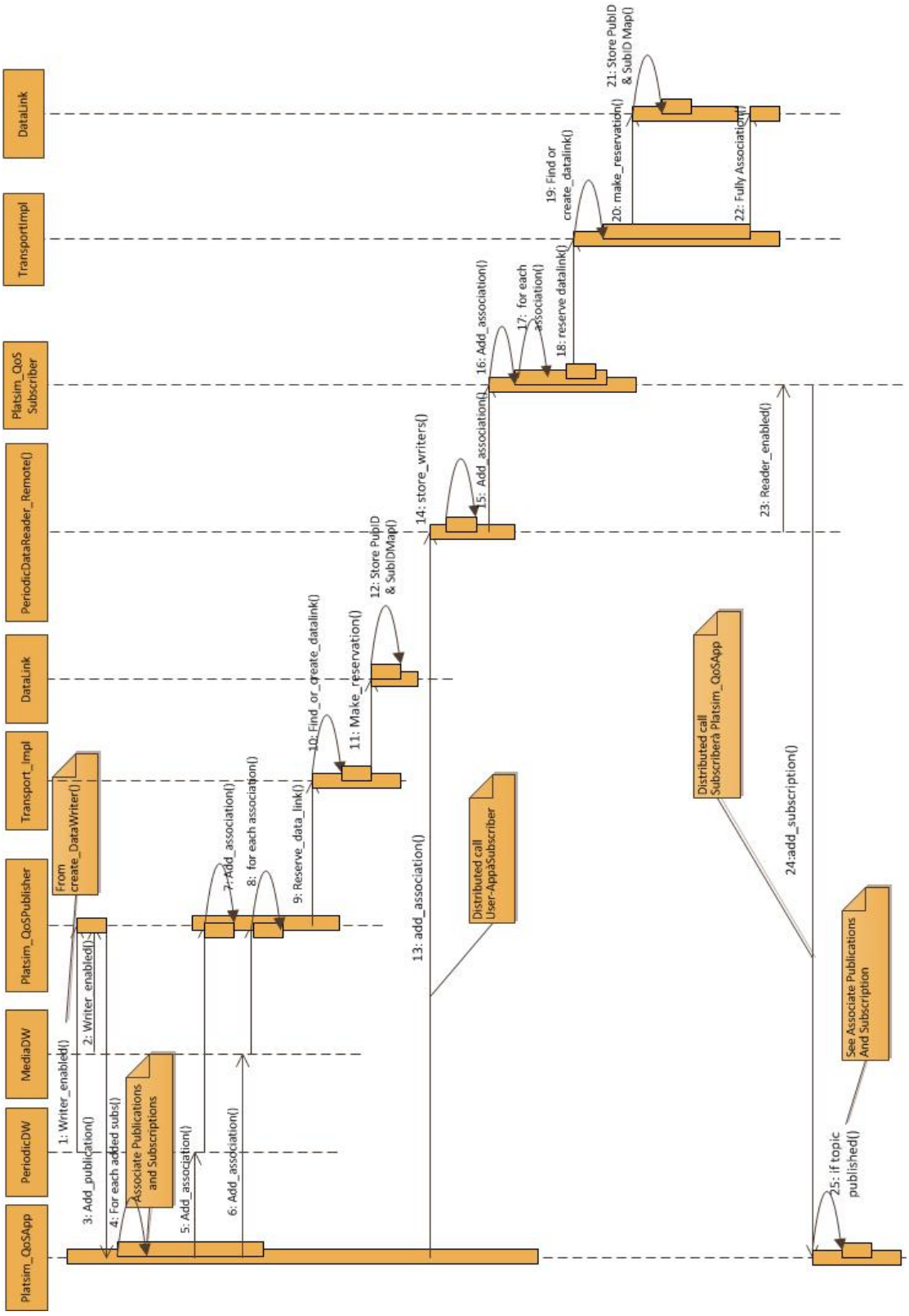


Figure 3.18: Enregistrement des données dans l'espace partagé

3.5 Environnement de développement et évaluations dans les réseaux locaux

L'un des objectifs du projet PlatSim est de développer une plateforme capable d'émuler précisément le comportement de l'application de simulation de conduite afin de pouvoir mesurer les performances qu'auraient les applications de simulation distribuées et si on désirait les intégrer dans des réseau de nouvelle génération (NGN). Dans cette partie, nous décrivons la plateforme de test "LaasNetExp" et nous illustrons les différentes évaluations de performance des approches HLA et DDS et les résultats de l'architecture de la simulation distribuée.

3.5.1 Environnement de développement et plateforme d'évaluation

Le LAAS-CNRS a installé une plateforme de tests pour les différents projets de recherches auxquels le laboratoire participe activement, permettant d'évaluer les communications des réseaux locaux aux communications par satellite, tout en passant par les réseaux Internet mono et multi-domaines, fixes et mobiles [104].

3.5.1.1 LaasNetExp : la plateforme de test

LaasNetExp (Figure 3.19), a été conçue et mise en place pour être indépendante du réseau opérationnel du LAAS pour éviter toute perturbation mutuelle avec celui-ci. Elle est raccordée directement au réseau RENATER [1], et à travers RENATER à plusieurs réseaux de recherche européens. Les travaux et les expérimentions de cette thèse ont été réalisé sur cette plateforme. Nous considérons trois domaines IP avec des adresses publiques qui émulent une architecture DiffServ reliés par le biais de 3 routeurs de type Juniper M7i : "Posets", "Aneto" et "Montperdu". Nous utiliserons trois machines Euqos 6, Euqos7 et Euqos8 comme entités de PlatSim, réparties sur les deux réseaux d'extrémité (Euqos6 représente le serveur de simulation, Euqos7 et Euqos8 représentent les simulateurs).

3.5.1.2 Les outils nécessaires à l'évaluation

3.5.1.2.1 Les outils de base Afin de pouvoir réaliser les tests et les expérimentations au plus proche de la réalité, il est tout d'abord important d'installer au niveau de chaque machine utilisée comme simulateur un middleware HLA et un middleware DDS pour mettre en oeuvre les mécanismes de communication entre les différents émulateurs développés et de pouvoir synchroniser l'horloge locale des simulateurs sur une référence d'heure commune à toutes les machines via le protocole NTP.

- **NTP Deamon v 4.2.4p8 [2],** est un démon du protocole NTP (Network Time Protocol) [86] en version 4 fonctionnant sur Windows. Il met et maintien le système à l'heure synchronisé avec le serveur de temps, avec la possibilité d'avoir plusieurs niveau hiérarchiques.
- **Middleware HLA-RTI,** PRTI est un middleware HLA multi-plateforme développé par la compagnie Pitch Technologies [3]. Nous avons choisi cette solution en version académique

1. http://www.renater.fr/
2. http://www.meinberg.de/english/sw/ntp.htm
3. http://www.pitch.se/products/prti

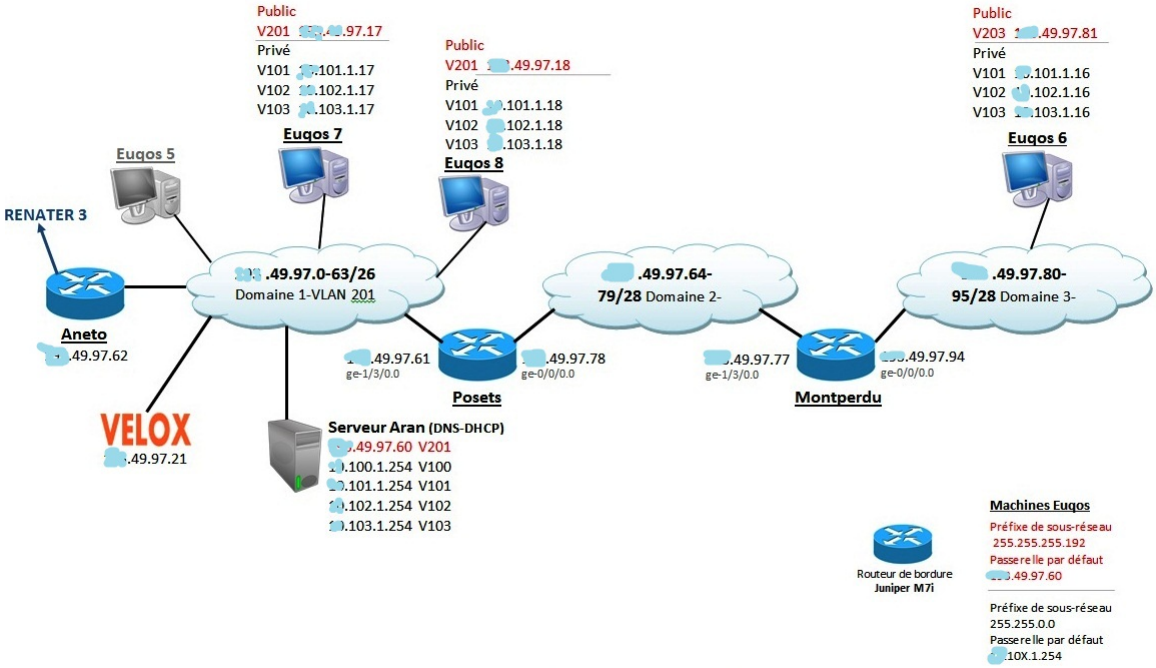


Figure 3.19: Plateforme d'expérimentation réseau au LAAS

pour sa simplicité d'utilisation et sa conformité avec la version IEEE-1516 de HLA que les solutions gratuites ne fournissent pas.

– **Middleware DDS,** parmi les solutions existantes sur la marché nous avons préféré les deux solutions en version académique "OpenSplice DDS" [1] et "NDDS" [2] car elles supportent toutes les politiques de QoS du standard DDS.

3.5.1.2.2 Les outils pour la capture et l'analyse des flux : Pour les mesures des performances, nous avons utilisé une suite d'outils pour DDS qui permettent de capturer et d'analyser le trafic envoyé/reçu par les différentes entités de l'exercice. Cette suite comprend les logiciels "RTI Addin pour Wireshark", "RTI Monitor", "RTI Analyser" et "RTI Spreadsheet Add-in pour Microsoft Excel".

– **RTI Addin pour Wireshark,** est un plug-in ajouté à l'analyseur de trafic Wireshark [3] supportant le protocole RTPS (Real-Time Publish-Subscribe), une couche spécifique ajoutée pour le transfert de données DDS. Il permet de capturer à la volée, pendant une durée donnée, l'ensemble du trafic de données DDS IPv4 et IPv6 échangé entre les différentes entités simulées.

– **RTI Monitor,** permet le contrôle et la supervision des données échangées entre un ou plusieurs producteurs et un ou plusieurs consommateurs dans une application DDS. Il permet de vérifier la consistance du modèle conçu, de s'assurer que les communications

1. `http://www.prismtech.com/opensplice`
2. `www.rti.com`
3. `www.wireshark.org`

DDS se déroulent comme prévu et d'aider à améliorer les performances en fournissant des statistiques sur tous les aspects internes du fonctionnement du middleware RTI DDS.

- **RTI Analyser,** est un outil de diagnostic de communications égal-à-égal entre deux participants pour dresser la topologie des entités communicantes. Il écoute le méta-trafic envoyé par une application RTI-DDS dans un domaine spécifique, ensuite construit une base de données des noeuds, applications, topics et des autres objets DDS. Ces informations sont par la suite affichées sur une ou plusieurs fenêtres selon une arborescence organisée selon la configuration des participants.
- **RTI Spreadsheet Add-in pour Microsoft Excel,** est un plug-in de DDS qui permet de réaliser une analyse rapide des données en offrant la possibilité de visualiser des données en temps-réel échangées dans un même domaine DDS. Il s'appuie sur des classeurs Excel pour tracer des histogrammes, des cylindres et des courbes en temps-réel. Cependant, il ne permet pas de donner des statistiques sur les communications entre les participants en termes de performances réseau. Pour cela, cet outil sert à vérifier que l'exercice de simulation interactive fonctionne selon le modèle de communication qui lui a été établi.

3.5.2 Evaluations des performances dans les réseaux locaux

3.5.2.1 Evaluation des Performances et Benchmark : HLA versus DDS

Pour évaluer la performance de l'architecture sous HLA, nous allons analyser le comportement de celle-ci selon 3 métriques : le délai de transfert, la gigue maximale admissible et la bande passante. Dans cette configuration, nous avons utilisé deux machines fédérés : un fédéré producteur génère du trafic HLA, un fédéré consommateur souscrit aux données envoyées par le producteur et sur lequel nous avons installé le moteur d'exécution HLA-RTI qui assure la communication entre ces deux fédérés. Toutes les expériences menées avaient une durée minimale de 15 minutes.

Pour ces expériences, nous démarrons le moteur d'exécution à l'instant t=0s, ensuite nous lançons le fédéré consommateur dans un premier temps, et ensuite nous lançons le fédéré producteur dans un second temps. L'ensemble des entités de cet exercice de simulation est synchronisés par le protocole NTP en utilisant le daemon **ntpd**.

Concernant le scénario sous DDS, nous avons utilisé un seul publisher et un seul subscriber, tous les deux synchronisé par NTP. Nous avons démarré le subscriber à l'instant t=0s ensuite nous avons lancé le publisher dans le même domaine DDS pour qu'il puisse communiquer. Cette configuration utilise un seul Topic DDS de taille variable passée en ligne de commande pour chaque expérimentation.

Une série de tests avec différentes configurations a été conçu pour mesurer les effets de la latence, de la gigue et d'établir des comparaisons de performances entre les middlewares HLA-RTI et DDS.

La Figures 3.20 illustre certaines traces récupérées des expérimentions avec le middleware HLA. Il s'agit donc de déterminer à partir des fichiers traces obtenus lors des expérimentions le meilleur choix entre les deux middlewares.

```
============== INIT_HANDLES FUNC Attributes ==============
DEBUG [main] : [Request] getObjectClassHandle(): name=ObjectRoot.VehiculeSimule
DEBUG [main] : [Request] getAttributeHandle(): class=508, attribute=voitureVitesseArbre
DEBUG [main] : [Request] getAttributeHandle(): class=508, attribute=voitureCoeffDerapT
DEBUG [main] : [Request] getAttributeHandle(): class=508, attribute=voitureCoeffDerapL
============== ATTRIBUTES INITIALIZED ==============

============== INIT_HANDLES FUNC PARAMETERS ==============
DEBUG [main] : [Request] getInteractionClassHandle(): name=InteractionRoot.Global
DEBUG [main] : [Request] getParameterHandle(): class=574, parameter=exitCode
DEBUG [main] : [Request] getParameterHandle(): class=574, parameter=codeLangue
DEBUG [main] : [Request] getParameterHandle(): class=574, parameter=environnement
============== PARARMETERS INIZIALIZED ==============
```

Figure 3.20: Trace issue de simulation de Platsim_QoS sur HLA

Evaluation du délai de transfert : L'objectif est de déterminer l'évolution du délai de transfert en fonction, d'une part de la taille des paquets envoyés, et d'autre part de la charge du réseau. Nous allons pour cela analyser les graphiques de la Figure 3.21 correspondant à la courbe du délai (Latence (μs)), et la Figure 3.22 correspondant à la gigue observée.
La Figure 3.21 compare les résultats de la latence sous HLA et DDS. La latence spécifie le délai maximum acceptable du moment où les données sont écrites jusqu'à ce que les données soient insérées dans le cache de l'application du récepteur et que l'application réceptrice en est notifiée. L'exemple de la simulation sur l'architecture DDS présente des résultats meilleurs que

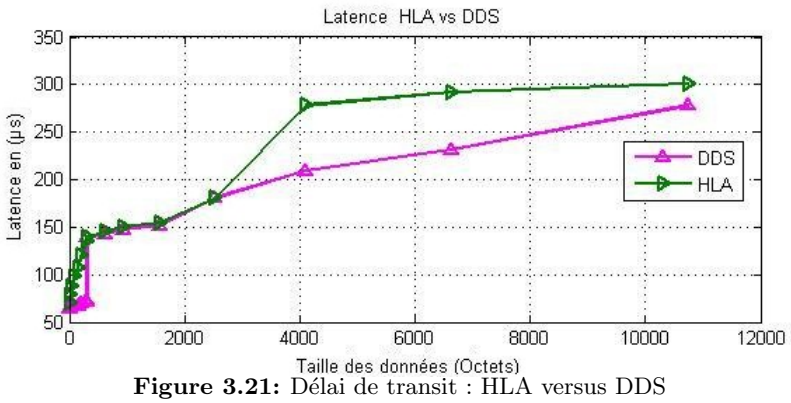


Figure 3.21: Délai de transit : HLA versus DDS

celui sur une architecture HLA. En effet, pour une taille de données applicatives inférieure à 1500 octets, DDS présente un léger avantage par rapport à HLA. Dès que la taille des données dépasse 2500 octets, DDS affiche des performances meilleures ; par exemple, le délai de bout-en-bout en réseau local pour une taille de données applicative de 4000 Octets est 210μs pour DDS et 280μs pour HLA. Ces expérimentations ont été réalisées au moins deux fois pour s'assurer des résultats trouvées. En plus, la variation des délais (gigue) illustré sur la Figure 3.22 montre aussi de meilleurs résultats pour DDS.

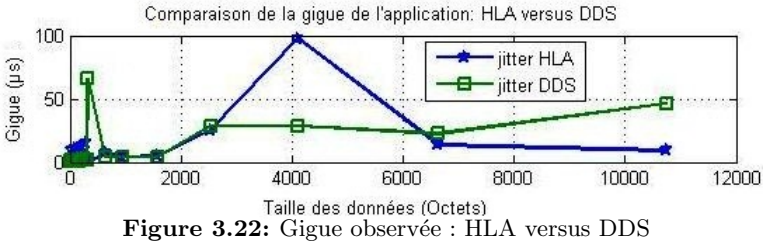


Figure 3.22: Gigue observée : HLA versus DDS

Evaluation de la bande passante : Le Tableau 3.6 évalue les débits binaires pour HLA et DDS en fonction de la taille des données (en octets). Cette évaluation montre que DDS

Taille des données (Octets)	10	100	1000	5000
Débit en Mbps (HLA)	2	30	128	350
Débit en Mbps (DDS)	6	40	112	800

Table 3.6: Débit (Mbps) : HLA versus DDS

affiche aussi des performances bien meilleures concernant le débit. En effet, lorsque la taille des données applicatives augmente à 5000 octets, DDS peut mieux supporter le trafic. Nous avons pu atteindre la limite théorique du lien physique du commutateur local de 1000Mbps.

Conclusion sur les deux architectures : Les évaluations de performances réalisées dans le cadre du réseau local montrent que la mise en place de la simulation distribuée avec DDS montre beaucoup d'avantages par rapport à HLA au niveau du délai de transit et au niveau de la bande passante. DDS est un middleware temps-réel orienté vers les données (data-centric) ; ces paramètres de QoS lui permettent de réaliser l'équilibre entre les besoins de l'application et la fourniture de la QoS au niveau réseau. Nous étudierons dans la suite de cette section uniquement DDS.

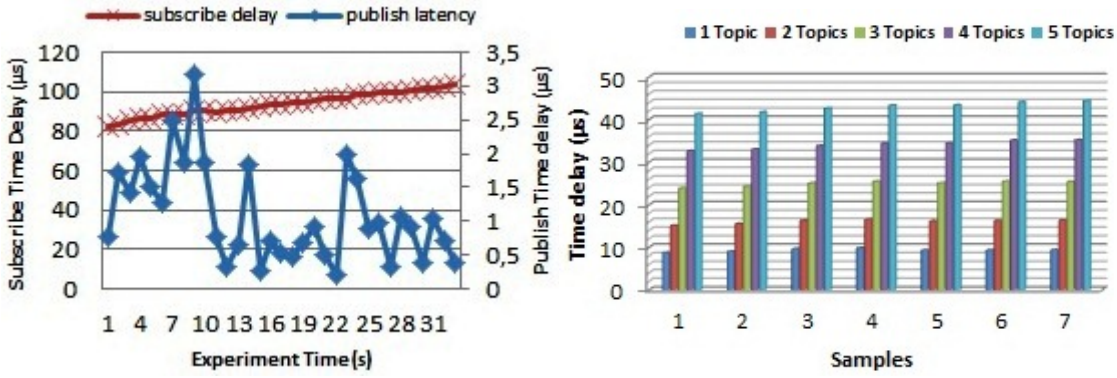
3.5.2.2 Evaluation de l'architecture avec DDS

Dans cette partie, nous allons évaluer plus en détails les performances de la solution DDS. Pour cela, nous allons commencer à étudier l'impact de l'ajout de la couche middleware sur le délai de transfert de bout-en-bout dans le contexte réseau local.

3.5.2.3 Evaluation de latence entre l'application et le middleware DDS

Modèle de Test : Nous considérons alors deux simulations distribuées de Platsim_QoS sur le réseau local de la plateforme, synchronisées par le protocole NTP. Chaque application est formée par un seul publisher et un seul subscriber contenant chacun 17 Topics correspondant au différents flux de PlatSim_QoS que nous avons décrits dans la section précédente.
Nous allons analyser le graphique de la Figure 3.23 correspondant au délai moyen (One Way Delay) de traversée des données depuis l'application vers le middleware et vice-versa. L'objectif est, alors, de mesurer le temps requit par le middleware pour traiter les flux applicatifs : les encapsuler dans les Topics, les envoyer au DataWriters et les communiquer vers les publishers.

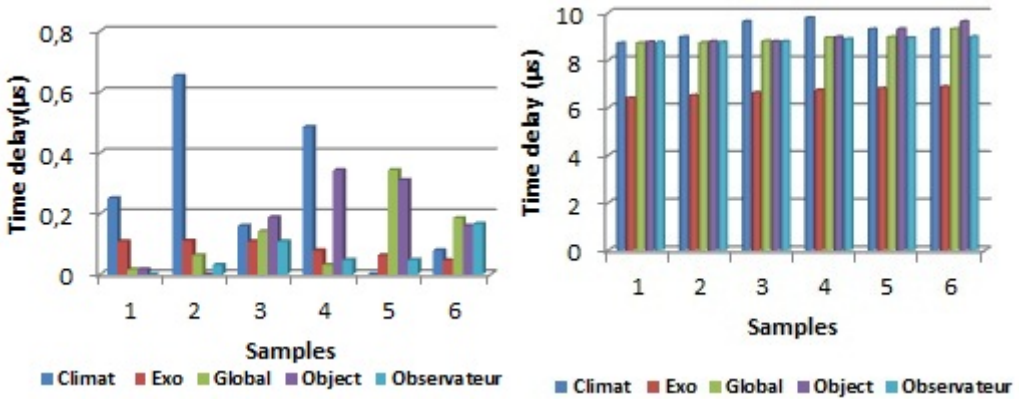
Impact du nombre de Topics : D'après la Figure 3.23a, nous constatons que le délai moyen nécessaire pour la publication des 17 Topics de Platsim_QoS depuis l'application vers le middleware est environ $3\mu s$. Par contre, le délai nécessaire pour récupérer les Topics depuis le middleware et son affichage par l'application est entre $80\mu s$ et $100\mu s$. La Figure 3.24b montre ce délai en fonction du nombre de Topics. En effet, lorsque le nombre de Topics augmente, le délai à la subscription augmente en conséquence, par exemple, pour 1 seul Topic se délai est de $8\mu s$, pour deux Topics est de $15\mu s$ pour 5 topics ce délai atteint $41\mu s$.



(a) Latence à la production et la souscription **(b)** Latence en fonction du nombre de Topics

Figure 3.23: Latence totale pour les Topics Platsim à QoS

Impact de la taille des Topics : La Figure 3.24a montre le délai entre l'application et le middleware à la publication par Topic. En effet, nous remarquons que les valeurs du délai de publication entre les 6 Tests sont proches les unes des autres. Par exemple, pour le Topic "Climat" qui a une taille de 200Octets, ce délai varie entre $0,2\mu s$ et $0,6\mu s$. Le Topic "Exo" dont la taille est de 20 Octets, ce délai est inférieur à $0,1\mu s$. Ceci est aussi le cas pour le reste des autres Topics. Nous constatons alors que la taille des Topics n'influence pas trop le délai de la publication vers le middleware.
En effet, le middleware doit préparer pour chaque topic les emplacements mémoires, enregistrer ces différentes instances, et préparer les données pour les mettre dans la file d'attente d'envoi du middleware. Ceci peut s'expliquer par le fait que le paramètre "**PublisherQosPolicy**" de QoS policy de DDS qui gère le mode de publication utilise le thread du middleware au lieu du thread utilisateur qui est prioritaire. Le mode de publication utilisé par le DataWriter détermine si les données sont écrites de façon synchrone dans le contexte du thread utilisateur lors de l'appel de la fonction "write()" ou de manière asynchrone dans le contexte d'un thread séparé interne au middleware. Ainsi, pour envoyer les données le plus rapidement possible, le thread utilisateur (moins prioritaire) lance le thread middleware (le plus prioritaire) qui prend en charge l'acheminement des données du flux applicatif vers (resp. depuis) les DataWriters (resp. DataReaders).
De même, la Figure 3.24b montre le délai à la souscription entre le middleware et l'application par Topic. Les résultats sont plus grands pour chaque Topic que ceux que nous avons trouvés à la publication. Par exemple, pour le Topic "Climat", nous avons trouvé $0,2\mu s$ à la publication



(a) Latence application middleware **(b)** Latence middleware application

Figure 3.24: Latence à la publication et la souscription

et $8\mu s$ à la souscription. Pour expliquer ces résultats, nous rappelons que le DataReader de chaque Topic doit consulter son cache pour récupérer les instances les plus fraîches avant de les envoyer pour l'affichage par l'application. Ceci explique le délai supérieur à la souscription aux Topics qu'à leur publication.

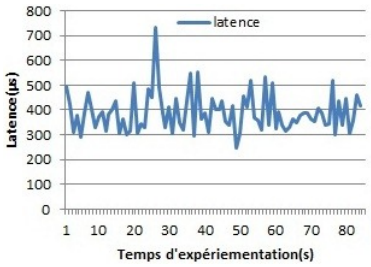
3.5.2.4 Evaluation de PlatSim_QoS du middleware DDS

Le deuxième scénario consiste à évaluer le délai de transfert de bout en bout entre deux simulations communicantes en point à point sur un réseau local. Les expériences avaient une durée de 15 minutes et ont été refaites ou moins deux fois pour s'assurer que les résultats obtenus sont constants. Pour nos tests, nous considérons le premier simulateur DDS sur EuQoS7 et le deuxième simulateur sur EuQoS8. Dans le cas où les mécanismes de QoS sont utilisés, la QoS est donc configurée au niveau du commutateur LAN, et sur chaque simulateur avec le marquage des paquets avec le champ DSCP 32.

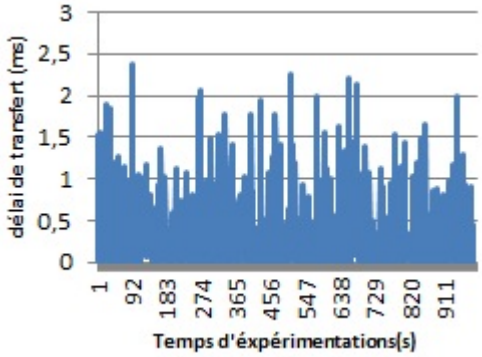
Scénario : L'objectif de cette expérimentation est d'étudier le comportement du flux DDS (ayant "EF" comme service de QoS) lorsqu'il est en compétition avec d'autres flux UDP non prioritaires. Pour cela, nous avons associé au flux DDS de l'application Platsim_QoS (ayant un débit de 400kbps) un flux UDP concurrent (ayant un débit de 100Mbps) issue du générateur de trafic *Jperf* [1] et nous avons analysé ces deux flux. Les expérimentations se sont déroulées de la façon suivante : à l'instant t=0s, nous avons commencé par l'envoi du flux DDS fiable(fiabilité est gérée au niveau middleware DDS par le biais du paramètre Reliability QOS) entre deux simulateurs Platsim_QoS (le consommateur le premier en attente du producteur), ensuite nous avons injecté un trafic UDP sans QoS pour perturber le comportement du premier flux.

Analyse : Nous allons analyser les graphiques de la Figure 3.25 décrivant le délai de transfert de l'application Platsim_QoS et celui du flux UDP concurrent. Les résultats de ces expérimentations sont illustrés sur la Figure 3.25a pour le flux DDS et sur la Figure 3.25b pour le flux

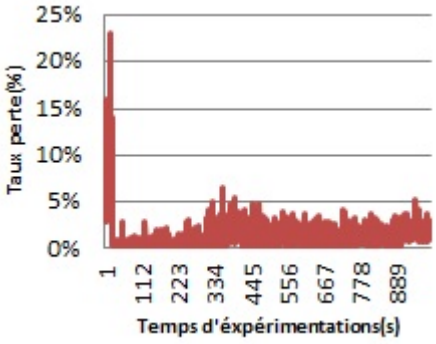
1. `http://sourceforge.net/projects/jperf/`



(a) Latence du flux DDS



(b) Latence du flux perturbateur UDP



(c) Taux de perte de paquets du flux perturbateur UDP

Figure 3.25: Latence de transfert observé pour une communication point à point

UDP. En effet, le taux de perte du flux DDS est nul pour tous les tests que nous avons réalisé, et ceci provient du fait que la QoS DDS "RELIABILITY" est fixée à fiable. Par contre, nous avons remarqué que le flux UDP issu du générateur de trafic présente un taux de perte moyen de 5% comme le montre la Figure 3.25c.
En plus, la latence moyenne constatée sur Platsim_QoS varie entre $300\mu s$ et $500\mu s$, alors que la latence du flux UDP concurrent est entre $500\mu s$ et $1000ms$ pour toutes les expérimentations. En effet, en plus de la perte des paquets, le flux UDP montre les plus grands délais. Ceci s'explique par le fait que 75% des files d'attentes sont allouées au flux DDS alors le flux UDP concurrent est retardé, voir supprimé dans certains cas, au niveau du commutateur pour permettre le traitement des paquets DDS plus prioritaires.

3.5.2.5 Impact du nombre de simulateurs sur les performances

Cette partie concerne l'étude de l'impact de l'augmentation du nombre de simulateurs sur la latence de Platsim_QoS dans la cadre des réseaux locaux. Pour cela, nous nous sommes placés dans le cadre des mêmes conditions que les expérimentations précédentes pour un seul simulateur, mais cette fois nous augmentons progressivement le nombre de participants. Ces expérimentations avaient une durée allant de 15 minutes jusqu'à deux jours. Nous allons analyser ces tests dans le reste de cette section.

Un publisher vers plusieurs subscribers : Dans cette expérimentation, nous considérons un seul simulateur producteur des Topics Platsim_QoS et nous augmentons le nombre de simu-

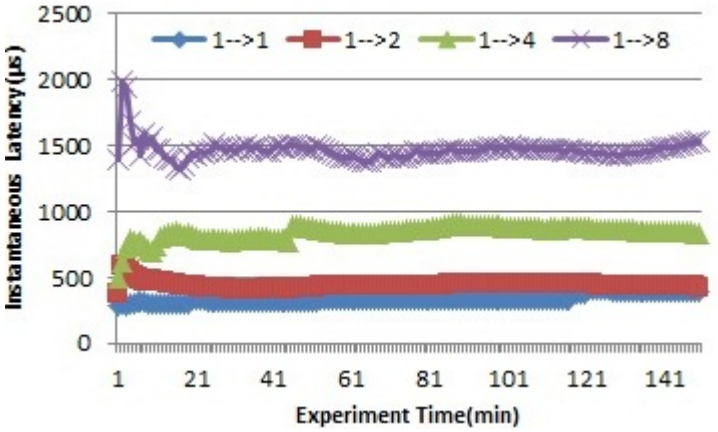


Figure 3.26: Délai de transfert en communication 1 vers 2 et 4 simulateurs

lateurs consommateurs de ces Topics progressivement de 2, 4 et 8. La Figure 3.26 montre la latence observée pour ces différentes configurations.

Analyse :

Nous avons remarqué que la latence moyenne dépend du nombre de simulateurs inscrits aux Topics. Par exemple, lorsque nous avons considéré une communication entre deux simulateurs, la latence obtenue est en moyenne $380\mu s$. Lorsque le nombre de simulateurs consommateurs est devenu 2, la latence moyenne observée est $550\mu s$, elle devient $820\mu s$ pour 4 consommateurs et $1400\mu s$ pour 8 consommateurs.

Plusieurs publishers vers un subscriber : Dans cette configuration, nous augmentons progressivement le nombre de simulateurs producteurs de Topics Platsim_QoS et nous gardons un seul simulateur abonné à ces Topics. Pour cela, nous avons présenté à la Figure 3.27 la latence moyenne observée pour 2, 4 et 8 producteurs respectivement.

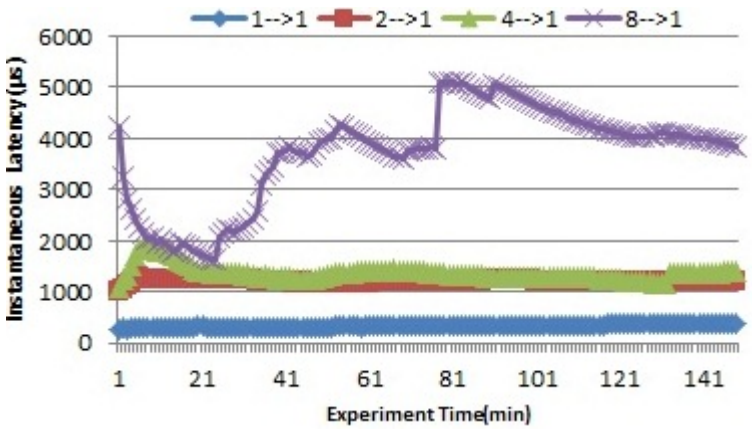


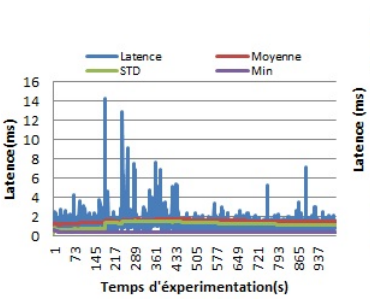
Figure 3.27: Délai de transfert en communication plusieurs vers 1

Analyse :

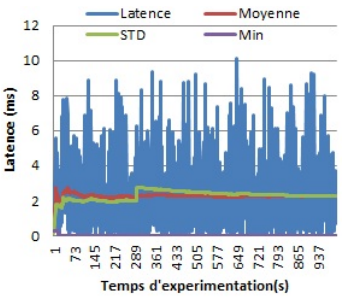
L'inspection de la Figure 3.27 montre que la latence dans le réseau local dépend du nombre

de producteurs des Topics Platsim_QoS dans la simulation. En effet, lorsque nous avons utilisé deux producteurs de topics, la latence moyenne observée est proche de $1000\mu s$. Pour 4 producteurs de topics Platsim_QoS, la latence moyenne observée est de environ $1500\mu s$, et enfin pour 8 producteurs nous avons obtenu une latence variable entre $2000\mu s$ et $4000\mu s$. Ces résultats montrent que le délai de bout-en-bout est directement lié au nombre de simulateurs à la production.

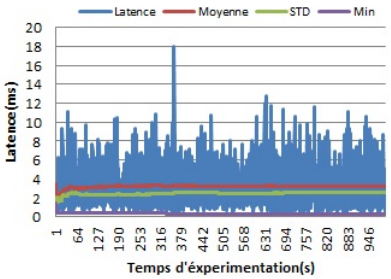
Plusieurs publishers vers plusieurs subscribers : Dans cette configuration, nous avons utilisés des simulateurs à la fois producteurs et consommateurs des Topics Platsim_QoS. Chaque simulateur génère ces Topics et s'abonne à tous les autres Topics envoyés en multicast par les autres simulateurs. Il ne s'abonne pas à son trafic puisqu'il s'abonne à tous Topics dont les clés sont différentes de la sienne. Pour cela, nous avons testé respectivement les trois configurations suivantes : le test 1 concerne la communication "2 vers 2" ; le test 2 met en jeu une communication "4 vers 4" ; et la configuration 3 concerne une communication "8 vers 8". Les résultats de ces différentes configurations sont présentés à la Figure 3.28.



(a) 2 publishers vers 2 subscribers



(b) 4 publishers vers 4 subscribers



(c) 8 publishers vers 8 subscribers

Figure 3.28: Latence moyenne pour une communication plusieurs vers plusieurs

Analyse : A partir de l'inspection de la Figure 3.28, nous avons constaté une augmentation de la latence en fonction du nombre de simulateurs. Par exemple, nous avons montré à la Figure 3.28a la latence moyenne observée sur le test 1 demeure légèrement inférieure à $2000\mu s$. La latence moyenne pour le test 2 et le test 3 est respectivement voisinage de $3000\mu s$ (Figure 3.28b) et dans le deuxième cas, elle est au voisinage de $4000\mu s$ (Figure 3.28c).

Quelques remarques importantes peuvent alors être faites concernant la dépendance entre le délai de transfert et le nombre de simulateurs mis en jeu. Nous pouvons constater en analysant ces figures que les délais de transfert dépendent du nombre de simulateurs à la publication et/ou la souscription pour toutes les configurations. Il est néanmoins à noter que ces délais sont principalement introduits par les noeuds terminaux DDS à cause de la politique de gestion des files de messages du middleware et du temps de traitement et d'attente au niveau du système d'exploitation Windows qui n'est pas un OS temps-réel.
Pour ce qui concerne le taux perte, il est calculé sur un nombre total d'échantillons de l'ordre de 3 millions. Aucune perte n'a été constatée sur le réseau en dépit de la charge globale du système ce qui justifie le choix des politiques de QoS DDS présentées dans le paragraphe 3.4.2. Le peu de pertes observées se produit au niveau des noeuds terminaux DDS et prennent effet au début de l'expérience (quelques échantillons).

3.6 Conclusion

Dans ce chapitre, nous avons présenté nos contributions pour répondre à la première problématique de la mise en réseaux locaux de la simulation distribuée, qui consiste en la spécification et l'évaluation de l'architecture de communication satisfaisant les exigences de QoS que nous avons donnée dans l'état de l'art.
Pour cela, dans la première section, nous avons redéfini les flux de données générés par l'application PlatSim que nous avons utilisés pour la réalisation des tests sur HLA reposant sur les services offerts par le réseau sous-jacent. Ces flux découlaient directement des spécifications issues de la décomposition que nous avons proposée en spécifiant les différents "sous-flux". Chaque différent sous-flux a des caractéristiques propres données par son profil de trafic applicatif augmenté des exigences applicatives. Ensuite, nous avons décrit la définition des interfaces de programmation que nous avons ajoutées pour assurer la QoS pour les applications HLA, et qui portaient dans notre cas sur les caractéristiques des flux de simulation, des flux audio et des flux vidéo.
Ensuite, nous avons présenté une proposition de l'implémentation de PlatSim_QoS sur un middleware DDS. Moins général que HLA en termes d'hétérogénéité, DDS offre par contre des fonctions évoluées et précises de garanties de QoS nécessaires pour nos simulations temps réel. Nous avons identifié le profil de trafic des flux qui circulent durant un exercice de simulation. Nous avons distingués 4 types de données selon la périodicité et le besoin en QoS de chacun d'eux : les signaux périodiques, les états, les données de contrôle et les flux multimédia. En effet, cette proposition profite des politiques de QoS fournies par DDS et associe ces politiques aux données correspondantes. Toutefois, DDS ne garantit rien si le réseau sous-jacent, celui qui va interconnecter physiquement les simulateurs, ne satisfait pas lui aussi certaines contraintes temporelles requises. Pour cela, nous avons étudié les mécanismes et outils réseau permettant d'offrir la QoS dans un réseau de simulateurs et nous avons décrit les mécanismes que nous avons déployés sur le réseau local de simulateur PlatSim_QoS.
Les prochaines sections vont être consacrées à l'interconnexion des simulateurs sur des réseaux grandes distances sans et avec des mécanismes de gestion de la QoS.

Chapitre 4

Interconnexion sur des réseaux grande distance sans QoS

4.1 Introduction

Les caractéristiques des réseaux actuels comme les longs délais, la limitation de la bande passante et l'interdiction de l'utilisation de certains protocoles comme le multicast réseau, rendent difficile voire impossible l'établissement de la communication avec des garanties que celle-ci se déroulera dans les conditions prévues à priori.
Pour cela, dans la première partie de ce chapitre, nous allons décrire à la section 4.2 notre approche de navigation à l'estime que nous avons proposé pour la prédiction et l'anticipation du comportement des objets simulés dans des réseaux qui n'offrent pas de garanties de QoS. Ensuite, nous présenterons dans la partie 4.3 une proposition pour l'interconnexion de simulation distribuée DDS et cette approche de navigation à l'estime par deux mécanismes différents : dans un premier temps, nous montrons à la section 4.3.3 qu'il est possible d'utiliser le service de routage DDS pour mettre en place un "pont-fédéré" DDS (Bridge-Federate) permettant d'interconnecter des domaines DDS différents dans un même domaine IP, et ensuite nous proposons dans la section 4.3.4 un "Proxy DDS" qui va nous permettre d'interconnecter des simulations DDS situées dans des domaines DDS différents et des domaines IP hétérogènes.

4.2 Interconnexion sur des réseaux grande distance sans QoS : Approche ANFIS Reckoning

4.2.1 Introduction

L'objectif des applications de simulation distribuée est de fournir et maintenir une vision consistante et synchronisée des entités simulées pour refléter les changements quasi-réels de leurs états. Toutefois, la mise à jour des états de ces entités génère une grande quantité d'information à faire circuler sur les liens physiques ce qui contraint la bande passante du réseau. Le support à la simulation distribué doit être capable de supporter l'ensemble du trafic et d'écouler le débit correspondant et ensuite, il doit assurer un transfert des données suffisamment rapide et suffisamment fiable afin d'avoir une simulation cohérente du point de vue des simulateurs

et de la simulation. Cette cohésion se manifeste par la représentation cohérente de l'état de toutes les entités simulées et de tout événement survenu. Toutefois, les verrous qui empêchent cet objectif parviennent de la limitation des ressources du réseau et de la variation de la QoS disponible sur les canaux de communication.
Pour résoudre ce problème, plusieurs techniques de prédiction ont été proposées dans la littérature. Dans cette section, nous proposons une technique d'anticipation et d'estimation de la position d'une entité simulée qui, en absence d'un réseau à QoS, permet une prédiction du comportement des entités simulées au sein d'une application DIS. Cette approche est basée sur l'estimation de la position d'une entité simulée par une approche dite 'Neuro-Flou Adaptative' ou ANFIS.

4.2.2 La navigation à l'estime (Dead Reckoning)

Afin de réduire le nombre de messages de mise à jour des états de ses entités, chaque site maintien, en plus de la représentation réelle, un modèle de haute fidélité (représentation extrapolée ou modèle Dead Reckoning) pour estimer localement les états des entités distribuées simulées [82]. Les états anticipés sont calculés à partir des informations d'états de leur passé en utilisant des équations d'extrapolation. L'écart entre l'état extrapolé et l'état réel ne doit pas excéder l'un des seuils définis dans le standard $IEEE1278.1a-1998$, avec Th_{pos} pour la position et Th_{or} pour l'orientation. Sur tous les autres sites distants, la réception d'un paquet contenant une nouvelle position engendre la mise à jour de l'état de l'entité concernée par une technique de dérivation polynomiale. Nous focalisons sur la dérivation du 2^{eme} ordre : si l'on dispose de la position P_i, de la vitesse V_i et de l'accélération A_i à l'instant t_i, on peut alors estimer la position, $P_{DR}(t)$, de l'entité à tout instant t$> t_i$ grâce à une extrapolation quadratique exprimée par la relation 4.1 :

$$P_{DR}(t) = P_i + V_i(t - t_i) + A_i(t - t_i)^2 \tag{4.1}$$

Ainsi, après chaque mise à jour de son entité, un site compare les valeurs réelles obtenues à partir du modèle de haute fidélité et son image extrapolée. Si la différence entre les deux modèles excède un certain seuil prédéfini, des messages de mise à jour doivent être envoyés aux sites distants. L'extrapolation de l'entité doit être corrigée par le modèle DR sur chaque site. Par conséquent, au lieu d'envoyer un message de mise à jour à chaque mouvement de l'entité distante, ces messages seront envoyés uniquement lorsque cet état excède un certain seuil prédéfinie [82]. Le filtrage de relevance concerne l'élimination de l'envoi des mises à jour inutiles. Le seuil est un paramètre crucial au sein d'un algorithme DR : d'une part, un seuil faible permet au DR de générer des messages mise à jour à haute fréquence de haute fidélité et d'autre part un DR utilisant un seuil large génère un nombre moins important des messages de mise à jour avec moins de précision.

4.2.3 Qualité de service requise par les applications

Les Exigences en QoS d'une application DIS sont exprimées en termes de paramètres de haut niveau qui spécifient le besoin de l'utilisateur. Ces exigences en QoS de la simulation dis-

tribuée englobent les spécifications pour caractériser le degré de :

- Cohérence : elle est exprimée par la cohérence spatiale et la cohérence temporelle :
  - la cohérence spatiale exige la connaissance de toute occurrence des événements survenant sur les sites distants, c'est à dire exige, que à tout moment de la simulation, l'écart entre l'état de l'entité du site émetteur S_e et du site récepteur S_r ne dépasse pas un seuil bien défini. Par exemple, sur la Figure 4.1, l'écart de la variation de la position représentée par E_P devrait remplir les conditions suivantes : $Th_{pos} \geq \| E_p \|$ et $Th_{or} \geq \| E_{or} \|$.
  - la cohérence temporelle implique la perception de l'état des entités distantes de cette erreur maximale admissible sur tous les autres sites dans un délai borné.

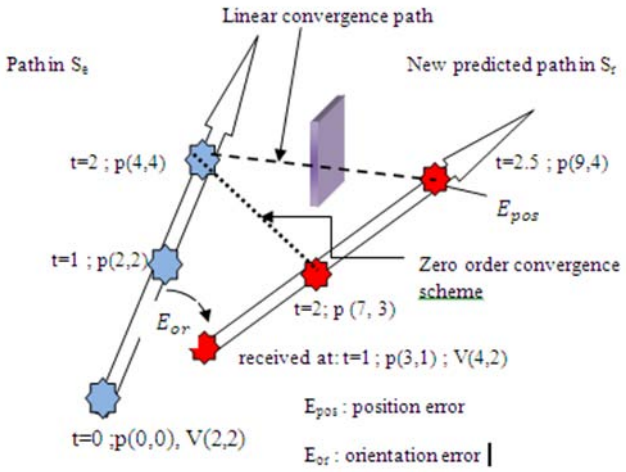


Figure 4.1: Cohérence spatiale versus cohérence temporelle

- Performance : Les caractéristiques de performance attendues nécessaires peuvent être exprimées au moyen de trois termes :
  - Fiabilité ou taux perte de messages maximum admissible, noté τ, qui est fortement lié à l'erreur d'extrapolation maximale admissible pour assurer la cohérence spatiale de la simulation.
  - Latence (Délai de transit maximum) admissible sur le réseau, noté DT_{max}, qui permet d'assurer la cohérence temporelle de la simulation.
  - Variation maximale de ce délai ou gigue, notée ΔDT.
- Couplage : le degré de couplage entre les entités est défini par :
  - Un couplage fort : Un couplage fort se manifeste lorsque plusieurs entités se trouvent dans une zone étroite et, les paquets qu'ils transmettent nécessitent des performances plus grandes pour assurer la cohérence et la consistance de la simulation. Les valeurs admissibles de ces paramètres sont : D $\leq$100 ms et $\tau \leq 2\%$ [128] .
  - Un couplage faible : ce type de couplage arrive lorsque des entités sont assez nombreuses et les distances qui les séparent sont suffisamment grandes pour tolérer les erreurs de transmission. Les exigences sont de l'ordre de D $\leq$300 ms et $\tau \leq 5\%$ [128] .

L'expression de la QoS à partir du degré du couplage entre entités présente trois limites principales :

– L1 : la détermination du degré de couplage (susceptible d'évoluer dans le temps) est coûteuse en termes de temps de calcul.
– L2 : l'expression de la QoS par le degré du couplage, c'est à dire dépendant de chaque couple (site émetteur, site récepteur), s'accorde mal avec un transport multicast.
– L3 : le standard ignore l'influence du délai de transit (latence) sur l'erreur de la position/orientation et néglige les contraintes de cohérence spatiale. Nous illustrons les limites de cette troisième contrainte.

Considérant le cas de la Figure 4.2, où deux Sites Se et Sr appartenant à un même exercice de simulation échangent des informations concernant une entité simulée qu'on note A. Nous focalisons sur le comportement aperçu de A sur le site distant en supposant que l'approche DR est utilisée pour alléger le trafic sur le réseau. La figue 4.2 illustre l'erreur d'extrapolation de la position de l'entité E_s sur le site émetteur et l'erreur E_r sur le site récepteur.

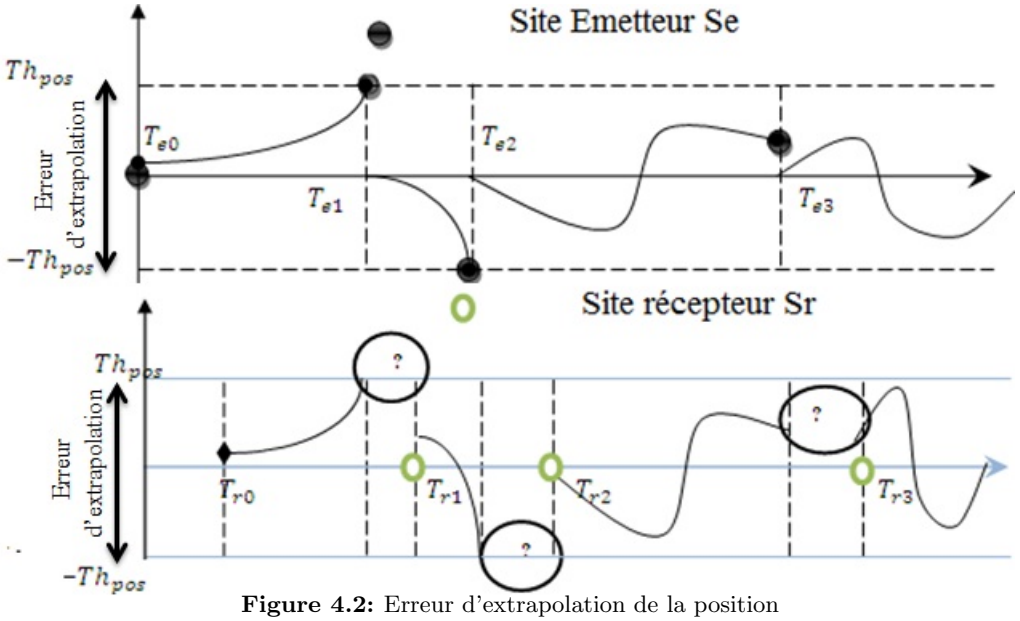


Figure 4.2: Erreur d'extrapolation de la position

Nous avons choisis, les cercles en noir pour désigner les paquets transmis par le site Se et, les cercles en couleur gris pour marquer les PDU reçus par le site récepteur Sr. Nous avons choisi pour origine des temps une date d'émission (T_{e0}).
Les dates T_{e0}, T_{e1}, T_{e2} et T_{e3} correspondent aux dates d'émission des PDUs et T_{r0}, T_{r1}, T_{r2} et T_{r3} les dates de réception de ces PDUs.

- A partir de la date T_{e0}, l'erreur d'extrapolation de la position E_s augmente et atteint la valeur maximale (Th_{pos} ou seuil DR) à la date T_{e1} : le mécanisme DR provoque alors l'émission d'un PDU contenant la position réelle de A : l'erreur E_s s'annule ;
- A partir de T_{e1}, le scénario précédent se reproduit jusqu'à la date T_{e2} ;
- A partir de T_{e2}, l'erreur oscille entre Th_{pos} et $-Th_{pos}$ sans jamais quitter l'intervalle autorisé, au bout du HEART_BEAT_TIMER, soit T_{e3}=T_{e2} + 5s, le DR émet un PDU et l'erreur revient à zéro.

Analysons l'évolution de l'erreur E_r sur le site récepteur :

– A partir de T_{r0}, l'erreur d'extrapolation de la position E_r commise sur le site récepteur est identique à celle faite au même instant sur le site émetteur S_e ; en particulier elle atteint la valeur maximale admissible (Th_{pos} à T_{e1}, date d'émission du paquet de rafraîchissement de A. Cependant, la mise à jour de la position de A n'est effective qu'à la date T_{r1}, l'intervalle séparant T_{e1} et T_{r1} correspondant au délai de transit à travers le réseau. Il apparaît donc une indétermination quant à la valeur de E entre les instants T_{e1} et T_{r1}, période durant laquelle E_r peut dépasser la valeur du seuil DR (Th_{pos}) engendrant une violation de la cohérence spatiale ;
– de manière générale, on met en évidence une absence de maîtrise de E_r durant les intervalles de temps indiqués par un point d'interrogation de la Figure 4.3.

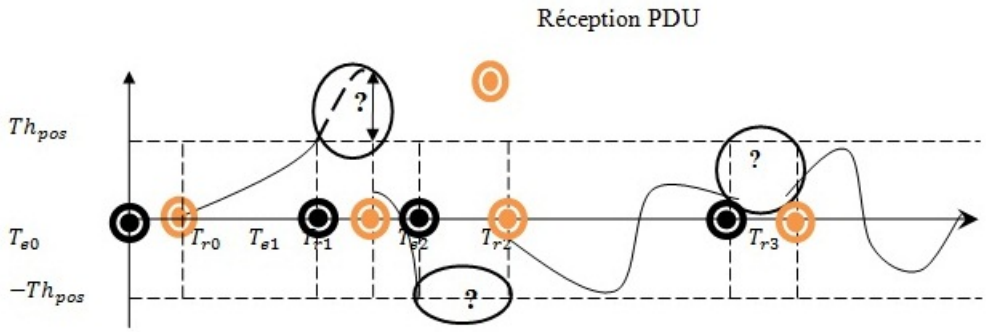


Figure 4.3: Excès de l'interpolation du déplacement de l'entité

Deux questions se posent alors : (1) l'absence de garantie de cohérence spatiale est-elle funeste au déroulement correcte d'un exercice de simulation distribuée interactive ? (2) si oui, peut-on remédier à ce problème, c'est-à-dire maîtriser l'excès transitoire de E_r ?
Concernant la question (1), l'excès transitoire potentiellement observable devient préjudiciable dès lors que le délai de transit DT n'est plus négligeable devant l'intervalle de temps séparant deux réceptions consécutives de paquets dead reckoning relatifs à une même entité. Si c'est le cas, les sites récepteurs n'auront plus une vue spatiale cohérente de l'état de l'entité durant la période de temps non négligeable, au pire "presque tout le temps" si la période de rafraîchissement de l'état d'une entité est voisine au inférieure au délai de transit des paquets qui lui sont associés dans le réseau. Ce risque qui n'apparaît pas dans les réseaux locaux, peut apparaître de façon plus importante dans les réseaux grande distance.
L'erreur d'extrapolation de la position sur un site récepteur n'est inférieure au seuil Th_{pos} que dans l'intervalle de temps [T_{ei}+DT, T_{ei+1}]. Dans l'intervalle [T_{ei}, T_{ei}+DT], l'erreur peut excéder de manière drastique et conduire à une incohérence. On s'aperçoit que l'excès transitoire E_T dépend de la dynamique du vecteur accélération de l'entité simulée, et par conséquent, le délai de transit maximal D_{max} admissible pour garantir une valeur maximale de l'erreur E_{max} dépend lui aussi de la dynamique de l'entité. Une façon de garantir une borne supérieure Er_{max} à l'excès transitoire Er est de disposer d'une garantie de la part du réseau que le délai de transition DT des paquets ne dépasse pas une valeur limite DT_{max}, comme l'illustre la Figure 4.4.
L'influence du délai de transit DT sur la connaissance de l'écart des entités simulées par chaque site est liée à l'expression de l'écart e_p(t) (en valeur absolue) entre la position réelle P_A(t) de l'entité et sa position extrapolée (extrapolation de second ordre) par le mécanisme du Dead

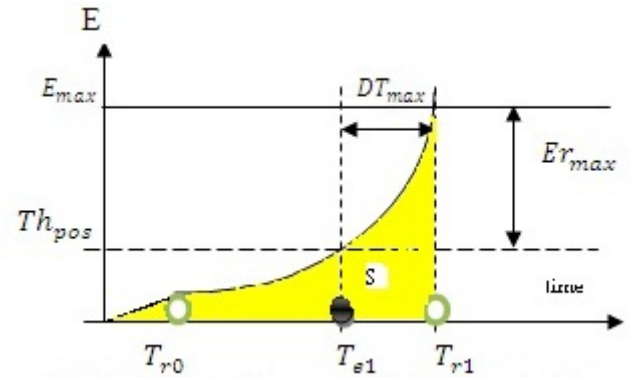


Figure 4.4: Garantie de l'excès transitoire maximal

Reckoning P_{DR}(t) (relation 4.3 et relation 4.2).

$$e_p(t) = \| P_{DR}(t) - P_A(t) \| \leq E_{max}; E_{max} \geq 0 \tag{4.2}$$

$$\begin{aligned} e_p(Te_{i+1} + DT) = & \| \int_{Te_i}^{Te_{i+1}} du \int_{Te_i}^{Te_{i+1}} [A_a(\tau) - A_i] d\tau \| \\ & + \| \int_{Te_i}^{Te_{i+1}+DT} du \int_{Te_{i+1}}^{Te_{i+1}+DT} [A_a(\tau) - A_i] d\tau \| + \\ & + \| \int_{Te_i}^{Te_{i+1}+DT} du \int_{Te_{i+}}^{u} [A_a(\tau) - A_i] d\tau \| \leq M \end{aligned} \tag{4.3}$$

Notons que $A_a(t)$ et $A_i(t)$ sont respectivement l'accélération instantanée de l'entité simulée à tout moment (t) et son accélération au moment T_{ei}, et en plus les termes de l'equation 4.3 sont :

- le premier terme est majoré par le terme du Th_{pos}, qui est un terme constant ;
- le second terme est à déterminer à partir de la différence entre la vitesse effective et la vitesse extrapolée multipliée par le terme DT exprimant le délai de transit, que nous écrivons aussi par la relation 4.4.

$$\int_{Te_i}^{Te_{i+1}} [A_a(\tau) - A_i] d\tau = [(V_{i+1} - V_i) - V_i(Te_{i+1} - Te_i)] = [V_a(Te_{i+1}) - V_{DR}(Te_{i+1})] \tag{4.4}$$

- Enfin, le troisième terme peut être déterminé par $\frac{A_{max} \times DT^2}{2}$ pour une extrapolation linéaire ou $A_{max} \times DT^2$ pour une extrapolation quadratique.

Par conséquent, le terme de différence d'accélération étant maximal et constant sur $[Te_i, Te_{i+1}]$, et par la suite, majoré par le $\sqrt{2A_{max} \times Th_{pos}} \times DT$ pour l'extrapolation linéaire et $2\sqrt{A_{max} \times Th_{pos}} \times DT$ pour l'extrapolation quadratique. Nous pouvons donc obtenir une borne supérieure de l'erreur instantanée, c'est à dire si nous voulons garantir un délai de transit respectant les contraintes de QoS, nous devons alors garantir que l'erreur d'extrapolation à la réception E_r soit inférieur à une borne maximale ET_{max}. Faisons l'hypothèse que le délai de transit de bout en bout a une valeur bornée fonction de l'accélération maximale de l'entité associée par la relation 4.5 pour une extrapolation linéaire ou par la relation 4.6 pour une relation quadratique :

$$DT_{max} \leq \frac{\sqrt{Th_{pos} + ET_{max}} - \sqrt{Thpos}}{\sqrt{1/2 \times A_{max}}} \tag{4.5}$$

$$DT_{max} \leq \frac{\sqrt{Th_{pos} + ET_{max}} - \sqrt{Thpos}}{\sqrt{A_{max}}} \tag{4.6}$$

Ainsi, nous pouvons assurer une cohérence spatiale, et par conséquent, temporelle (vu la forte corrélation entre le déplacement spatial et la variation instantanée de la position, vitesse et accélération de l'entité simulée) en garantissant la position dans un intervalle inférieur à $Th_{pos} + E_{max}$, ce qui implique la garantie d'un délai de transit de bout-en-bout maximal DT_{max}.
Pour la deuxième question, plusieurs travaux présentaient des moyens de mettre en oeuvre la maîtrise de l'excès transitoire [7]. Dans [77] deux algorithmes adaptatifs sont décrits. Le premier est basé sur un mécanisme d'ajustement adaptatif du niveau du seuil pour contrôler l'erreur de l'extrapolation et le second algorithme est basé sur la sélection automatique de l'équation de l'extrapolation pour assurer le contrôle des entités pendant la simulation. L'inconvénient de cette approche est qu'elle a besoin de beaucoup de détails sur les entités et leur environnement pour intégrer les deux mécanismes de contrôle.
Taylor [129] présente un estimateur déterministe de positions des objets de la simulation. Il utilise des événements déterministes pour établir un algorithme de prédiction permettant d'estimer le comportement d'un mobile en déplacement. Les inconvénients de ce modèle est que le comportement des entités simulées est décrit par des équations mathématiques strictes. Si les entités ont un comportement imprévisible, alors l'erreur de l'estimation sera plus grande.
Les auteurs dans [75] ont proposé une approche basée sur le filtre de Kalman pour l'estimation des paramètres des entités mobiles dans un réseau ad-hoc afin de réduire le trafic entre les différents noeuds. Bien qu'elle permette de gagner 10% de la bande passante du réseau, cette approche ne corrige pas efficacement l'erreur d'extrapolation.
Dans [27] une approche d'estimation de la position basée sur la logique flou est appliqué dans une simulation HLA. Cet algorithme prend en compte le degré de corrélation flou lors des mesures des relations entre les entités (position, taille, angle de vision,...) et considère une approche multi-niveaux pour estimer le niveau du seuil que l'application doit respecter. Également, une approche basée sur les réseaux de neurones pour améliorer l'extrapolation du prédicateur des entités simulées a été proposée [1]. Un banc de réseaux de neurones (prédicateur de position, de vitesse, d'orientation et de vélocité) permet d'estimer la nouvelle propriété estimée de l'entité.
Toutefois, chacune de ces techniques d'intelligence artificielle possède des propriétés particulières qui la rendent bien adaptée à des problèmes particuliers. Par exemple, l'estimation par filtre de Kalman est bien adaptée pour la prédiction du comportement des entités simulées en leurs associant des observateurs d'états, mais elle nécessite beaucoup de calcul pour le faire. Les réseaux de neurones sont très intéressants pour la reconnaissance et l'apprentissage des trajectoires, mais ils sont mauvais lorsqu'il s'agit de prendre les décisions portant sur le choix du seuil admissible de l'erreur. La logique flou, qui raisonne sur des informations vagues et imprécises, est bien adaptée pour la prise de décision sur le choix du seuil de l'erreur admissible, mais ne peut pas automatiser le choix des règles pour prendre ces décisions. Ces limitations ont été la motivation principale pour la création de systèmes intelligents hybrides, où ces deux dernières techniques sont conjuguées pour surmonter les limitations individuelles de chacune d'entre elles.
Notre approche permet de réduire le délai pendant lequel l'information est utilisable en bornant l'erreur maximale admissible, et ce jusqu'à ce que la nouvelle position soit mise à jour par un nouveau paquet. La position estimée est maintenant calculée par un réseau neuro-flou adaptatif préconfiguré [121]. Notre approche implique, tout d'abord, l'abandon de la notion de couplage et

ensuite, la détermination de la valeur de l'erreur à partir des seules informations contenues dans chaque paquet de mise à jour des états des entités. Cette proposition remédie à la contrainte L1 de manière à garantir à tout instant une erreur maximale admissible, en particulier durant les périodes transitoires, et cela permet de remédier aussi à la limite L3.

4.2.4 Modèle théorique de l'approche ANFIS Reckoning

De cette partie, nous proposons un algorithme neuro-flou adaptatif (Adaptive Neuro-Fuzzy Inference System :ANFIS) qui nous servira pour construire le modèle de haute fidélité qui se rapproche le plus proche possible de la réalité. Cet algorithme permet d'émuler le comportement humain lors de la détermination de l'erreur maximale admissible satisfaisant les contraintes de la QoS évoqués précédemment.

4.2.4.1 Définitions utiles

Pour des raisons de clarté, nous commençons par quelques définissions utiles pour introduire le concept du raisonnement flou.

Raisonnement flou : Dans un ensemble flou le passage entre les termes "appartient à" et "n'appartient pas à " est graduel et ce passage lisse est caractérisé par des fonctions d'appartenances qui permettent de donner plus de flexibilité pour modéliser les termes linguistiques. La logique flou définit deux concepts important [12] :

- implication floue : la notion d'implication floue (une implication est de la forme "Si A, alors B").
- Modus Ponens généralisé : le modus Ponens est une règle de déduction (dite aussi implication flou) telle que si on a : $A => B$ et A, alors on peut en déduire B.

Définition1 : Si U est une collection d'objets, dite aussi univers de discours, notée par x, alors l'ensemble flou A dans U est défini par un ensemble ordonné de paires :

$$A = (x, \mu_A(x))|x \in U \tag{4.7}$$

Avec : $\mu_A(x)$ est dit la fonction d'appartenance (Membership Function ou MF) de x dans A. Cette MF translate chaque élément de U en des valeurs continues entre 0 et 1. Toutefois il existe plusieurs fonction permettant de paramétrer les fonctions d'appartenance, par exemple les fonctions triangulaires, les fonctions rectangulaires, les fonctions trapézoïdales, etc. [12].

Définition2 : Soit U un ensemble défini sur $\Re^n$. La transformation de distance (DT) de S est donnée comme étant l'image $(x,\ D_s(x))|x \in \Re^n$ dans $\Re^n$ où D_s est l'ensemble de valeurs de DT dans x définis par la relation 4.8, avec inf est une borne inférieure pour tout ensemble non vide dans $\Re^n$ et $\|.\|$ est une norme euclidienne.

$$D_s(x,\ y) = inf\|x - y\|; avec\ \ x,\ y \in \Re^n \tag{4.8}$$

4.2.4.2 Architecture du modèle :

Le modèle d'inférences neuro-flou adaptatif est basé, tout d'abord, sur un système d'inférence flou pour la formulation de termes linguistiques de règles flous. Ces règles sont entraînées à l'aide un algorithme d'apprentissage par réseaux de neurones artificiels pour automatiser le choix du seuil du dead reckoning des entités distribuées simulées. Pour simplifier l'illustration du modèle, on se contente d'utiliser le modèle de second ordre pour estimer la position d'une entité en se basant sur ces paramètres de corrélation flou (position, vélocité, accélération).

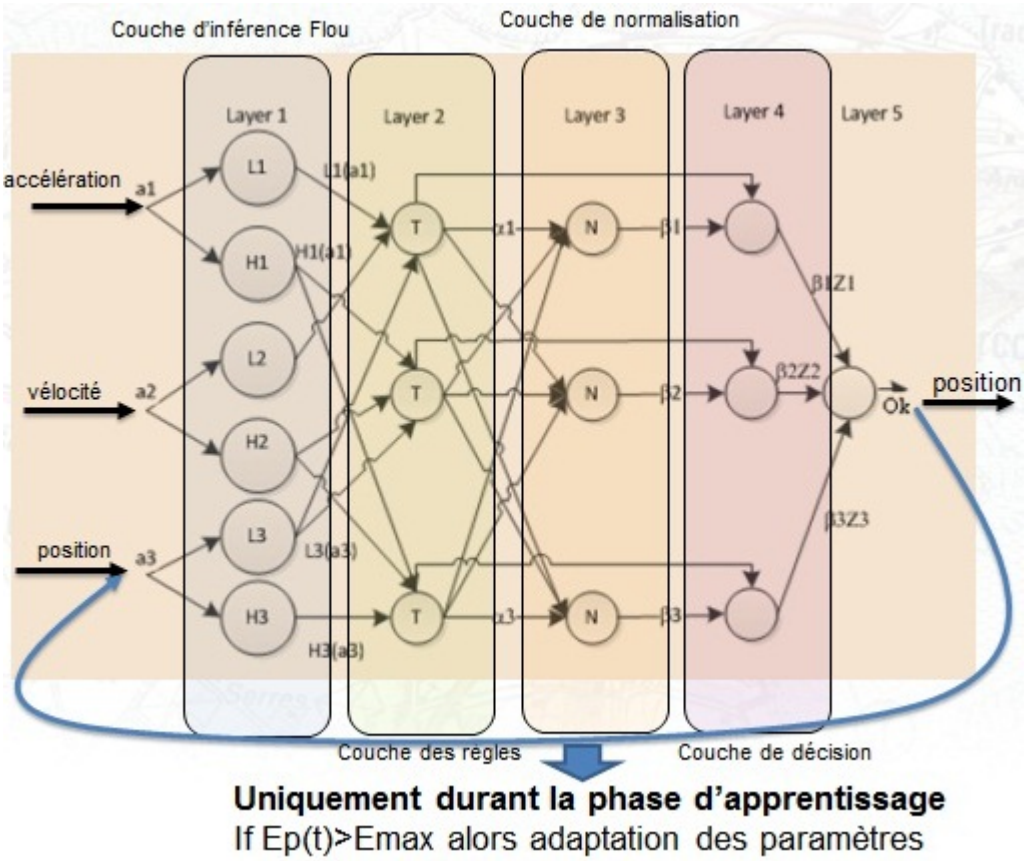


Figure 4.5: Système ANFIS à une entrée et une sortie

Le réseau ANFIS utilisé dans cet algorithme est composé de cinq couches, comme le décrit la Figure 4.5, avec six noeuds dans la première couche qui représente la dimension de chacun des vecteurs d'entrée (trois entrée (a1, a2, a3)), un noeud dans la dernière couche qui représente la sortie (O), et 3 couches cachées composées de trois noeuds dans chaque couche. Ce réseau vise à développer les fonctions liant les vecteurs d'entrée aux vecteurs de sortie en utilisant l'algorithme d'entraînement neuro-flou adaptive. La couche 1 est la couche d'entrée des paramètres (position, vitesse, angle) dite aussi couche d'inférence flou, la couche 2 est la couche des règles, la couche 3 est la couche de normalisation, la couche 4 est la couche de prise de décision et la couche 5 est la couche de sortie de la nouvelle position estimée.

La couche d'entrée : Dans notre exemple a_1 représente la position, a_2 représente la vitesse et a_3 représente l'orientation. La fonction de mapping qui permet d'avoir la sortie O est donnée par l'équation 4.9 :

$$f^k = f(a^k) = f(a_1^k, a_2^k, ..., a_n^k); \tag{4.9}$$

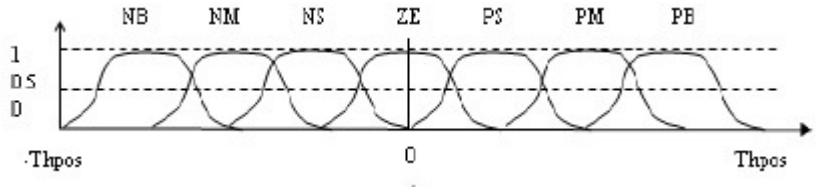
Avec : $k \in 1..K$ (temps discret) et les ensembles d'entraînement flou sont donnés par la relation 4.10 :

$$\{(a^1, a^2, a^3, f^1), ..., (a_1^k, a_2^k, a_3^k, f^k)\} \tag{4.10}$$

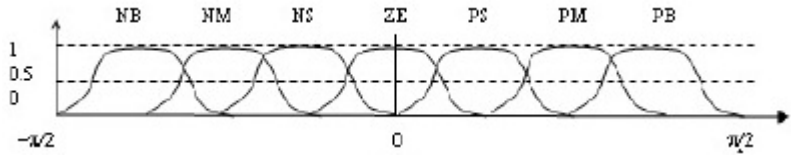
La sortie d'un noeud est le degré d'appartenance de l'entrée satisfaisant les variables linguistiques associées à ce noeud (Cf. Figure 4.5). La méthode qui permet de décrire les termes linguistiques, les fonctions d'appartenances et les règles des prémisses consiste à utiliser les règles SI "A" ALORS "B" des relations suivantes :

$$(S_1)\begin{cases} R_1 = SI\ \ a_1^1\ est\ A_{i1} et ... et\ a_1^n\ est\ A_{in}\ ALORS\ f\ =\ z_1 \\ R_2 = SI\ \ a_2^1\ est\ B_{i1} et ... et\ a_2^n\ est\ B_{in}\ ALORS\ f\ =\ z_2 \\ R_3 = SI\ \ a_3^1\ est\ C_{i1} et ... et\ a_3^n\ est\ C_{in}\ ALORS\ f\ =\ z_3 \end{cases}$$ Avec : i = 1,..m, A_{in} et B_{in} les degrés d'appartenances de la fonction d'appartenance et z_i des nombres réels.

Nous avons choisi pour chaque variable linguistique 7 termes linguistiques (NB pour Negative Big ; NM pour Negative Medium ; NS pour Negative Small ; ZE pour ZERO ; PS pour Positive Small ; PM pour Positive Medium et PB pour Positive Big). La Figure 4.6a montre les termes linguistiques de la variable position qui représentent le déplacement de l'entité par rapport à sa position initiale (dégrée de corrélation flou) dans l'interval $[-Th_{pos}, Th_{pos}]$. Pour la variable orientation, les termes linguistiques décrivent l'angle d'orientation (Figure 4.6b) d'une entité simulée dans l'intervalle $[\frac{-\pi}{2}, \frac{\pi}{2}]$.



(a) position



(b) orientation

Figure 4.6: Termes linguistiques de la position et l'orientation

Le Tableau 4.1 illustre la variation des termes linguistiques en fonction de la position de l'entité simulée dans l'intervalle $[-Th_{pos}, Th_{pos}]$. Par exemple, lorsque l'entité est à proximité de Th_{pos} et le terme linguistique de sa position est "PB", la valeur de la nouvelle position prédite est autour de 90% du seuil. Cette position nécessite alors une correction immédiate de la position pour la ramener au plus proche du terme 0%. Pour cela, l'algorithme calcule en permanence le nouvel état de l'entité à partir des valeurs actuelles de la position, l'orientation, la vitesse et l'accélération (déduite de la vitesse) en utilisant le modèle "Takagi-Sugeno" de la Figure 4.7.

Valeurs des nombres flous							
Distance(m)	NB	NM	NS	ZE	PS	PM	PB
$-Thpos$	85%	65%	0%	0%	0%	0%	0%
$-2Thpos/3$	25%	70%	40%	0%	0%	0%	0%
$-Thpos/3$	0%	35%	70%	10%	0%	0%	0%
0	0%	0%	30%	90%	30%	0%	0%
$Thpos/3$	0%	0%	0%	25%	90%	30%	0%
$2Thpos/3$	0%	0%	0%	0%	30%	85%	35%
$Thpos$	0%	0%	0%	0%	20%	40%	90%

Table 4.1: valeurs des membres flou calculés sur la couche d'inférence flou

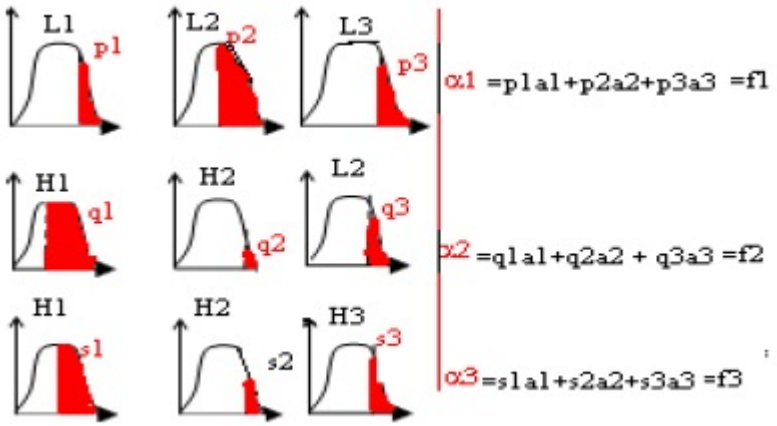


Figure 4.7: Termes linguistiques avec 3 règles pour a) les variables position et vélocité, b) pour la variable orientation

L'inspection de la Figure 4.7 montre que les nouvelles valeurs de la position deviennent vagues et imprécises, ce qui rend la décision de choisir la valeur ainsi trouvée incertaine. En particulier, les informations actuelles dont nous disposons sur les différents résultats possibles à appliquer sur les sorties et leurs nouvelles mises à jours sont ambiguës, puisque les conditions changent avec une dynamique rapide. Pour cela, la couche 2 se chargera de régler ces valeurs pour limiter ces imprécisions.

La couche de Règles : chaque noeud de cette couche réalise une fonction de normalisation dite "T-norme". Les sorties respectives de chaque noeud, appelés "rule nodes" (haut, milieu et bas) sont données par le système d'équations 4.11 (Cf. Figure 4.7) :

$$\begin{aligned} \alpha_1 &= L1(a1) \wedge L2(a2) \wedge L3(a3) \\ \alpha_2 &= H1(a1) \wedge H2(a2) \wedge L3(a3) \\ \alpha_2 &= H1(a1) \wedge H2(a2) \wedge H3(a3) \end{aligned} \tag{4.11}$$

La couche de normalisation : Les sorties des T-normes sont normalisées. Après combinaison linéaire des variables d'entrée dans la couche 2, la sortie prédite est obtenue dans la couche 3 par une moyenne pondérée des sorties des différentes règles, comme le présente les

relations 4.12 :

$$\begin{aligned} \beta_1 &= \frac{\alpha_1}{\alpha_1+\alpha_2+\alpha_3} \\ \beta_2 &= \frac{\alpha_2}{\alpha_1+\alpha_2+\alpha_3} \\ \beta_3 &= \frac{\alpha_3}{\alpha_1+\alpha_2+\alpha_3} \end{aligned} \tag{4.12}$$

La couche de décision : la phase d'apprentissage se réalise dans cette couche. Les données stockées dans une base de données de l'expert sont comparées à celles issues des couches précédentes pour réaliser l'apprentissage hors ligne. Une autre alternative (non discutée ici) correspond à l'apprentissage en ligne qui consiste à comparer les données au fur à mesure de l'évolution de l'algorithme. La sortie de chacun des trois neurones est le produit normalisé de la couche du tir (niveau apprentissage) et de la règle particulière correspondante, données par le système d'équations 4.13 :

$$\begin{aligned} \beta_1 z_1 &= \beta_1 \vee B^-1(\alpha_1) \\ \beta_2 z_2 &= \beta_2 \vee B^-1(\alpha_2) \\ \beta_3 z_3 &= \beta_3 \vee B^-1(\alpha_3) \end{aligned} \tag{4.13}$$

La couche de sortie : un seul noeud calcule la sortie du système en réalisant la somme de toutes les entrées, comme le démontre l'équation 4.14. La valeur obtenue à la couche de sortie est comparée aux valeurs précédentes pour décider si celle-ci pourrait être retenue comme la valeur estimée ou bien l'algorithme doit reboucler pour avoir une estimation meilleure.

$$O_k = \beta_1 z_1 + \beta_2 z_2 + \beta_3 z_3 \tag{4.14}$$

Pour cela, nous avons défini l'erreur quadratique qui permet de calculer la différence entre les valeurs de la position de l'entité entre les instants k et $k+1$ (l'équation 4.15), avec : y_k est k^{me} composant du p^{me} vecteur de sortie désiré et O_k est le composant actuel du vecteur de sortie réel produit suite au retour de la p^{me} sortie dans le vecteur d'entrée dans le réseau ANFIS Reckoning aux instants k=1,...,K..

$$e_p(t, t+DT) = E_k = \frac{1}{2} \times (y_k - o_k)^2 \tag{4.15}$$

Enfin, La méthode du gradient descendant de la relation 4.16 est utilisée pour l'apprentissage des paramètres des parties conditions et conséquences des règles flous. Plutôt que de choisir les paramètres associés à une fonction d'appartenance donnée de manière statique par le système d'inférence flou générique, ces paramètres sont choisis de manière à adapter les fonctions d'appartenance pour les données d'entrée/sortie afin de tenir compte de ces types de variations dans les valeurs des données (les états des entités). La méthode d'apprentissage neuro-adaptative fonctionne de manière similaire à celle des réseaux neuronaux.

$$b_i(t+1) = b_i(t) - \eta \times \frac{\partial E}{\partial b_4} = b_i(t) - \frac{\eta}{b_i^2} \times \delta_k \frac{\alpha_1+\alpha_2-\alpha_3}{\alpha_1+\alpha_2+\alpha_3} \tag{4.16}$$

δ_k=($y_k - o_k$) dénote l'erreur. η est le pas d'apprentissage (toujours positif) et t le nombre. Cet algorithme construit un système d'inférence floue, dont les paramètres des fonctions d'appartenances sont ajustés en utilisant l'algorithme de rétro-propagation du gradient. Le modèle ANFIS réussit à réduire à la fois la cohérence spatiale et la cohérence temporelle associées à l'entité distante.

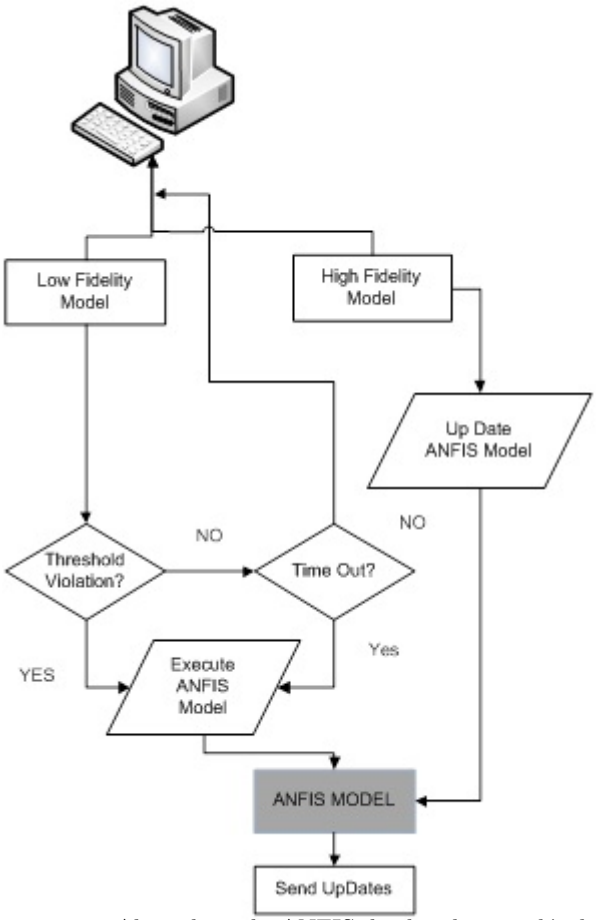


Figure 4.8: Algorithme du ANFIS dead reckoning développé

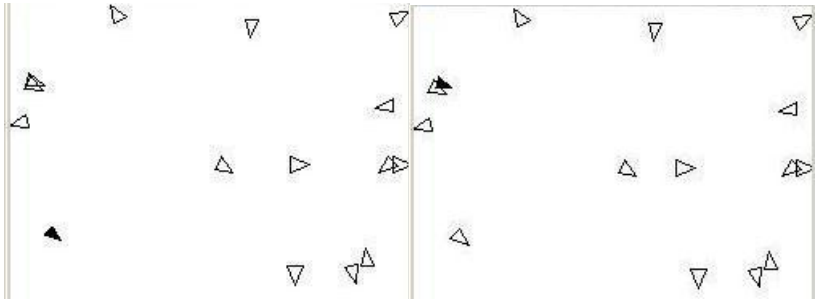
4.2.4.3 Implémentation de L'algorithme

La Figure 4.8 montre le diagramme du ANFIS appliqué sur chaque noeud : le modèle de haute fidélité est le modèle que l'entité doit suivre. Sur la base de ce modèle, le modèle à faible fidélité est donc lié au modèle de haute fidélité afin de vérifier chaque fois si le seuil d'erreur Th_{pos} peut atteindre sa valeur maximale. Ensuite, le modèle ANFIS est déclenché et le Dead Reckoning est appliqué. Ce processus est appliqué en boucle jusqu'à ce que la 5^{me} couche de sortie (l'erreur de sortie) est simplement acceptée par le modèle.

4.2.5 Impact du ANFIS Reckoning sur l'erreur d'extrapolation

4.2.5.1 Evaluation de l'algorithme

La Figure 4.9 montre l'exemple de simulation distribué entre un site émetteur (Figure 4.9a) et un site récepteur (Figure 4.9b). Dans cet exemple, nous avons 16 entités simulées qui représentent 16 simulateurs distribuées partageant le même environnement virtuel. Les triangles noirs sur la Figure 4.9a décrivent l'entité réelle gérée par le simulateur local, et le triangle blanc dans un emplacement quasi-identique sur la Figure 4.9b montre son image extrapolée.



(a) Site émetteur **(b)** Site récepteur

Figure 4.9: Les entités simulées sur le site émetteur et sur le site récepteur

Evaluation de l'erreur de position : Afin d'évaluer la performance de notre approche ANFIS Reckoning, nous l'avons comparée avec deux autres approches : l'approche dead reckoning basée sur l'historique de position [124] [123] présentée dans la Figure 4.10, et l'approche dead reckoning adaptative [77] présentée dans la Figure 4.11.
Du fait que l'approche basée sur l'historique de position est utilisée pour le suivi de la trajectoire en utilisant uniquement la dernière valeur pour l'estimation position future, elle calcule un seuil très serré qui nécessite des taux élevés de rafraîchissement de paquets.
L'algorithme utilisant le seuil multi-niveaux est utilisé pour remédier à ces problèmes et pour contrôler la précision de l'extrapolation et l'influence de la fréquence de rafraîchissement des paquets : un seuil important favorise la génération de paquets plus fréquemment, et une petite valeur du seuil génère moins de paquets à envoyer avec une fréquence de rafraîchissement plus faible.
La Figure 4.12 illustre la trajectoire de l'entité lorsque le seuil est adapté en utilisant le modèle

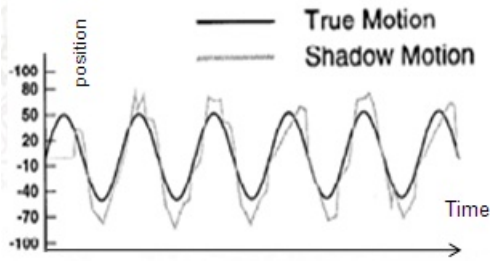


Figure 4.10: Erreur d'extrapolation par l'approche Historique

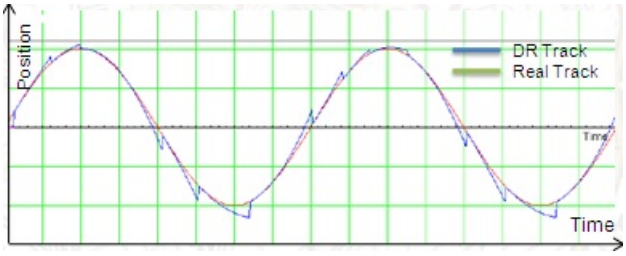


Figure 4.11: Erreur d'extrapolation par l'approche seuil multi-niveaux

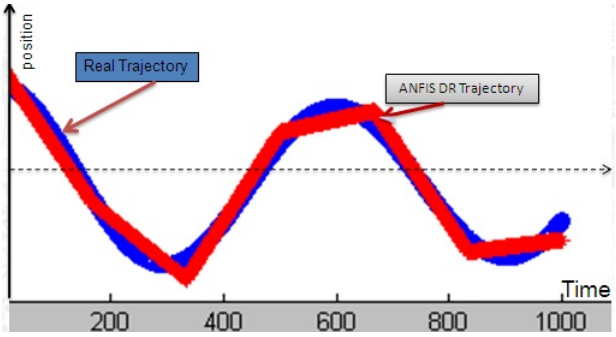


Figure 4.12: Erreur d'extrapolation par l'approche ANFIS Reckoning

ANFIS Reckoning. Lorsque l'entité se rapproche de la trajectoire réelle l'erreur extrapolée devient plus petite. Un ajustement optimal est fait lors de l'extrapolation, la rendant très proche du chemin réel de l'entité. La fréquence de mise à jour est très faible, le nombre de paquets de mise à jour demeure également très faible. Par conséquent, l'utilisation de cette approche intelligente pour adapter dynamiquement le seuil d'erreur fournit des résultats meilleurs que l'approche multi-niveau. Cette approche permet alors de mieux respecter les contraintes de QoS présentées dans la section 4.2.3 en permettant un rafraîchissement moins important de la trajectoire.

Ces résultats nous permettent de noter que le modèle DR ANFIS a des résultats légèrement meilleurs que celles trouvés dans l'approche multi-niveau. L'erreur demeure proche de 0 et l'algorithme de prédiction converge au bout de 10 itérations. En effet, les valeurs initiales des paramètres des prémisses choisies lors de la phase de fuzzification (Fuzzification, Inférences floues et Défuzzification) à l'entrée de la couche 1 sont suffisamment bien réparties pour recouvrir la trajectoire de l'entité. Ceci est expliqué par le fait que pour chaque entrée du système ANFIS il existe au moins une condition exhaustive dite ϵ Completeness [53], qui veux dire que $\mu_A(\mathrm{x}) \geq \epsilon = 0{,}5$. Il y a une assurance que l'entité ne pourra éventuellement pas heurter un obstacle puisqu'elle suit la trajectoire presque réelle. De plus, ces messages de correction générés par le site l'émetteur sont réduits aux moments de détection de l'erreur.

A partir de ces résultats, il apparaît que le banc de réseau de neuronal entraîné par les entrées floues permet une meilleure consistance dans les tests en le comparant aux autres approches.

La Figure 4.13 illustre l'erreur de seuil effectuée par trois approches : la prédiction de l'erreur de seuil à l'aide du modèle ANFIS donne une réduction de l'erreur moyenne jusqu'à 0.308m. Les deux autres résultats ont également été prises à partir de [77] : l'erreur est d'environ de

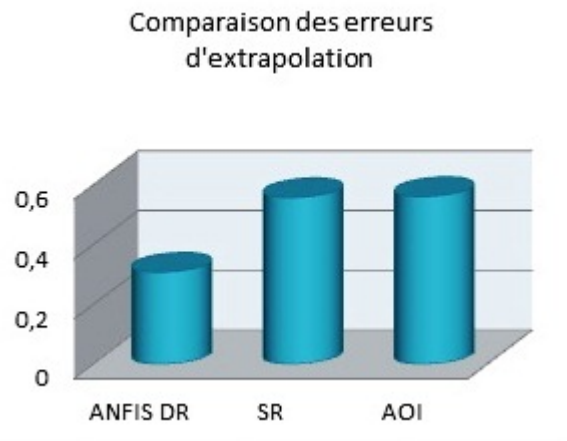


Figure 4.13: Comparaison des erreurs d'extrapolation avec l'approche ANFIS Reckoning

0,5608 m pour l'approche de zone d'intérêt (AOI ou area Of Interest) et 0,5573 pour la région sensible (Sensitive Region (SR)).

Nous avons reproduit à la Figure 4.14 l'excès transitoire de l'erreur satisfaisant la QoS requise

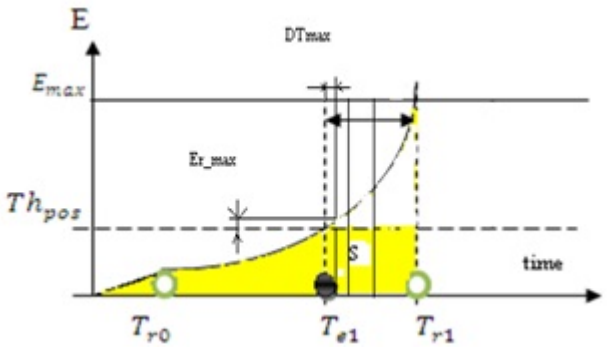


Figure 4.14: Excès transitoire de l'erreur E_r par le ANFIS Reckoning

par l'application après l'application de l'algorithme ANFIS. Nous pouvons conclure que l'approche ANFIS a réussi la réduction de l'erreur spatiale, et par conséquent l'erreur temporelle est également optimisée pour répondre mieux aux exigences de QoS des points de vue utilisateur et application.

4.2.6 Conclusion sur Dead Reckoning

Dans cette section, nous avons proposé une extension de l'algorithme de dead reckoning en utilisant l'algorithme d'apprentissage hybride basé sur la technique neuro-flou. Nous avons montré son intérêt pour optimiser les besoins de QoS des applications DIS dans le cas où l'infrastructure réseau ne supporte pas des services de gestion de QoS. Cette extension utilise les avantages de la technique d'optimisation pour fournir à la fois les entrées floue et l'entraînement par les réseaux neuronaux : la logique floue permet d'encoder le seuil de l'erreur directement en utilisant des règles avec des étiquettes linguistiques, et puis ces étiquettes quantitatives sont

injectées dans le processus d'apprentissage des réseaux de neurones qui automatisé par l'algorithme de rétro-propagation augmenté par la méthode du gradient descendant. Les résultats de la simulation permettent de valider cette approche pour résoudre le problème d'inconsistance et d'incohérence des simulations vis à vis à d'autres approches trouvées dans la littérature.
La navigation à l'estime nécessite la diffusion des états des entités simultanément vers tous les simulateurs de la simulation. Toutefois, la diffusion vers plusieurs entités n'est pas possible dans les réseaux de type Internet vu que les services de diffusion sont désactivés au niveau des équipements réseau. Pour remédier à cela, nous verrons dans la prochaine section, l'interconnexion de simulations distribuées sous DDS pour la navigation à l'estime dans les réseaux étendus sans QoS.

4.3 Interconnexion par Internet Sans QoS avec DDS

4.3.1 Introduction

Nous proposons dans la suite de ce chapitre une architecture utilisant DDS et notre approche de navigation à l'estime pour une interconnexion de simulateurs sur un réseau de type Internet sans QoS. En effet, la combinaison des services offerts par DDS avec notre mécanisme permet d'envisager une telle solution même si sa qualité peut être inférieure à celle utilisant un réseau à QoS.

4.3.2 Problématique de l'interconnexion par Internet avec DDS

Dans ce qui précède, nous avons montré que DDS est très bien adapté pour les communications sur des réseaux locaux qui peuvent offrir aux différents participants la possibilité d'utiliser les services unicast, multicast ou même broadcast vu que la communication locale reste maîtrisée (le préfixe 01-00-5E(/25) est réservé pour les groupes multicast par la RFC1112 [32]). Toutefois, l'interconnexion des applications DDS à travers des réseaux distants nécessite l'activation du service multicast réseau sur les routeurs. Pour des raisons de sécurité et de performance de routage, ces derniers ne permettent pas généralement le multicast réseau, rendant donc le déploiement de DDS sur des réseaux étendus confronté à des problèmes d'intégration et de performance, d'où un premier problème de découverte et de communication entre les différentes entités DDS situées dans des larges envergures.
En plus, pour permettre l'intégration des applications DIS sur des réseaux étendus, il est nécessaire d'ajouter d'autres domaines DDS distants. La possibilité d'utiliser plusieurs domaines permet de séparer les données circulant dans le réseau de celles d'autres applications. Le problème est que le middleware DDS par défaut ne permet pas l'échange des Topics partagés avec d'autres domaines DDS. Il est alors impossible de réaliser une communication entre deux domaines DDS.
En conséquence, les échanges de données ne peuvent être faits sur des réseaux WAN en multicast ou en broadcast que si un producteur de données se comporte comme un serveur de multicast applicatif. Cependant, lorsqu'un flux de données doit être envoyé à N récepteurs, ce flux est répliqué N fois par DDS. La bande passante étant limitée dans un contexte de réseau grande distance, ce problème nécessite de trouver une solution appropriée pour réduire le trafic

échangé. Pour cela, nous allons proposer deux solutions pour l'interconnexion de simulations distribuées DDS que nous avons appelées respectivement "pont-fédéré" et "proxy DDS".

4.3.3 Interconnexion de Simulation Distribuée avec le pont-fédéré

En manipulant l'implémentation "RTI DDS", nous avons constaté que le service " DDS Routing Service " (lié avec RTI) permet de faire communiquer différentes applications DDS distantes. Nous avons alors implémenté un "pont-fédéré" basé sur cette extension pour résoudre ces problèmes.

4.3.3.1 Présentation du DDS Routing Service (DDS-RS)

L'objectif de ce service est de faciliter une communication à grande échelle entre applications DDS à travers des réseaux distants, même à travers des pare-feu et des translations d'adresses réseau (Network Address Translation ou NAT [126]). Rappelons que DDS est initialement conçu pour les applications temps-réel et pour les systèmes embarqués. Sa réussite, en termes de performances de communication à faible latence et à très haut débit, a rendu son utilisation plus large par les extensions qui lui ont été apportées. Le service DDS-RS propose des fonctions de bridge 'DDS-to-DDS' entre applications pour permettre la transformation de données. Ceci permet d'utiliser les applications DDS sans les modifier même si elles ont été développées en utilisant des définitions d'interface incompatibles. C'est souvent le cas lors de l'intégration d'applications existantes dans des systèmes développés indépendamment. Il permet également d'offrir les fonctionnalités suivantes :

- Domain Bridging : Permet de créer un pont entre des domaines DDS. Le middleware seul ne permet pas d'assurer un échange de Topic entre deux applications DDS qui utilisent des espaces de partages globaux différents. Ce service de routage assure alors la communication entre un DataWriter et un DataReader situés dans des domaines DDS différents.
- Topic Bridging : Le service classique de distribution de données se fait par l'échange d'une même instance d'un Topic entre un DataWriter et DataReader, alors que ce service permet aux différentes entités DCPS d'échanger des Topics qui portent des noms et des instances différents. Il permet donc d'assurer une interopérabilité entre des Topics différents.
- Data Transformation : Ce service est capable de transformer des Topics en d'autres. En effet, on peut le configurer de telle sorte qu'il consomme certains Topics et les retransmette en changeant des valeurs d'échantillons de ces Topics. Cette fonction peut être utilisée pour empêcher les données privées d'être publiées dans un réseau étendu.
- Configuration XML : Le comportement du "DDS-RS" peut être configuré à l'aide de fichiers XML pour faciliter sa manipulation.
- Full QoS Configuration : Paramètre le service DDS-RS pour la prise en compte de la qualité de service par la configuration de profils de QoS dans des fichiers XML.

4.3.3.2 Interconnexion entre domaines DDS différents

La première solution pour interconnecter des applications entre des domaines DDS différents consiste à définir de manière statique les différents participants en spécifiant leurs adresses IP respectives dans une variable d'environnement appelée `"NDDS_DISCOVERY_PEERS"`. Par conséquent, le service DDS regarde dans cette variable toutes les adresses des hôtes distants et envoie

les messages "*Discovery*" en unicast à chaque participant. Ainsi, pour permettre à des applications de simulations distribuées sur différents domaines DDS de communiquer, nous introduisons la notion de "pont-fédéré" pour se référer à un pont DDS qui utilise le "Routing Service", et qui se comporte comme un routeur de Topics.
Le choix de ce type de pont vient essentiellement du fait qu'il est réalisé avec les APIs de DDS sans introduire de nouveaux protocoles qui rajoutent de la complexité à la simulation distribuée, sans pour autant perdre l'avantage de l'interopérabilité souhaitée. Ce pont de simulation, montré sur la Figure 4.15, se comporte comme un unique fédéré différents des autres applications DDS. Du point de vue DDS, il s'assure que tout évènement ayant lieu sur un simulateur est propagé vers les autres fédérés. Par exemple, le serveur de simulation sur la Figure 4.15 est configuré de telle sorte qu'il puisse découvrir le pont-fédéré où s'exécute le "service routage DDS" pour assurer la distribution des Topics entre le serveur, situés dans une machine distante, et les différents simulateurs situés dans un autre domaine. Ainsi, le serveur transfère toutes les données présentes dans le domaine DDS 0 vers le domaine DDS 1.

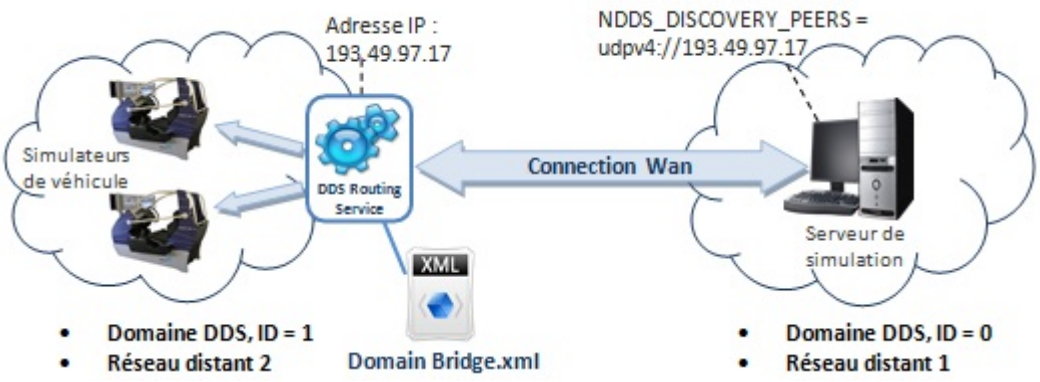


Figure 4.15: Mise oeuvre du "Routing Service" avec PlatSim à QoS

Le pont-fédéré peut découvrir les adresses IP des simulateurs par la configuration d'un fichier XML. La Figure 4.16 nous permet de mieux comprendre la configuration et le principe du service. Ce fichier contient en plus de la description des différents participants, une description des Topics à transférer durant la session, le comportement à suivre pour les router vers différents domaines DDS. Les sessions DDS contiennent (entre autres) les informations suivantes : pour la Session1 nous définissons le domaine d'entrée comme "Input" suivi du participant qui va générer les différents Topics, et le domaine de sortie comme "Output", vers lequel les Topics vont être redirigés. Pour la Session2 [1], nous définissons les opérations dans le sens inverse, c'est à dire les éléments nécessaires pour transférer les Topics présents dans le domaine DDS1 vers le domaine DDS0. Cette opération nous permet donc d'assurer une communication bidirectionnelle entre les deux domaines. De cette manière, le pont-fédéré basé sur le "Routing Service" permet alors d'assurer la découverte des entités et l'échange des Topics entre deux domaines DDS différents.

1. N'est montrée dans la Figure 4.16 pour des raisons de l'espace

```
<routing_service name="defaultBothWays">
   <domain_route name="TwoWayDomainRoute">
        <participant_1>
           <domain_id>0</domain_id>
        </participant_1>
        <participant_2>
           <domain_id>1</domain_id>
        </participant_2>
         <session name="Session1">
            <auto_Topic_route name="AllForward">
               <publish_with_original_info>true</publish_with_original_info>
               <input participant="1">
                  <allow_Topic_name_filter>*</allow_Topic_name_filter>
    <allow_registered_type_name_filter>*</allow_registered_type_name_filter>
                  <creation_mode>ON_DOMAIN_AND_ROUTE_MATCH</creation_mode>
               </input>
               <output>
                  <allow_Topic_name_filter>*</allow_Topic_name_filter>
    <allow_registered_type_name_filter>*</allow_registered_type_name_filter>
                  <creation_mode>ON_DOMAIN_AND_ROUTE_MATCH</creation_mode>
               </output>
            </auto_Topic_route>
         </session>
    </domain_route>
 </routing_service>
```

Figure 4.16: Configuration du service de routage pour connecter de domaines DDS différents

4.3.3.3 Configuration du pont-fédéré

Afin de valider la proposition d'utiliser le service de routage DDS pour les échanges de messages, nous avons considéré une application qui échange des topics DDS avec le middleware NDDS [112] entre différentes machines de la plateforme LaasNetExp. Cette application contiennent un producteur qui émet un grand nombre de messages (en DDS cela correspond à un producteur du Topic "Message" qui contient juste une variable clé : UserID de type long et une autre variable pour le contenu du message Content de type séquence de caractère) et de l'autre côté un récepteur (en DDS cela correspond à un subscriber du même Topic " Message ").

Dans cette configuration, le pont-fédéré est lancé et exécuté dans *Euqos7* pour se comporter à la fois comme consommateur et pont avec une configuration correspondant à "Domain Bridge" avec deux domaines DDS différents pour chaque réseau IP. L'application productrice est lancée sur la machine *EuQoS6*. Nous avons remarqué que tous les messages (Topics) échangés dans le domaine DDS 0 (émis par Euqos6) sont transmis vers toutes les machines consommatrices (Euqos7 et 8) situées dans le domaine DDS 1, mais le plus intéressant dans ce test est que la machine Euqos6 envoie deux instances du même flux vers le pont-fédéré ce qui correspond aux deux machines consommatrices, comme le montre la Figure 4.17. Même si le pont-fédéré permet la retransmission des Topics du domaine DDS0 vers le domaine DDS1, toutes les données transmises vers Euqos8 (ou EuQoS7) mènent à la création d'un nouveau socket sur le port 7413 (resp. 7411) depuis la machine source (producteur) Euqos6 vers le pont-fédéré. De ce fait, les données sont émises de manière dupliquée vers le pont-fédéré, alors que cela n'est pas indispensable.

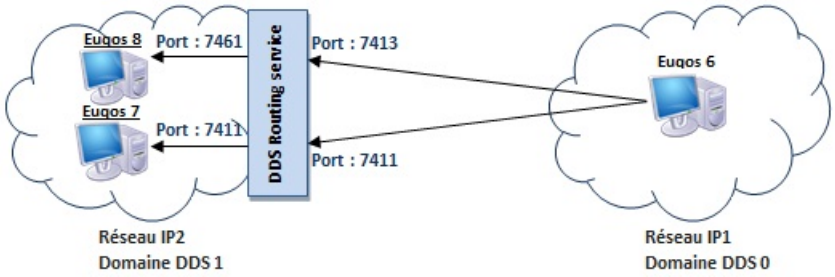


Figure 4.17: Test du pont-fédéré avec le Domain Bridge

4.3.3.4 Test du pont-fédéré avec le Topic Bridge

Nous avons vu dans ce qui précède que l'utilisation du pont-fédéré en configurant le service "Domain Bridge" pose un problème de duplication des flux envoyés par la machine productrice "EuQoS6" vers les autres machines consommatrices situées sur le réseau du domaine IP distant. Cette solution permet d'acheminer le trafic sur deux réseaux IP hétérogènes. Toutefois, cette solution est partielle, puisque le flux envoyé par la machine productrice est dupliqué par le nombre de machines consommatrices découvertes par le service "Discovery" de DDS.

```
<domain_route name="DomainRoute" enabled="true">
    <participant_1>
        <domain_id>0</domain_id>
    </participant_1>
    <participant_2>
        <domain_id>1</domain_id>
    </participant_2>
    <!-- A session with default configuration
    that contains a single topic route -->
    <session name="Session" enabled="true">
        <auto_topic_route name="SquaresToCircles">
            <!-- Reading data from participant_1 -->
            <input participant="1">
                <!-- Reading a type whose registered name is ShapeType.
                As we don't register ourselves on participant_1, its actual
                type code will have to be discovered when the router runs -->
                <registered_type_name>ShapeType</registered_type_name>
                <!-- Reading topic Square -->
                <topic_name>Square</topic_name>
            </input>
            <output>
                <!-- Writing the same type -->
                <registered_type_name>ShapeType</registered_type_name>
                <!-- With the same topic -->
                <topic_name>Square</topic_name>
            </output>
        </auto_topic_route>
    </session>
</domain_route>
```

Figure 4.18: Configuration du service de routage avec le Topic Bridge

Pour résoudre ce problème, le pont-fédéré est maintenant configuré pour jouer le rôle de pont de Topics (Topic Bridge). Cette configuration a pour objectif de transférer les Topics produit

dans un domaine DDS 0 vers un domaine DDS 1 en transformant ces "Topics" en d'autres Topics. Mais, puisque ces Topics ne doivent pas être transformés dans le cas de l'application Platsim_QoS, alors ils doivent être les mêmes à l'entrée (réception depuis les producteurs) et la sortie (envoi aux consommateurs). Pour combler cette difficulté, la Figure 4.18 montre comment le pont-fédéré ainsi conçu est capable de régénérer les Topics originaux, même pour un grand nombre de Topic échangés entre les différentes machines.

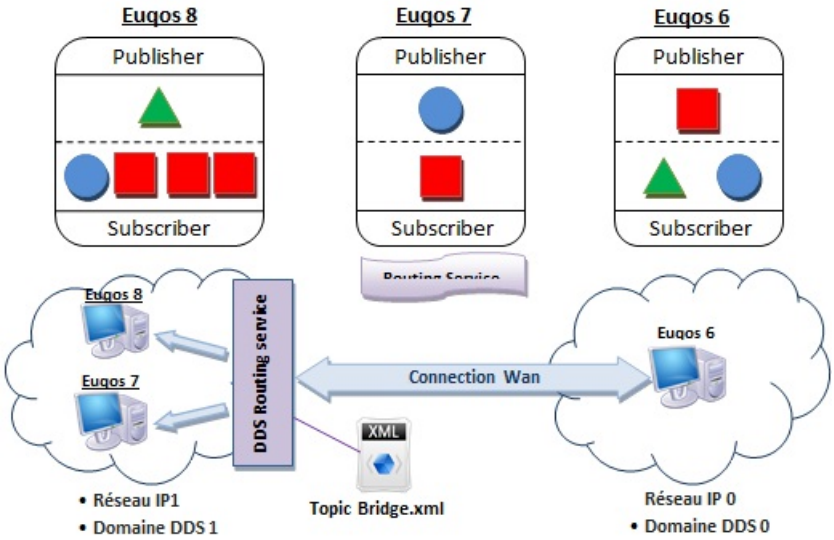


Figure 4.19: Scénario de communication du pont-fédéré le Topic Bridge

Afin de vérifier si le volume du trafic transmis du producteur dépend du nombre de machines réceptrices et du nombre de consommateurs dans chacune, nous avons effectué le test suivant. Ce test consiste à publier des Topics (les carreaux sur la Figure 4.19) du côté de l'un des producteurs et déclarer plusieurs consommateurs (Subscribers) des mêmes Topics répartis sur les autres machines et situées dans des réseaux IP différents et des domaines DDS différents.
Les résultats de ce test sont présentés dans la Figure 4.20, et la Figure 4.21. La Figure 4.20 représente les flux de données envoyés et reçus par la machine Euqos6 :

- Le trafic (1) représente le flux de données envoyé d'Euqos 6 vers le pont-fédéré avec un débit de 20Kbps et correspond à la distribution d'un seul Topic (Carreau).
- Le trafic (2) représente le flux de données reçus du routeur DDS avec un débit de 40Kbps et correspond à la consommation de deux Topics (Cercle et Triangle).

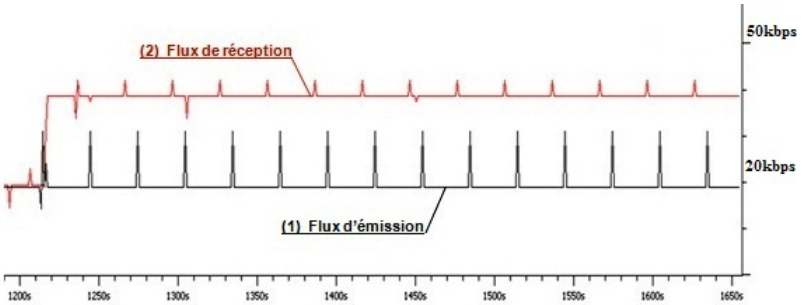


Figure 4.20: Les flux de données captés sur Euqos 6

La Figure 4.21 représente les flux de données envoyés et reçus par le pont-fédéré exécuté sur Euqos7 :

- Le trafic (1) représente le flux de données reçu d'Euqos6 qui est similaire au trafic (1) de la Figure précédente.
- Le trafic (2) représente le flux de données envoyé vers Euqos6 qui est similaire aussi au trafic (2) de la Figure précédente.
- Le trafic (3) représente le flux de données envoyé d'Euqos7 vers Euqos8 avec un débit de 80Kbps et correspond à la distribution de deux Topics et 4 instances (3 carreaux et un cercle).

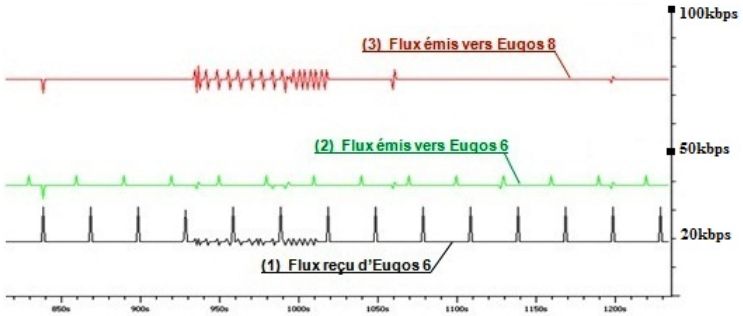


Figure 4.21: Flux de données capté par le pont-fédéré avec le Topic Bridge

D'après ces analyses, nous pouvons retenir que le volume de données envoyé du domaine DDS 0 (depuis Euqos6) vers le domaine DDS 1 (Euqos7 qui contient en plus du consommateur, le pont-fédéré) ne dépend ni du nombre de simulateurs ni du nombre de consommateurs dans le site distant, mais juste des types de Topics envoyés. Nous remarquons aussi que le transfert des Topics d'un site à un autre commence dès que le pont-fédéré s'inscrit à ces Topic et qu'il les duplique par la suite dans son réseau à destination des participants consommateurs.
Nous pouvons donc constater qu'il est possible d'optimiser le volume de données échangées sur les liens physiques entre des applications DDS situées dans deux réseaux IP distants et deux domaines DDS différents. Cependant, le problème est que cette solution n'est pas capable d'offrir les mêmes performances dans le cas d'un seul domaine DDS. En effet, nous avons remarqué que le volume de données envoyé d'Euqos6 vers Euqos7 est deux fois plus grand quand ils sont dans un seul domaine DDS.

4.3.4 Interconnexion de Simulation Distribuée par Proxy DDS

Malgré les avantages que peut offrir le pont-fédéré dans le cadre de PlatSim_QoS, il reste toujours incapable de résoudre tous les problèmes cités dans le paragraphe précédent. Nous proposons une nouvelle alternative : le "Proxy DDS" qui doit être capable de réaliser les objectifs suivants :

- interconnecter des applications DDS à travers un réseau étendu de type Internet,
- géré les communications dans un seul ou plusieurs domaines DDS comme dans un seul espace global de partage de données,
- optimiser le volume de trafic échangé entre les entités distantes,
- offrir des profils de QoS DDS différents en réseau local et en réseau distant,

– assurer l'interopérabilité entre les QoS-Policies DDS et les politiques de QoS réseau (on va le voir dans le chapitre suivant).

Dans la suite de cette section, nous détaillons les différentes fonctions du Proxy DDS ainsi que les services qu'il offre.

4.3.4.1 Relier différentes applications DDS à travers un réseau WAN

A l'inverse du pont-fédéré qui résulte d'une configuration du RTI Routing Service, le proxy DDS est une solution logicielle que nous avons implémentée en C++.

Le proxy DDS permet de relier des simulations distribuées DDS situées sur des réseaux grande distance. Pour ce faire, le proxy DDS gère la communication de tous les participants de son domaine IP avec les autres proxies DDS du réseau. Cette configuration nécessite alors de déployer autant de proxies DDS que de domaines IP sur une machine à côté du routeur de bordure. L'architecture de déploiement permettant d'assurer la communication entre les différents proxies DDS distants est montrée par la Figure 4.22. Elle consiste à configurer le service de découverte ("*NDDS_DISCOVERY_PEERS*") avec toutes les adresses IP des différents proxies.

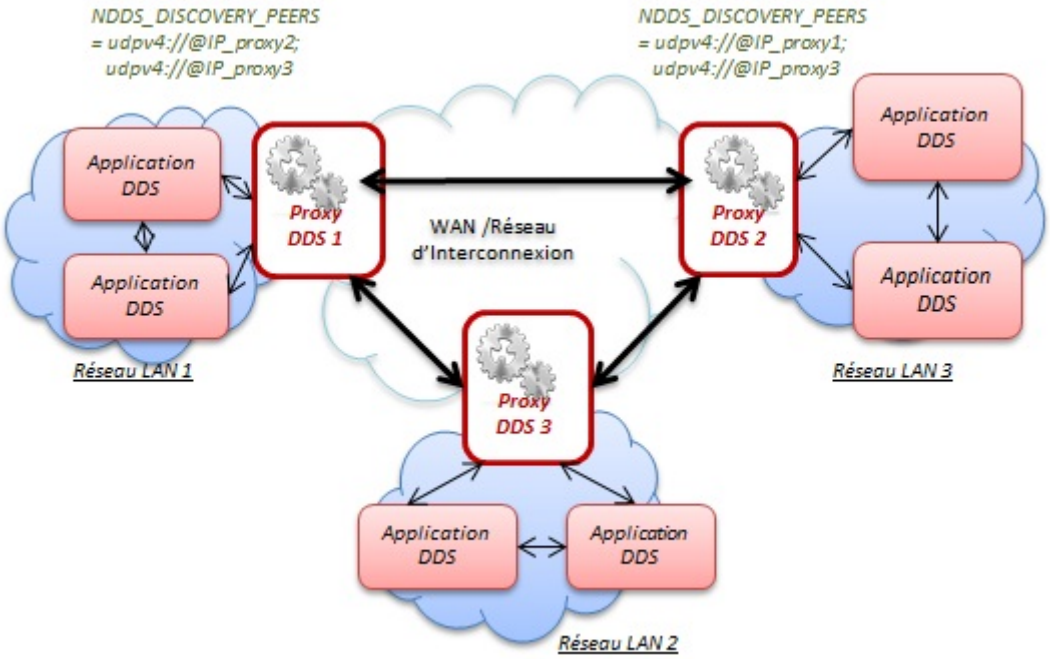


Figure 4.22: Déploiement de Proxies DDS dans un réseau grande distance multi-domaines IP

4.3.4.2 Utilisation d'un seul domaine DDS

Le proxy DDS est composé de différentes entités qui communiquent dans un seul domaine DDS, mais pouvant être dans des domaines IP différents. Il consomme alors tous les Topics auxquels il est intéressé dans son réseau et retransmet les Topics destinés aux consommateurs distants vers les autres proxies DDS distants respectifs. Ensuite, les autres proxies DDS vont redistribuer tous ces Topics reçus dans leur domaine, et à travers ce mécanisme toutes les applications DDS situées dans des réseaux différents pourront partager le même domaine DDS

et considérerons le réseau WAN comme étant un espace de partage global. La Figure 4.23 montre le déploiement de cette architecture.

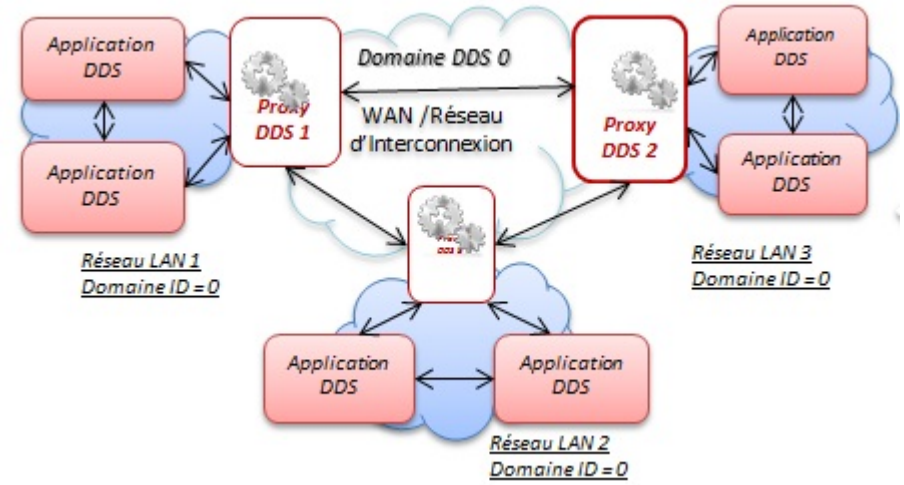


Figure 4.23: Déploiement du proxy DDS sur un seul domaine DDS

4.3.4.3 Implémentation du proxy DDS

Le Proxy DDS est une application qui se lance sur une machine et s'exécute explicitement comme un service DDS. Il adopte comme technologie de base le middleware DDS en utilisant les API et les services nécessaires du RTI DDS. Ce proxy est composé de toutes les entités de base de DDS, et son développement et son déploiement dépendent du besoin de l'application mise en oeuvre. Cela nécessite une spécification de tous les Topics échangés entre les différentes entités de l'application. Pour mieux comprendre le fonctionnement du proxy DDS, nous présentons les différentes étapes du déroulement sur un exemple.

Mise en place d'un Proxy DDS Nous considérons une application qui souhaite émettre un seul Topic "**T1**" d'un réseau IP à un autre à travers une connexion gérée comme un seul et unique domaine DDS, et deux proxies installés à l'extrémité de chaque réseau. Comme le montre la Figure 4.24, cette application dispose d'un producteur géré par un DataWriter de "**T1**" et deux consommateurs gérés par deux DataReaders du même Topic.

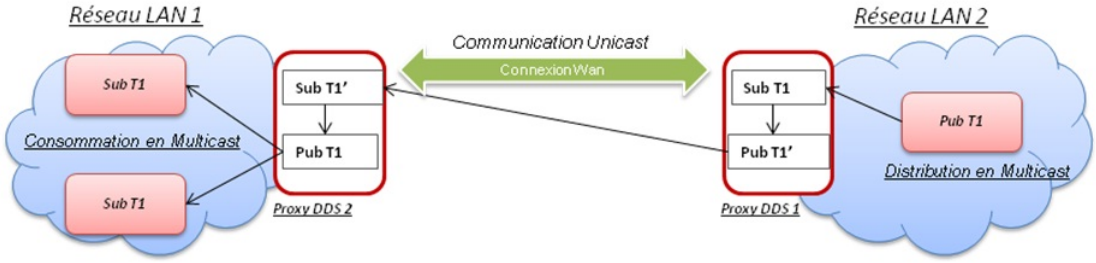


Figure 4.24: Interconnexion des deux proxies DDS

Les étapes illustrées sur cette figure montrent le scénario d'échange de données entre les différents participants, qui se déroule comme suit :

- Étape 1 : Le producteur de l'application publie dans le réseau local 2 des échantillons du Topic **T1** en multicast.
- Étape 2 : Le proxy DDS1 s'inscrit au Topic **T1** et consomme tous les échantillons présents dans le domaine DDS1 (le Subscriber de **T1** consomme uniquement les échantillons publiés en multicast).
- Étape 3 : A l'aide d'une file d'attente, le proxy DDS1 redistribue le Topic **T** consommé dans le domaine DDS1 en le renommant en un autre Topic **T1'** afin d'éviter que **T1** soit consommé une autre fois après avoir été redistribué dans le domaine DDS1, et par la suite éviter une boucle locale de consommation et production des échantillons du même Topic au niveau du Proxy DDS1.
- Étape 4 : De l'autre côté du réseau, un autre proxy DDS2 commence à consommer le Topic **T1'** dès sa présence dans le domaine DDS2 (les échantillons entre les deux proxies sont échangés en Unicast puisqu'ils sont transmis à travers le réseau WAN), ensuite les proxies transforment **T1'** en **T** et le redistribue en multicast dans le réseau local 1.
- Étape 5 : Finalement, les consommateurs de l'application reçoivent en multicast tous les échantillons publiés par le proxy de leur réseau ce qui correspond au Topic **T1** envoyé par le producteur de l'application dans le réseau local 2.

4.3.4.4 Evaluation du proxy DDS

Pour évaluer le proxy DDS, d'abord nous avons réalisé une application DDS distribuée qui assure un échange de données dans un seul domaine DDS. Deux Topics ont été utilisés : "UserInfo" et "CarInfo". Côté producteur de l'application on considère un Publisher du Topic «UserInfo» et un Subscriber du Topic «CarInfo», et côté client de l'application on considère un Subscriber de «UserInfo» et un Publisher de «CarInfo».
Nous avons déployé le Proxy DDS pour mettre en réseau l'application de test. Ce Proxy est responsable de l'échange des deux Topics entre Euqos6 et les machines clientes de l'application situées dans deux réseaux locaux différents et gérées par un seul domaine DDS.
Le scénario utilisé pour ce test est présenté dans la Figure 4.25 : Considérons l'architecture de la plateforme LaasNetExp pour ce test. Le serveur de l'application est lancé sur la machine (Euqos 6) du domaine 3 (Vlan203) et les clients sur les machines (Euqos 7 et 8) du domaine 1 (Vlan 201). Par la suite, le Proxy DDS est installé sur les deux réseaux (précisément sur Euqos 6 et 7). Le scénario de test consiste à lancer sur :

- Euqos6 : Un Publisher de "UserInfo", un Subscriber de "CarInfo" et le Proxy DDS1,
- Euqos7 : Deux Subscribers de "UserInfo" et le Proxy DDS 2,
- Euqos8 : Deux Subscribers de "UserInfo" et un Publisher de "CarInfo"

4.3.4.5 Résulats et Analyses

Nous donnons ici les résultats (relevés avec Wireshark) du trafic échangé entre les deux réseaux. La Figure 4.26 illustre les différents flux échangés : entre Euqos6 et Euqos7 qui correspond au flux transmis entre les deux Proxy DDS 1 et 2, le flux retransmis du Proxy DDS 1 vers Euqos8 et finalement le flux transmis d'Euqos8 vers le Proxy DDS 1. Ces flux de trafic sont présentés sous forme de courbes dans les figures suivantes.

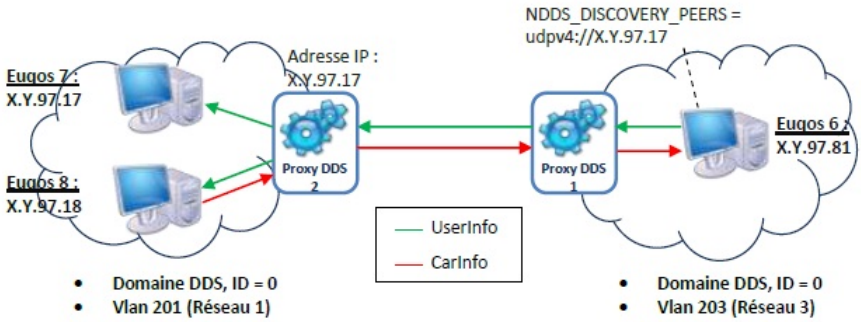


Figure 4.25: Scénario de test du Proxy DDS

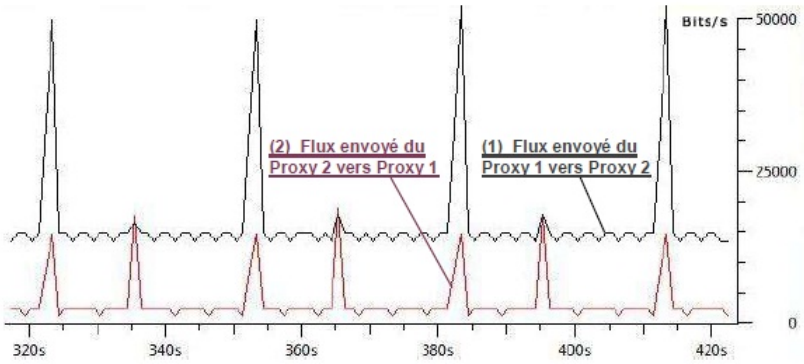


Figure 4.26: Courbes du trafic échangé entre les deux proxies DDS

La Figure 4.26 illustre le trafic transmis entre les deux Proxy DDS distants. La courbe (1) : représente la distribution d'une seule instance du Topic "UserInfo" envoyé du proxy 1 vers le proxy 2 avec un débit moyen de 15Kbps. La courbe (2) : représente la distribution d'une seule instance du Topic "CarInfo" envoyé du proxy 2 vers le proxy 1 avec un débit moyen de 2,5Kbps.

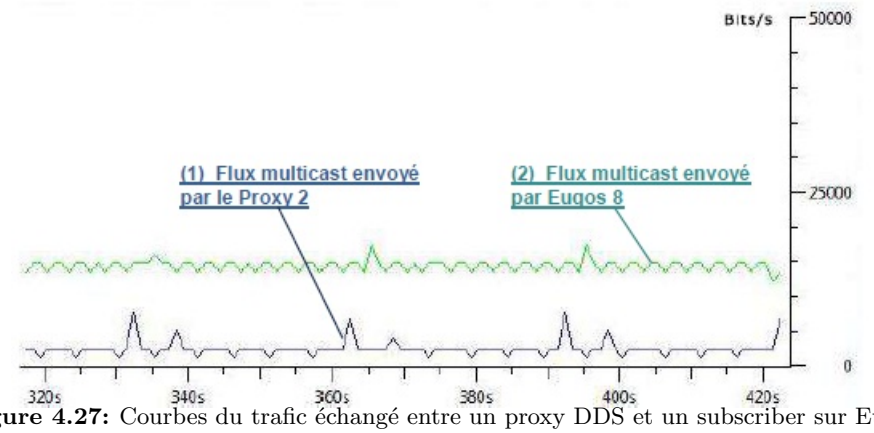


Figure 4.27: Courbes du trafic échangé entre un proxy DDS et un subscriber sur EuQoS8

La Figure 4.27 illustre le trafic transmis en multicast dans le domaine DDS1 entre le proxy DDS 2 et les participants DDS dans son domaine (EuQoS 7 et 8). La courbe (1) représente la redistribution du Topic "UserInfo" en multicast, envoyé par le proxy DDS 2 sur le réseau

avec un débit moyen de 15Kbps. La courbe (2) représente la distribution du Topic "CarInfo" en multicast, envoyé par Euqos 8 sur le réseau avec un débit moyen de 2,5Kbps.
Les résultats obtenus ont confirmé que le volume de données envoyé par un proxy DDS (flux des Topics DDS de l'application productrice) dans un seul domaine DDS, à travers le réseau d'interconnexion ne dépend ni du nombre de simulateurs ni du nombre de subscriber sur les ces simulateurs, mais dépend uniquement du profil de production des Topics envoyés (Type et taille).
Nous pouvons conclure que le proxy DDS représente une solution efficace pour l'optimisation de la bande passante des applications DDS distribués et du coût du transfert de données partagées dans un seul Domaine DDS à travers un réseau multi-domaine à QoS. Ainsi, comme prédit, le trafic sur le réseau longue distance est réduit, à sa bande passante minimale.

4.4 Conclusion

Dans ce chapitre, nous avons pu présenter deux contributions pour la gestion de la communication des applications de simulation distribuée en réseaux étendus : la première concerne la communication dans un réseau distant qui ne supporte pas la QoS, et dans laquelle nous avons étudié la mise en place d'un modèle de dead reckoning permettant, en cas de fonctionnement anormal ou de défaillance du réseau, d'offrir une simulation optimisée consistante, et la plus proche possible du cas normal.
Dans la deuxième contribution, nous avons montré comment mettre en réseau une simulation DDS tout en assurent son interfonctionnement en grande distance. Pour cela, nous avons d'abord expliqué brièvement le service de routage que fournit DDS en présentant ses caractéristiques essentielles ainsi que les éléments qui le composent. Ensuite, à la base des contraintes que nous avons trouvées lors de la mise en place du "Routing Service", nous avons pu présenter le Proxy DDS dans ce mémoire. Nous avons donc commencé par aborder les fonctionnalités du proxy DDS et les avantages que nous a fourni son utilisation entre domaines DDS différents et les contributions pour l'optimisation des flux échangés entre les différents simulateurs.
Bien que ces solutions améliorent notre réponse à la problématique de cette section, elles n'offrent qu'une réponse imparfaite au problème de gestion de la QoS. Nous verrons dans la prochaine section, notre proposition pour la gestion de la QoS réseau pour les communications à grandes échelles.

Chapitre 5

Interconnexion sur des réseaux grande distance à QoS

5.1 Introduction

Garantir la qualité de service des flux de données ne se limite pas à implémenter une politique de différentiation de trafic et définir un traitement spécifique pour chaque classe de service au niveau des équipements d'interconnexion. En effet, garantir la QoS des flux de données de la plateforme passe par la définition de QoS-Policies pour le middleware DDS, mais aussi de politiques qui assurent la continuité de la QoS de bout-en-bout pour les flux DDS, d'abord par l'application qui fournit ces besoins à la couche middleware, et ensuite en traversant le réseau d'interconnexion multi-domaines. Pour cela, notre implémentation de PlatSim_QoS en réseaux étendus utilise la différentiation des classes de trafic et signale la QoS requise par l'application au réseau en utilisant des requêtes envoyées sur un plan de service. Ce plan de service les achemine vers le plan de contrôle pour demander une réservation des ressources satisfaisant les besoins de l'application, et en se basant sur deux concepts fondamentaux : a) la définition des flux DDS et le marquage des paquets en tant que composants de ces flux ; b) l'implémentation de la QoS sur chaque routeur de bordure via des règles et des politiques de QoS qui vont définir le comportement de chaque classe de service.
Ce chapitre va présenter ces deux contributions concernant l'étude et la mise en place d'une architecture de réseau à grande distance à QoS garantie. Dans la Section 5.2, nous présentons une architecture de signalisation de la QoS pour des applications de simulation en se basant sur l'utilisation conjointe du protocole COPS et de la signalisation SIP. Ces travaux rentrent dans le cadre d'une coopération que nous avons réalisée avec *l'Université Vanderbilt* aux USA durant un stage de mobilité de doctorants.
Dans la Section 5.3, nous introduisons notre deuxième contribution qui extend des travaux réalisés au LAAS dans le projet européen EuQoS. Le choix de EuQoS est motivé par le fait qu'il a proposé une architecture ouverte technologiquement virtualisée de type NGN et qui préfigure ce que pourrait être l'Internet à QoS. Nous avons alors utilisé des composants de cette architecture que nous avons adaptés pour fournir, à l'utilisateur final ou à l'administrateur de l'application, des interfaces simples lui permettant de demander le type de service requis pour son application sans avoir besoin de changer le protocole de signalisation.

5.2 Couplage de SIP et DDS pour la signalisation

5.2.1 Introduction

Nous avons précédemment proposés une solution pour la mise en réseau locaux de simulations distribuées sur le middleware DDS en utilisant la décomposition des flux applicatifs en classes de Topics DDS, chacune de ces classes ayant ses paramètres de QoS. En effet, jusque-là, la configuration de la QoS réseau se faisait aisément en configurant les services disponibles sur le commutateur LAN afin de répondre aux besoins de l'application de manière statique.
Dans cette section, nous présenterons notre approche pour la signalisation de la QoS DDS en utilisant les extensions de SIP pour le protocole COPS (voir section 2.5.6). Notre proposition consiste à faire le lien, entre les besoins de l'application de simulation distribuée et le dimensionnement des ressources, par l'intermédiaire du protocole SIP. Nous commencerons par décrire les correspondances entre les paramètres DDS et les liens à faire avec SIP ; ensuite, nous présenterons l'architecture du Framework et les modifications que nous avons apportées pour faire le mapping entre SIP et DDS et ainsi permettre la signalisation de la QoS DDS dans le réseau. Cette approche permet d'assurer l'interopérabilité des sessions DDS sur des réseaux implémentant d'une part des mécanismes de gestion de QoS de type DiffServ, et d'autre part implémentant COPS-DRA comme protocole de réservation de ressources. Enfin, nous présenterons les scénarios de mise en place de la signalisation, de la mise en place de la QoS, et de la libération des ressources.

5.2.2 Proposition de l'architecture

Avant toute chose, nous devons préciser que l'utilisation de SIP est motivée par sa large diffusion sur l'Internet et par le consensus qui existe autour de son utilisation pour la réservation de ressources dans un réseau à QoS. SIP est utilisé pour la gestion des sessions multimédia afin de permettre l'adaptation et le choix de l'envoi des flux multimédia avec différents Codecs. Cela nécessite d'abord une phase de négociation pour choisir le codec supporté par les deux clients SIP distants. Ensuite, la mise en place de la QoS réseau nécessite un proxy SIP conscient de la QoS, qui puisse communiquer avec un serveur de QoS, serveur qui appliquera alors les politiques de QoS réseau au nom de l'application (et en connaissant les profils de trafic de celle-ci. i.e., les codecs).
DDS possède néanmoins une différence de paradigme de communication avec SIP car le premier est orienté diffusion alors que le second est orienté point à point. De plus, SIP utilise des messages SDP prédéfini à échanger entre les clients SIP, ces messages contenant des attributs SIP qui ne supportent pas les champs des QoS policies de DDS.
De même, la notion de Codec de SIP est absente au niveau DDS pour lui permettre de choisir et d'adapter le rythme de production/consommation. Une autre particularité de DDS est que ses paramètres de QoS sont régit par le contrat Request/Offer (R/O)qui doit absolument être respecté pour que la communication puisse s'établir entre le Publisher et le Subscriber. Le choix doit donc aussi prendre en compte les QoS policies respectant ce contrat pour contrôler la communication de bout-en-bout, car seules les QoS policies qui contrôlent simultanément les entités DataReader, DataWriter et les Topic peuvent être utilisées. Néanmoins, comme la majorité de ces QoS policies ne sont pas modifiables pendant la communication, elles doivent donc être fixées à l'avance, et ceci empêche l'adaptation des applications DDS à la variabilité

de la charge du réseau. Pour cela, le choix des QoS policies doit porter sur celles qui sont à la fois modifiables et permettent de coupler les entités DDS distantes.
Ces particularités ne sont pas rédhibitoires et devront être prises en compte dans notre architecture. De plus, le lien entre le besoin applicatif et les ressources réseau doit reposer sur des améliorations du protocole SIP pour permettre le traitement des attributs DDS par le descripteur de session SDP, ceci afin de les connecter aux besoins de l'application. Pour cela, nous avons modifié SDP pour ajouter des descripteurs spécifiques à DDS. Nous avons apporté des modifications au proxy SIP classique pour qu'il devienne conscient de la QoS DDS.

5.2.2.1 Lien entre le service DDS et le réseau

Le service réseau de DDS (Networking Service) contient des paramètres de configuration de l'application décrits dans un fichier XML. Ce fichier XML spécifie le mode de communication que va utiliser le middleware pour l'envoi et la réception des données entre les participants. La Figure 5.1 montre un extrait ce fichier. Le service réseau DDS utilise le service de diffusion (multicast ou Broadcast) sur une seule "partition" réseau nommée "GlobalPartition". Dans ce cas, tous les messages sont envoyés en Broadcast. Pour l'extension Platsim_QoS proposée, l'utilisation du protocole SIP ne permet pas d'utiliser un service de diffusion de type Broadcast ou multicast. Ceci nous a demandé de préciser l'adresse IP source et l'adresse IP destination pour communiquer en unicast dans les attributs "NetworkInterfaceAddress" et "GlobalPartition" respectivement.
Il est important de noter que toutes les machines exécutant DDS doivent avoir strictement le même fichier de configuration pour le service networking sinon il peut y avoir des incohérences dans les communications. A titre d'exemple, si les numéros de ports indiqués dans le fichier de configuration sont différents, une unité enverra les données sur un port X et l'entité destinataire s'attendra à recevoir les données sur un port Y, et les données ne seront jamais remises au destinataire.
Platsim_QoS tirera profit de ce fichier de configuration. Pour cela, nous avons implémenté un "parseur" pour récupérer le contenu des balises XML et traduire ce contenu en des informations utilisables pour effectuer une réservation de ressources. Nous avons développé un module qui convertit les politiques de QoS DDS sous forme d'attributs spécifiques dans le descripteur de session SDP. Ce module est intégré au proxy SIP que nous avons enrichi pour qu'il devienne conscient de la QoS DDS. Ces paramètres de QoS DDS sont rattachés à l'envoi et à la réception des Topics DDS dans un même canal.
Dans le service networking de DDS, l'élément Channel permet de créer et de configurer des "Channels Users". L'attribut "default" permet de spécifier que le Channel est sélectionné comme le Channel par défaut si aucun autre Channel n'offre la qualité de service requise par un message. L'attribut "enabled" permet d'activer ou pas un Channel. L'attribut "name" permet d'identifier un Channel et ce nom doit être unique. L'attribut "Reliable" permet d'indiquer si le Channel envoie les messages de manière fiable ou pas. L'attribut "PortNr" permet de spécifier le numéro de port destination des données transportées via ce Channel.
L'attribut "DiffServField" permet de spécifier une valeur qui sera utilisée pour marquer le champ DSCP des paquets IP encapsulant les données qui utiliseront ce Channel. L'attribut "MaxBurstSize" permet de spécifier la taille maximale des Topics (en octet) à envoyer dans le Channel par unité de temps, définie par l'attribut "Resolution" (en millisecondes). Ces deux

```
<NetworkInterfaceAddress>193.49.97.81</NetworkInterfaceAddress>
</General>
  <Partitioning>
    <GlobalPartition Address="unicast"/>
                <GlobalPartition Address="193.49.97.17"/>
  </Partitioning>
  <Channels>
    <Channel default="true" enabled="true"
                name="BestEffort" reliable="false">
      <PortNr>54100</PortNr>
                        <Resolution>10</Resolution>
                        <FragmentSize>554</FragmentSize>
                        <AdminQueueSize>4000</AdminQueueSize>
                        <Sending>
                                <DiffServField>18</DiffServField>
                                <MaxBurstSize>500</MaxBurstSize>
                                <QueueSize>400</QueueSize>
                                <ThrottleLimit>1024</ThrottleLimit>
                                <MaxRetries>10</MaxRetries>
                                <RecoveryFactor>3</RecoveryFactor>
                        </Sending>
                        <Receiving>
                        <ReceiveBufferSize>1000000</ReceiveBufferSize>
                         <Scheduling>
                                        <Priority>60</Priority>
                                       <Class>Realtime</Class>
                        </Scheduling>
                        </Receiving>
                </Channel>
    <Channel enabled="true" name="Reliable"
                reliable="true">
      <PortNr>54110</PortNr>
    </Channel>
  </Channels>
  <Discovery enabled="true">
    <PortNr>54120</PortNr>
                <ProbeList>193.49.97.81;193.49.97.17;193.49.97.18;193.49.97.16</ProbeList>
  </Discovery>
</NetworkService>
```

Figure 5.1: configuration des différents canaux DDS

derniers paramètres permettent de spécifier le débit nécessaire pour l'application DDS utilisant le Channel. Le débit total alloué au trafic DDS transmis sur le Channel est donné par : $dbit = (MaxBurstSize \times 8 \times (\frac{1000}{Resolution}))$, en bits/s. L'élément Discovery permet de créer un "Discovery Channel". L'élément "PortNr" permet de spécifier un numéro de port sur lequel chaque nœud DDS va recevoir les messages de découverte. L'élément "ProbeList" contient les adresses des machines qui vont être contactées pour récupérer une liste initiale des participants dans l'espace de partage global de DDS.

5.2.2.2 Les paramètres de QoS DDS utilisés pour la signalisation SIP

L'implémentation des attributs de QoS DDS au sein du message SIP utilise les paramètres de QoS "deadline", "latency_budget", "Transport_priority" et "Reliability" indiqués dans le Tableau 5.1. Nous avons utilisé ces paramètres essentiellement pour deux raisons : 1) ils caractérisent mieux le lien entre la QoS DDS et les besoins de l'application en termes de QoS réseau ; 2) ce sont les seuls paramètres dont leurs valeurs sont modifiables au cours de la simulation

sans avoir besoin de recompiler le code source de l'application [1].

Table 5.1: QoS DDS Policies introduites dans le message SDP

QoS DDS Policy	Description
Deadline	Le DataReader attend le rafraîchissement des instances des Topics DDS ou moins une fois pendant une période égale au deadline. Le DataWriter s'engage alors à lui fournir ces nouvelles instances dans un intervalle ne dépassant pas ce deadline. Le deadline est décrit par une structure qui a deux éléments de type "long" : la durée en secondes et en nanosecondes.
Latency_budget	La durée depuis l'envoi des données depuis le DataWriter jusqu'à son arrivée dans le cache du DataReader distant et la notification de celui par cet évènement. Le latency_budget est aussi défini par une structure qui contient la durée en secondes et en nanosecondes.
Reliability	Elle spécifie la fiabilité du service pour indiquer si le mode de transport est UDP ou bien TCP.
Transport_Priority	Décrit une valeur qui sera utilisée pour marquer le champ DSCP des paquets IP encapsulant les données qui utiliseront ce Channel.

En effet, nous utilisons le marquage DSCP pour spécifier la classe de trafic à utiliser. Ce champs est accessible directement via la QoS Policy "Transport_Priority" associée à un "Channel user". En plus, l'interface de service réseau utilisée par le middleware lui permet d'exprimer une demande d'établissement de service réseau avec des exigences sur le débit et le délai. Pour cela, la prise en compte du débit se fait assez aisément par l'extraction des paramètres "MaxBurstSize" et "Resolution" pour déterminer le profil du trafic de Platsim_QoS. L'analyse du profil de production du flux de la simulation permet par exemple de spécifier un débit moyen de 400kbps $(8 \times 500 \times (\frac{1000}{10}))$.
Pour ce qui concerne les exigences de délai, l'approche suivie consiste à décrire la latence maximale admissible dans le paramètre de QoS DDS "Latency_budget" afin qu'elle puisse être utilisée par la signalisation.

5.2.2.3 Extension apportées à SIP/SDP pour supporter les QoS policies de DDS

SIP nous a servi pour transmettre les paramètres de la QoS DDS nécessaires pour la réservation des ressources requises par l'application DDS dans le message "INVITE". Ces paramètres sont incorporés dans les champs de description de session SDP [57] : le champ 'b' permet de décrire le débit nécessaire par la session DDS et à demander au réseau ; le champ "a" (attribute), décrit sous forme $< attribute >=< value >$, est une extension de SDP décrite dans la RFC3312 [23] pour la définition de nouveaux paramètres non définis par les standards SIP. Toutefois, les QoS DDS ne sont pas supportées dans cette RFC, et pour cela nous l'avons étendue pour permettre l'utilisation des attributs de la QoS DDS.
Cette extension particulière permet la mise en place des mécanismes nécessaire à la signalisation de la QoS réseau. Le proxy SIP (que nous allons décrire dans la prochaine section), conscient de la QoS DDS, se charge de négocier au nom des clients DDS les mécanismes de mise en place de la QoS avec les équipements réseau. Le nouveau message SDP envoyé par une application DDS via le client SIP est de la forme décrite dans le Tableau 5.2.
Pour décrire la nouvelle extension apportée à SIP/SDP, nous décrivons la sémantique complète

1. Tableau de QoS DDS, pages 97-108, accessible sur http ://www.omg.org/spec/DDS/1.2/

Table 5.2: Sémantique de l'attribut "a" contenant les paramètres de DDS QoS

a= status-type SP qos-dds SP direction-tag avec : status-type=("desired"|"current"|"acceptable"|"unacceptable") SP= space qos-dds=("Deadline" "Latency" "Reliability" "Priority") direction-tag = ("send"|"recv"|"send-recv")

du champ "a" décrivant cette extension dans le Tableau 5.2 et la signification des différents paramètres dans le Tableau 5.3.

Table 5.3: Description des éléments du nouvel attribut "a" contenant les QoS policies de DDS

Champ	Contenu	Signification
status-type	desired current acceptable unacceptable	Le participant indique la QoS qu'il souhaite mettre en place. Informe le participant sur la QoS dans la session en cours. Indique que la QoS est acceptable et conforme au contrat O/R. Indique que la QoS ne satisfait pas le contrat O/R.
qos-dds	QoS DDS	"Deadline" "Latency" "Reliability" "Priority"
direction-tag	send recv send-recv	Indique la direction dans laquelle l'attribut est appliqué

Le Tableau 5.4 montre la description du jeton "qos-dds" (token dans la sémantique SIP) dans le message SDP. Les deux premiers paramètres indiquent la latence et le deadline admissibles par l'application, qui ont pour valeurs $20ms$. Le deuxième paramètre signifie que les données seront envoyées en mode non fiable, et finalement le dernier paramètre est le Transport_Priority qui représente le champ DSCP.

Table 5.4: Le nouveau jeton SDP contenant les paramètres de DDS QoS

token =/ qos-dds-send-recv-attr token =/ qos-dds-send-attr token =/ qos-dds-recv-attr qos-dds-send-recv-attr='qos-dds-send-recv' ' :'[[SP]qos-dds*(qos-dds))] qos-dds-send-attr='qos-dds-send' ' :'[[SP]qos-dds*(qos-dds))] qos-dds-recv-attr='qos-dds-send' ' :'[[SP]qos-dds*(qos-dds))] qos-dds = '0 20000000' '0 20000000' 'U' '32' qos-dds = '0 20000000' '0 20000000' 'U' '32'

Le jeton "qos-dds" identifie les paramètres de QoS pris en charge par les entités générant la description de la session. Un jeton qui apparaît dans l'attribut "qos-dds-send" identifie les politiques de QoS DDS prises en charge par le DataWriter pour assister le Publisher générant le message SIP/SDP afin de réserver les ressources en accord avec le trafic producteur. Lorsque le jeton apparaît dans l'attribut "qos-dds-recv", il identifie les politiques de QoS DDS supportées par le DataReader pour la réservation de la QoS. Lorsque le jeton comporte l'attribut "qos-dds-send-recv", il identifie les QoS DDS dans les deux sens pour le participant qui est à la fois Publisher et Subscriber. Ceci est rendu possible par le contrat R/O si les noeuds distants ont des QoS DDS compatibles, et ceci permet de garantir que les ressources demandées sont celles requises par l'application.

5.2.2.4 Sémantique de la nouvelle extension apportée à SIP/SDP

Pour que le participant SIP/DDS soit conforme à DDS, nous avons choisi de l'implémenter en Java avec les APIs du middleware Opensplice DDS qui sont aussi interopérables avec les APIs de RTI-DDS. Ce choix est motivé par un souci de conformité avec le proxy SIP qui est implémenté avec les librairies JAIN-SIP [98], elles implémentées pour être compatibles avec la RFC3261 [111] et la RFC3312 [23].
En plus, le client SIP/DDS implémente 4 Topics du flux simulation de Platsim_QoS décrivant les 4 classes de trafic spécifiées dans la section 3.4, ce qui nous a permis de respecter le même profil de trafic (profil de production) initial. En ce qui concerne les QoS policies de DDS, nous avons gardé les mêmes paramètres sauf pour le champ "Transport_priority" qui diffère. En effet, nous avons utilisé les valeurs définies par les services DiffServ (PHB) que nous avons déjà introduits dans le paragraphe 2.5.5.2.
Le paramètre QoS Latency_Budget permet au Publisher et au Subscriber de respecter le délai de transit inférieur ou égal à 20ms, mais cela suppose que le réseau sous-jacent offre une garantie de QoS réseau. Le paramètre "Deadline" engage l'application à adapter son rythme de production à celui voulu par le Subscriber. Le paramètre "Transport_Priority" et le paramètre débit (champs "b" du message INVITE) sont utilisés par le serveur de QoS pour configurer les files d'attentes du routeur de sortie du Publisher selon le service EF de DiffServ. Ces paramètres seront interceptés par le Proxy SIP amélioré et envoyés vers le serveur de QoS qui se chargera de réserver la QoS pour l'application.
Le Tableau 5.5 montre le message SIP envoyé par le participant 1 (Publisher) vers le participant 2 (Subscriber). L'attribut "qos-dds" est montré sur les deux dernières lignes. Lorsque le jeton "qos-dds-send-recv" est spécifié, le Publisher désire envoyer et recevoir le flux DDS. Lorsque le jeton "qos-dds-send" est spécifié, le Publisher veut envoyer uniquement les données DDS, et finalement lorsque le jeton "qos-dds-recv" est spécifié, ceci signifie que le participant veut recevoir seulement le flux DDS.

Table 5.5: Extrait du message INVITE avec le champ "a" des QoS policies de DDS

INVITE sip :ahkiri@laas.org SIP/2.0
Via :SIP/2.0/UDP 193.49.97.17
Via :SIP/2.0/UDP 193.49.97.81
From :sip :ahkiri@iptel.org
To :sip :ahkiri@laas.org
CSeq : 1 INVITE
o = akram 53655765 2353687637 IN IP4 193.49.97.18
s = -
c = IN IP4 ahkiri@laas.org
t = 0 0
b = AS :5120
a=qos-dds-send : current 0 20000000 0 20000000 R 46 send
a=qos-dds-recv : current 0 20000000 0 20000000 R 32 recv

Le Publisher présente ces paramètres de QoS pour réserver la QoS sur le réseau ; le Subscriber analyse cette requête selon deux points de vue : si les valeurs des QoS policies de DDS respectent le contrat R/O de DDS, et puisque les 4 paramètres utilisés sont modifiables, alors le Subscriber adapte ces QoS au rythme voulu par le Publisher. Ce comportement est géré par le champ

"status-type" dans le message SDP. En effet, le Publisher spécifie dans le champ "status-type", par le jeton "current", les paramètres de QoS DDS qu'il supporte dans le sens de production "qos-dds-send" et spécifie les QoS policies souhaitées "desired" dans l'attribut "qos-dds-recv". Le Publisher lui répond par un message dont le champ "status-type" est l'attribut "acceptable" si il peut adapter son rythme ou bien "unacceptable" si les QoS policies ne peuvent pas être modifiées. Chaque participant a plusieurs profils de QoS enregistrés dans le fichier de configuration XML que nous avons introduit dans la Section 3.4 ; chaque profil est spécifique à un comportement des entités DDS de l'application (Publisher, DataWriter, Subscriber, DataReader, etc.) et est chargé en appelant la fonction *create_datawriter_with_profile (topic_media, "platsim_Library", "MediaProfile", NULL, STATUS_MASK_NONE)* dont le troisième paramètre est le nom du profil à charger.

5.2.3 Les outils pour la gestion de la QoS

5.2.3.1 Proxy SIP conscient de la QoS

L'une des possibilités pour la configuration de la QoS au niveau des routeurs de bordures est l'utilisation d'un proxy SIP conscient de la QoS en s'inspirant des concepts présentés dans la section 2.5.8. Pour cela, nous avons apporté toutes les modifications au proxy Jain-SIP [97] pour qu'il supporte la QoS DDS.
Concernant les nouvelles implémentations à réaliser sur le Proxy SIP, il faut d'abord lui permettre de reconnaître un message REGISTER spécifique lui indiquant qu'il doit prévenir un ou plusieurs autres Proxies SIP. Pour cela, à la réception du message REGISTER, il doit vérifier que les champs "From" et "To" sont différents et que les deux SIP URI sont de la forme "*sip* : *user_name*@*domain_name*". Les deux *domain_name* doivent être différents pour s'assurer qu'il achemine le message REGISTER vers le proxy SIP distant correspondant au champ "To". En plus des fonctions classiques de relais entre applications SIP, le proxy SIP à QoS a la capacité de configurer la QoS sur son domaine. Le proxy doit intercepter le descripteur des sessions dans les messages SDP pour en déduire les caractéristiques de la session QoS du Topic DDS qu'il va installer. Les messages SIP doivent alors dans tous les cas passer par ou moins deux proxy SIP améliorés pour la mise en place de la QoS de bout-en-bout. Les deux Proxies sont en mode "*Assured*" dans lequel l'établissement de la session ne se fait que si la réservation de QoS a bien été réalisée. Au contraire, dans le mode "*enabled*", l'établissement de la session avec une réservation des ressources n'est pas une condition bloquante. En effet, la spécification de DDS nécessite que le Publisher et le Subscriber se mettent d'accord sur les paramètres de QoS DDS afin que la communication entre eux puisse s'établir. Pour cela, il doit d'une part fonctionner en mode "stateful" pour maintenir les informations d'états de session, et d'autre part garder les traces des messages échangés ce qui nécessite la configuration du champ "Record Route". Cette solution requiert un serveur de QoS déjà installé qui, après avoir été sollicité par le proxy, se charge de configurer ces routeurs.

5.2.3.2 Le serveur de QoS amélioré

L'outil communément utilisé pour l'installation des politiques de QoS sur le réseau de sortie est le serveur de QoS qui a pour fonction principale de recevoir et analyser les messages de réservation et de libération des ressources en provenance du proxy SIP. Nous avons décrit dans

le paragraphe 2.5.7 les propositions qui ont été faites pour le choix du serveur de QoS. Nous avons choisi celui basé sur le protocole COPS-DRA parce que, d'une part il est dédié aux architectures de QoS implémentant DiffServ, et d'autre part parce que le nombre de messages échangés est plus faible que pour les autres solutions (qui augmentent la durée nécessaire à l'établissement de la session de QoS).
En effet, les modifications apportées au proxy SIP touchent les fonctions qui permettent la mise en place de la reservation des ressources garantissant une QoS appropriée au préalable de l'ouverture de la session SIP, ceci en introduisant une précondition de type reservation de resource via un module qui réalise les procédure de réservation et de libération de ressources en échangeant des messages Request (RQT), Décision (DEC), Report (RPT) avec le serveur QoS, et un module d'extraction et d'analyse de descripteur de session SDP.
De plus, une fois les caractéristiques de la session QoS (@IP, N° ports, attributs QoS DDS, descripteur champ DSCP, descripteur du media, etc. illustré dans le Tableau 5.5) déduites des descripteurs de session SDP, chaque proxy SIP se charge de communiquer ces informations à son serveur de QoS le plus proche. Chaque fois que le serveur de QoS reçoit une requête de réservation/libération en provenance du proxy SIP, il compare les adresses source et destination et le numéro du port du flux concerné contenu dans le message de réservation/libération avec les adresses enregistrées dans sa base. Si ces adresses correspondent, il prévient le proxy SIP de la réservation/libération des ressources par un message RESV/FREE en indiquant les paramètres correspondant aux flux et le service auquel le flux doit être associé, ainsi que le débit que nous l'avons décrit dans le paragraphe 5.2.2.1.

5.2.3.3 Configuration de la QoS au niveau routeurs

Pour la configuration de la QoS au niveau IP, lorsque le serveur de QoS reçoit une requête de réservation, il doit modifier les paramètres des différentes files d'attentes du routeur de sortie pour configurer le débit nécessaire pour l'application.
Cependant, nous ne disposons pas des informations à partir de l'application DDS ou de l'interface du service networking de DDS concernant le taux de perte des paquets et la gigue maximale admissible. L'approche suivie est alors de favoriser le traitement des données ayant des exigences les plus strictes au niveau des routeurs. Ainsi, pour la mise en oeuvre de la QoS au niveau IP, nous utilisons 3 catégories DiffServ principales, dont une EF et une AF avec deux sous-catégories.
La classification des paquets permet d'organiser les éléments du réseau en plusieurs flux de trafic et de mettre les politiques en application selon le champ de QoS de couche 3. Le routeur commence par identifier les flux de trafic ou les groupes de paquets. Il classifie ensuite ou reclassifie ces groupe à l'aide du champ DSCP. Pour cela, 4 classes de trafic sont identifiées à la base des concepts retenus dans la mise en place de la communication DDS sur des réseaux locaux. Ces différentes classes de trafic sont identifiées dans le Tableau 5.6.

- Pour les Topics périodiques, nous avons alloué le service du PHB-EF,
- Les Topics "State" et "Contrôle" ont des contraintes temps-réel plus lâches, auxquelles nous avons associé le traitement AF3.
- Les Topics multimédia ne nécessitant pas un service temps-réel dur, nous leur avons attribué le traitement AF1.

– Le trafic du Build-in Topic ne nécessitant aucun traitement particulier, nous lui avons attribué la valeur DSCP 0.

Topics DDS	PHB	classe de trafic	probabilité de suppression	DSCP	
				Binaire	Décimal
Périodiques	EF	RT		101110	46
State et Controle	AF3	NRT1	Low	010100	28
MultimediaMedia	AF2	NRT2	Medium	001100	20
Build-in Topic	BE	STD	High	000000	0

Table 5.6: Correspondance entre les Topics DDS, les classes de trafic et leurs valeurs DSCP

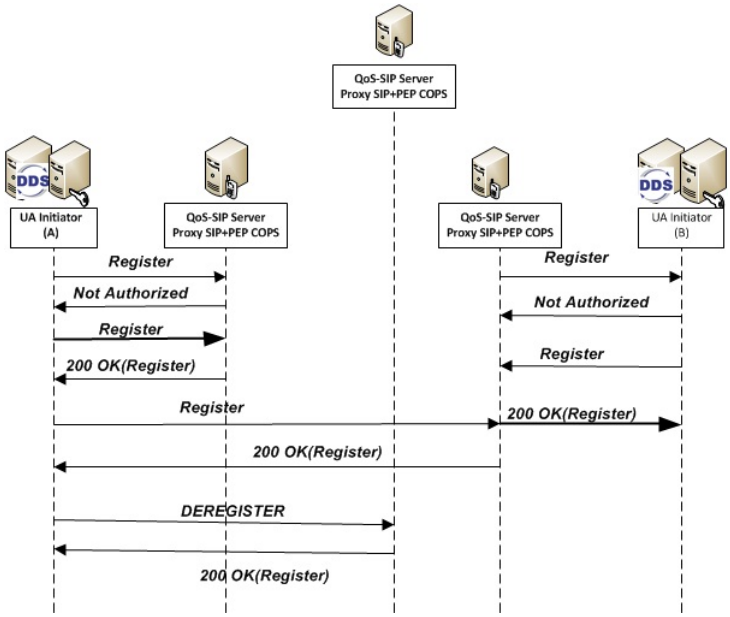


Figure 5.2: Enregistrement initié par le Proxy SIP amélioré local

5.2.4 Mise en place de la signalisation pour gérer la QoS

5.2.4.1 Enregistrement et Réenregistrement de session

Une fois que le client SIP est configuré pour émettre et recevoir des appels, le traitement commence avec la phase d'enregistrement dans le «REGISTRAR» (enregistrement des utilisateurs dans la base de données de localisation SIP). Nous avons considéré le cas où un enregistrement doit être fait au niveau des deux Proxies des domaines de chaque réseau : le proxy SIP local qui se trouve dans le même domaine que le client qui initialise la requête d'enregistrement, et le proxy SIP se trouvant dans le domaine distant auquel appartient le participant en réception des données DDS. La Figure 5.2 présente cette configuration simple mais largement suffisante pour un scénario où deux simulateurs distants se trouvent dans deux domaines IP différents. En plus, l'envoi du message REGISTER au proxy en charge du domaine distant permet une

utilisation quasi optimale des liens en offrant un temps d'enregistrement/des-enregistrement inférieur à 100ms.

5.2.4.2 Établissement de la Session

Nous avons décrit dans le paragraphe précédant le message *Invite* qui contient les informations nécessaires (@IP, N° ports, attributs QoS DDS, descripteur champ DSCP, descripteur du media,...) à la réservation des ressources satisfaisant la QoS requise par l'application DDS. Le proxy SIP local ajoute l'adresse d'enregistrement (AdressOfRecord) pour forcer les messages à le traverser. Une fois l'adresse destination trouvée dans sa base de donnée (il s'agit d'une base locale qui permet d'éviter l'utilisation d'un serveur de domaine ou *DNS* qui ne rentre pas dans le cadre de cette thèse), il redirige le message vers la destination.
En effet, le participant DDS1 offre une liste de profils QoS DDS qu'il est capable de supporter

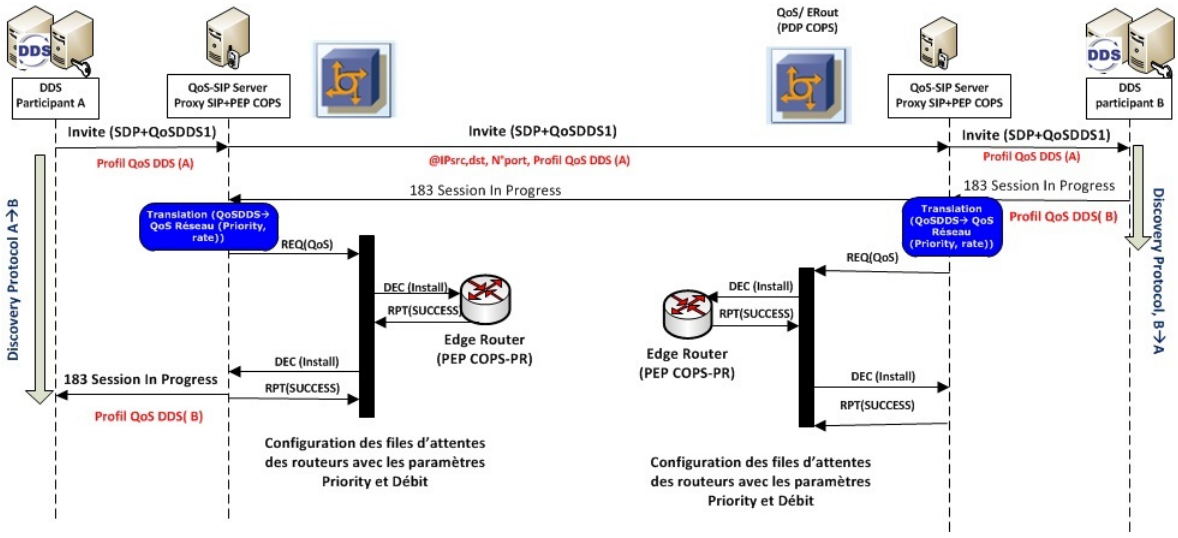


Figure 5.3: Déroulement normal de l'établissement de la session avec succès

durant la communication ainsi que l'adresse source et le N° de port sur lequel il est prêt à recevoir. Le participant DDS2 (initialement non conscient de la QoS) reçoit cette liste, la croise avec la sienne et se contente de choisir arbitrairement le premier profil QoS DDS compatible avec sa liste, ainsi que l'adresse IP et le N° port sur lesquels il est prêt à recevoir, et renvoie la réponse avec le paramètre "acceptable". Lorsque le participant DDS2 reçoit le message INVITE, il consulte la liste des profils supportés par l'appelant, et deux cas peuvent se produire :

1. Cas 1 :
   - ce scénario est illustré par la Figure 5.3. Le participant DDS distant supporte au moins un profil de QoS Policies de DDS qui respecte le contrat requête/offre qu'il peut utiliser (il a répondu par « acceptable ») ; il accepte la communication en répondant par le message "183 Session In Progress". Le proxy SIP distant reçoit ce message et lance la réservation des ressources et l'achemine vers le premier Proxy SIP dans le réseau du participant DDS1. Ainsi la session DDS peut s'établir. Le participant DDS 1 envoie le message PRACK et reçoit un message OK. Le participant DDS2 acquitte l'établissement

de la session et renvoie le message OK. Le flux DDS peut être dirigé vers la classe de service fournissant le support de QoS.

- Lorsque le proxy SIP reçoit le message INVITE, il en extrait le profil de QoS pour en déduire la priorité (Transport_Priority) et le débit de la transmission associés au profil de QoS DDS (Si le débit de transmission est assez élevé par rapport aux ressources disponibles, il demande à l'émetteur de changer le paramètre de QoS DDS "Deadline" pour agir sur le temps inter-échantillons et demander la diminution de la fréquence d'envoi des Topics (en changeant le profil de QoS DDS)).
- Le proxy SIP 2 reçoit ce message et constate que le client 2 que l'on cherche à joindre est à sa charge. Il lui transmet alors le message *Invite*. Les messages 'Session In Progress' sont envoyés à l'entité précédente pour la prévenir que le message *Invite* a bien été reçu. Lorsque le proxy SIP 2 reçoit le message 'session in progress', il le renvoie au client 1 pour lui indiquer que le client 2 a bien été informé de la session. Ensuite, le proxy SIP 2 envoie une requête de reservation de ressources (DEC ou Decision, RPT ou Report, REQ ou Request) au serveur QoS de son domaine. Ce dernier négocie la QoS avec l'équipement réseau de ce domaine pour réserver les ressources nécessaires. De même, ces informations sont stockées par le proxy SIP2 pour garder une trace de la session en cours, appelée *QoS Etat*. Il envoie en conséquence une requête QoS au COPS-PDP de ce domaine pour demander la réservation de QoS à la bordure du réseau.

2. Cas 2 : Le participant DDS2 ne supporte aucun profil de QoS DDS qui permette la mise en place de la QoS entre les deux participants. Face à cette situation, deux possibilités sont envisageables :
   - aucune QoS ne sera mise en place au niveau du réseau. Ce scénario est illustré par la Figure 5.4 : c'est le service par défaut sans QoS qui peut s'établir au niveau réseau.

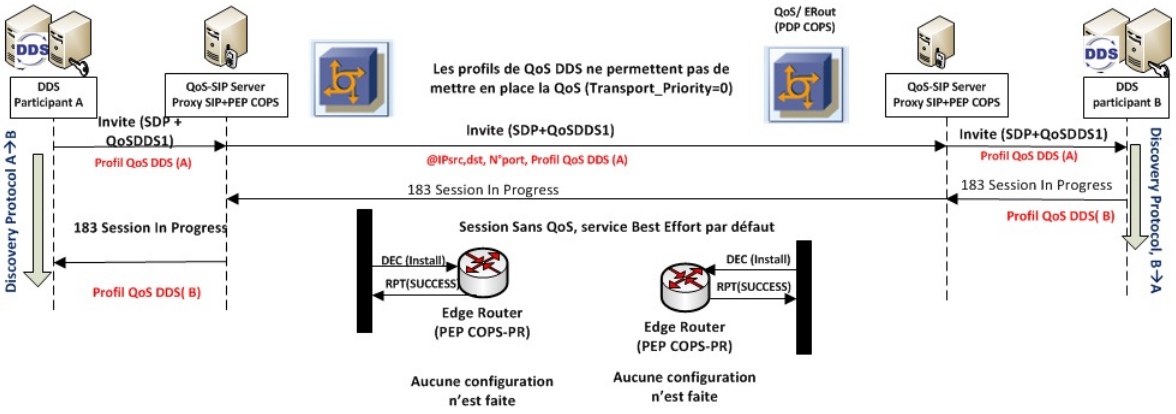


Figure 5.4: Établissement de la session sans QoS

- L'établissement de la session ne se produit pas et par conséquent le Proxy SIP renvoie un message de rejet (le paramètre "unacceptable" implique ce rejet) de la communication au simulateur DDS1 et la communication ne démarre pas (ex. violation des QoS DDS policies).

5.2.4.3 Terminaison de Session et Libération des Ressources

Lorsque un participant DDS souhaite mettre fin à la session, il envoie un message *BYE* au correspondant qui répond par un *200OK* (pour BYE). La procédure de dés-enregistrement auprès du proxy SIP est montrée sur la Figure 5.5. Elle est réalisée de la même façon que pour l'enregistrement : un message REGISTER, quasiment semblable au message d'enregistrement est envoyé avec comme seule différence un champ « Expires » à 0. Les routeurs de bordure doivent libérer les ressources allouées après la fin de session. Ensuite, lorsque le message BYE est échangé, le proxy SIP envoie bien un message FREE au Serveur de QoS. En particulier, le message "Uninstall" est envoyé pour effectuer cette opération.

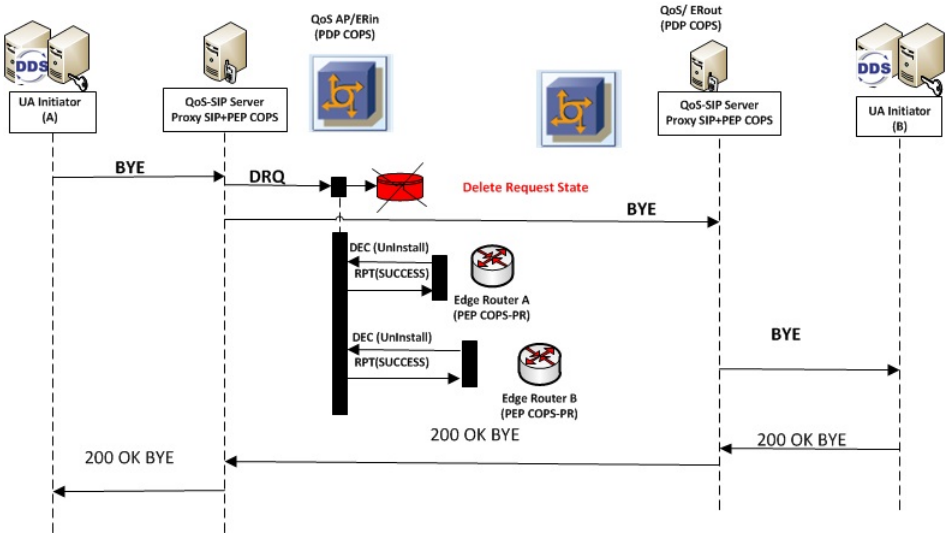


Figure 5.5: Terminaison de Session et Libération des Ressources

5.2.5 Conclusion sur la signalisation de la QoS DDS avec SIP

Dans cette section, nous avons présenté notre approche pour l'allocation dynamique des ressources pour les applications DDS en proposant l'enrichissement des messages SDP pour supporter les QoS policies de DDS. L'enrichissement de SIP pour l'intégration à des réseaux offrant une garantie de QoS propose un mécanisme simple pour la gestion de la QoS sur une architecture DiffServ en implémentant un contrôle par une politique d'admission dynamique COPS. Toutefois, il est laissé ouvert un certain nombre de problèmes qu'il est nécessaire de résoudre pour obtenir une garantie générale de QoS de bout-en-bout, à savoir :

- l'intégration de mécanismes de réservation de ressources au sein de l'établissement de session SIP souffre encore du problème de passage à l'échelle pour le support de milliers de clients dans une session. Même si certaines approches proposent l'utilisation du protocole Jabber [113] pour résoudre ce problème, il est nécessaire d'avoir un mécanisme de contrôle d'admission pour la gestion des requêtes concurrentes venant de différents clients SIP.
- SIP permet, pour des raisons de sécurité, de crypter le corps du message (session SDP) et une partie des entêtes qui permettent de diriger le message. Toutefois, cette opération présente une difficulté : le proxy SIP impliqué dans la mise en place de la QoS doit récupérer

des informations contenues non seulement dans les entêtes des messages, mais aussi dans le corps du message qui contient les informations sur la session DDS. Même s'il est possible d'envisager que le proxy SIP puisse réaliser des opérations de cryptage/décryptage, cette opération est très complexe dans un environnement mobile où les entités réseaux intermédiaires ne sont pas figées, ce qui augmente les délais de bout en bout et pose un problème de sécurité.

– Mais, par sa facilité de mise en œuvre, la solution SIP peut être considérée comme une solution à court terme. En effet, les différentes techniques utilisées pour l'intégration du contrôle par des politiques avec un établissement de sessions via le protocole SIP ne supportent pas la communication multicast, alors que DDS a été conçu pour permettre à plusieurs applications DDS situées sur des domaines IP différents de communiquer entre elles. Nous verrons dans la section qui va suivre une solution possible pour résoudre ce problème.

5.3 Interconnexion sur des réseaux Internet multi-domaines à QoS

Dans cette dernière contribution, nous proposons une solution à long terme basée sur l'architecture Internet du futur du projet EuQoS. Pour cela, nous présenterons, dans un premier temps, l'architecture générale du projet EuQoS en introduisant ses notions de base, sa structure et ses composants globaux. Ensuite, nous présenterons l'intégration de l'application de simulation distribuée sur la plateforme EuQoS à travers l'exemple du projet PlatSim_QoS. Nous présenterons alors le mécanisme de signalisation de la QoS, le contrôle d'admission et le provisionnement des ressources permettant de satisfaire les contraintes de l'application.

5.3.1 Architecture EuQoS pour la garantie de QoS

La motivation principale du projet EuQoS [20] est liée à l'utilisation croissante de l'Internet comme infrastructure globale de communication et à la volonté des opérateurs d'offrir de nouveaux services à valeur ajoutée avec QoS garantie [19]. EuQoS vise à fournir une architecture permettant de garantir la qualité de service de bout-en-bout dans un environnement le plus général possible, soit multi-technologies, multi-domaines et multi-services, et de supporter un large spectre d'applications comme la voix, la vidéoconférence, la vidéo à la demande, la téléingénierie, ainsi que des applications de télémédecine et télé-Learning.
L'architecture EuQoS est constituée d'une collection de composants qui coopèrent afin de fournir de la QoS de bout-en-bout. Ces composants intègrent un large ensemble de mécanismes, divers et cohérents, totalement coopératifs, permettant l'autorisation, l'authentification, la négociation de service, le contrôle d'admission, la signalisation, la surveillance, la métrologie, l'ingénierie de trafic et l'optimisation des ressources. Ils utilisent aussi des mécanismes pour effectuer d'une manière abstraite et virtuelle, indépendante des technologies, la signalisation, la négociation, la gestion et la fourniture de la QoS de bout-en-bout.
EuQoS gère la signalisation globale, en particulier de niveau applicatif sur le premier et le dernier domaine. Les applications doivent exprimer leurs besoins pour pouvoir établir, maintenir et fermer les sessions avec des garanties de QoS. De plus, EuQoS gère aussi la signalisation au

niveau réseau sur tous les domaines tout au long du chemin réseau de bout-en-bout, en utilisant le protocole de signalisation NSIS et en proposant une signalisation originale découplée du chemin des données. Les fonctions de signalisation interviennent dans les phases de réservation, de contrôle d'admission, de transfert et de libération des ressources.

5.3.1.1 Interfaces de EuQoS

Le système EuQoS a été conçu selon une découpe en deux axes : un axe vertical (Plans de Service, Contrôle et Transport) et un axe horizontal (différentes technologies pour les réseaux d'accès et de cœur, différents domaines et systèmes autonomes). Selon cette approche, le déploiement de l'architecture (Figure 5.6), permet de réduire la complexité du problème en séparant les interactions entre le client et le serveur, entre les différents plans et entre les sous-systèmes d'EuQoS sur des systèmes autonomes différents.

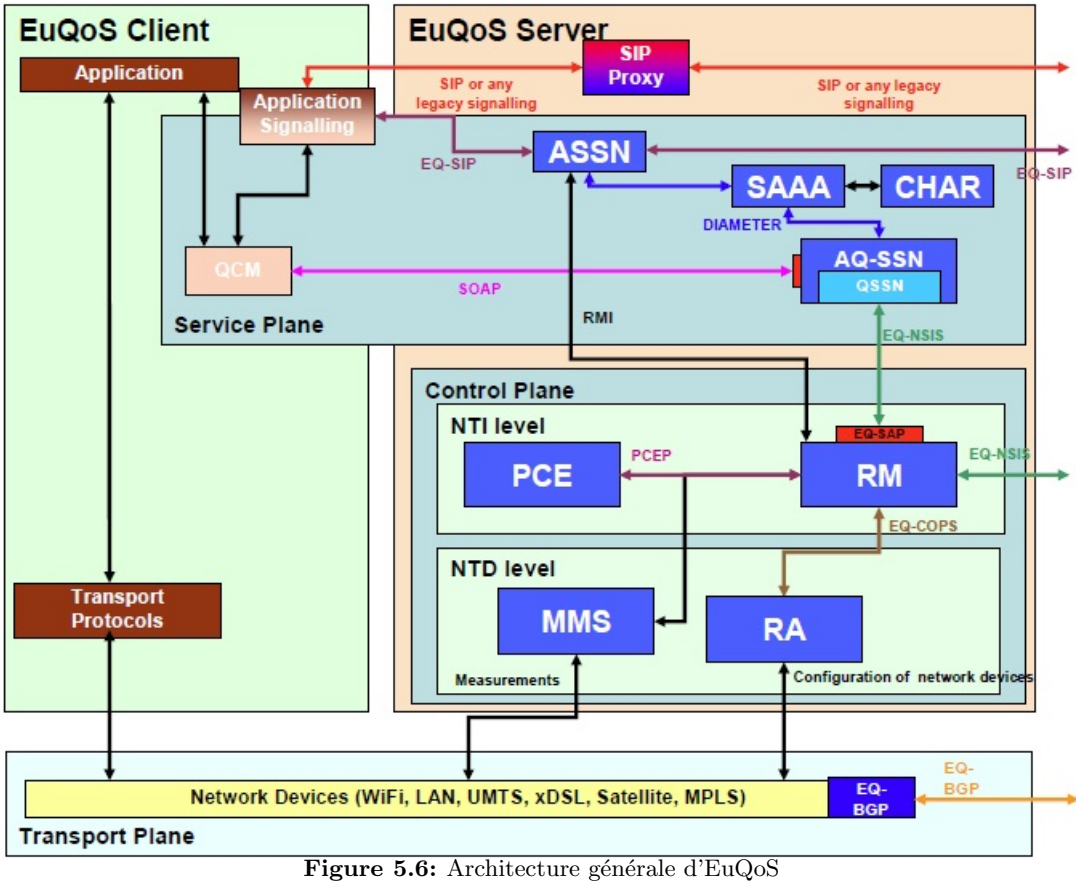


Figure 5.6: Architecture générale d'EuQoS

- **Le plan de service** permet l'accès aux différents services de QoS, et d'établir la communication entre les applications et le serveur de QoS.
- **Le plan de contrôle** assure la gestion de la QoS de bout-en-bout. Dans l'architecture EuQoS le plan de contrôle est composé de deux niveaux :
  - Un niveau indépendant de la technologie sous-jacente (NTI) : les interactions entre les niveaux NTI nécessaires pour assurer la QoS de bout en bout sont réalisées par le protocole NSIS [55]. Nous n'avons pas utilisé dans ce mémoire le niveau indépendant

parce que, d'une part, nos travaux sont destinés aux réseaux filaires, et d'autre part car les routeurs actuels ne supportent pas le protocole NSIS.
  - Un niveau dépendant de la technologie sous-jacente (NTD) : il permet la réservation/libération, le provisionnement des ressources, et la configuration des éléments du réseau. Il comprend aussi les éléments de supervision, de mesure et de contrôle du système. Cette proposition a été retenue dans cette thèse pour la gestion de la QoS.
- **Le plan de transport** est composé par les équipements et les mécanismes gérés par le plan de contrôle. Son rôle est de créer et gérer le chemin transportant les données à travers les différents réseaux et ASs.

5.3.1.2 Les composants logiciels :

L'architecture générale d'EuQoS, illustrée à la Figure 5.6, est composée de deux blocs fondamentaux. Le client EuQoS se situe au niveau du terminal utilisateur et le serveur EuQoS implémente les trois plans présentés précédemment. Sur le plan de service se trouvent les trois modules suivants :

- **AQ-SSN (Application Quality Service Signaling Negotiation) :** fournit une interface multi-protocoles permettant l'accès au service EuQoS pour ses utilisateurs.
- **CHAR (Charging) :** permet la tarification du service.
- **SAAA :** supporte l'autorisation, l'authentification des utilisateurs et la facturation. Il assure la communication des informations de facturation vers le module CHAR et échange les messages avec AQ-SSN par le protocole Diameter.

Le plan de service est formé par les deux modules NTI et NTD :

- Le niveau NTI comporte le Resource Manager (RM) en charge de la gestion du domaine et le Path Computation Element (PCE) [44] en charge du provisionnement réseau. Ces deux éléments communiquent via le protocole PCE.
- Le niveau NTD se charge de la gestion spécifique de la QoS selon la technologie sous-jacente. Il comporte le Resource Allocator (RA) qui se charge du traitement des requêtes de QoS pour assister la supervision et la métrologie du réseau (Monitoring et Mesurement System (MMS)). La communication entre le RA et le RM est faite par le protocole EQ-COPS.

Le plan de transport permet d'assurer le transfert des données de bout en bout. Il permet de transporter les informations du client par les interfaces du NTD (RA et MMS).

5.3.2 Interconnexion de Proxy DDS sur Internet à QoS

Nous avons montré dans la Section 4.3.4 le fonctionnement du Proxy DDS sur les réseaux grandes distances sans QoS. Dans ce paragraphe, nous décrivons les extensions que nous avons réalisées sur le Proxy DDS afin qu'il puisse profiter des services disponibles sur les réseaux implémentant déjà des mécanismes de gestion de la QoS réseau.

5.3.2.1 Support de deux profils de QoS : LAN/WAN

Comme nous l'avions présenté dans la section 4.3.4, les Proxies DDS permettent dans le cas d'une architecture similaire à celle de la Figure 5.7, de réduire au minimum la transmission d'informations sur le réseau WAN, entre Proxies, et de rediffuser ces données en local. Dans la

même idée, nous allons profiter de cette caractéristique pour introduire une gestion différente de la QoS, selon que l'on soit dans un réseau local ou sur un réseau WAN.
Pour répondre à cette problématique, l'architecture du proxy DDS doit alors être capable de gérer deux profils de QoS différents, l'un pour gérer les simulations locales, et l'autre pour gérer la communication inter-Proxies. Pour cela, nous avons alors proposé (Figure 5.7) deux profils de QoS différents pour chaque réseau :

- en LAN : ce profil de QoS DDS assure la redistribution des données envoyées par les Proxies distants vers les simulations locales. Il a pour objectif d'assurer la compatibilité (respect du contrat Request/Offer) entre les différentes simulations locales de chaque domaine DDS situé sur un domaine IP distant. C'est-à-dire qu'il se charge d'assurer l'interopérabilité entre les simulations distribuées fonctionnant sur des réseaux distants.
- en WAN : ce profil de QoS DDS est utilisé pour assurer la distribution de données inter-Proxies.

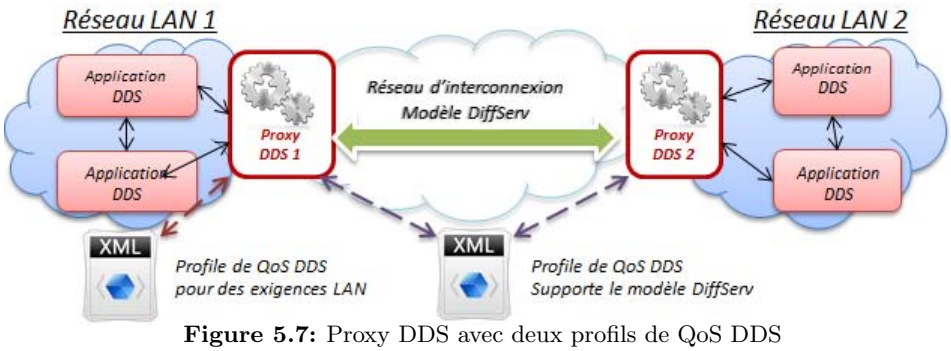


Figure 5.7: Proxy DDS avec deux profils de QoS DDS

Un intérêt particulier est donné aux politiques de QoS régissant la communication à grande distance. En effet, le proxy DDS permet de remarquer les paquets reçus contenant des Topics DDS en changeant la valeur du champ DSCP en sortie (en redistribution) à l'aide de la politique de QoS " Transport_Priority ". Par exemple, sur la Figure 5.8 il est possible de spécifier qu'un Topic X reçu par son DataReader avec une valeur de transport priorité 46 correspondant à un PHB EF sera retransmise avec un DataWriter qui porte la valeur du transport priority 10 AF11.

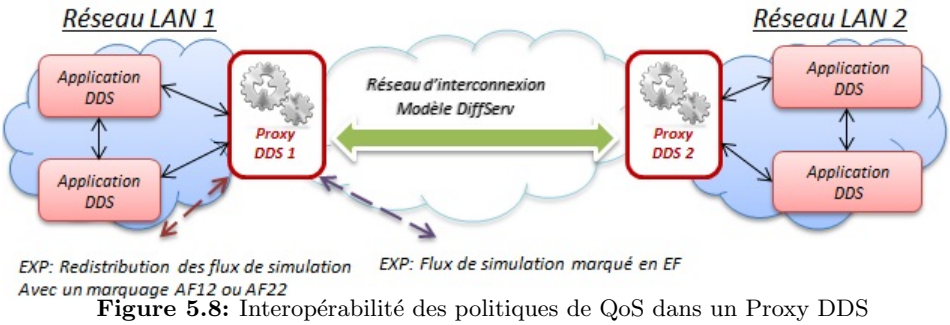


Figure 5.8: Interopérabilité des politiques de QoS dans un Proxy DDS

5.3.3 Définition de la politique de QoS réseau pour Platsim_QoS sur DDS

Pour l'implémentation de la QoS pour PlatSim_QoS sur DDS sur une architecture DiffServ dans un réseau multi-domaines, nous avons utilisé le serveur de gestion de la QoS proposé dans le cadre du projet EuQoS appelé "Velox". Velox gère la signalisation et la réservation des ressources au niveau réseau. Nous décrivons dans cette section l'architecture de Velox et son utilisation pour la mise en place de la procedure de signalisation dans le projet PlatSim. Ensuite, nous présenterons le mécanisme de réservation de ressources et de contrôle d'admission. Enfin, nous décrirons la mise en place du chemin de données permettant le transfert des données de simulation entre les différents participants.

5.3.3.1 Mapping de la QoS réseau à l'aide du middleware DDS

Rappelons que dans le chapitre 3, nous avons défini les différentes classes de trafic pour la fourniture de la QoS sur des réseaux locaux. Dans le même esprit, nous avons réutilisé ces classes de trafic pour assurer la signalisation de la QoS de bout en bout entre réseaux hétérogènes. Notons aussi que le marquage des paquets au niveau du middleware DDS est géré par la politique de QoS DDS "Transport_Priority". Ce paramètre permet de spécifier à l'application le niveau de priorité (l'attribut DiffServField) à prendre en compte lors de la distribution des échantillons (au niveau du profil de production), niveau qui sera par la suite converti en champs DSCP pour indiquer le traitement adéquat au niveau réseau.

5.3.3.2 Le serveur de QoS Velox

Velox est un gestionnaire de bande passante qui permet la gestion de la QoS et le contrôle des ressources réseau via des Web services [1]. Son objectif est de cacher la complexité du réseau sous-jacent pour offrir à l'utilisateur, par exemple de la plateforme Platsim_QoS ou à l'administrateur du réseau, des services stables permettant d'installer, de maintenir et de libérer les ressources de bout-en-bout avec plusieurs niveaux de QoS différents.
En effet, Velox est exploitable pour la gestion des ressources en réseau mono-domaine et/ou réseaux multi-domaines. Dans ce dernier cas, Velox communique en mode point-à-point avec d'autres serveurs de QoS et avec les routeurs de sortie de chaque domaine. Il permet de négocier et déployer des services de QoS différents (EF, AF, etc.) entre les différents domaines. Il offre une grande granularité de classes de services, gérées par deux types de services :

1. **Network Service :** représente une voie qui comporte un ou plusieurs routeurs. Lors de la création d'un service réseau une bande passante doit être assignée pour cette voie, permettant à plusieurs sessions réseaux d'utiliser cette bande passante. Plusieurs "Network Services" peuvent être créés et sont distingués par 7 couleurs différentes. En outre, il est possible d'associer des scripts pour les routeurs qui ont été affectés à un service réseau donné. Ces scripts sont utilisés pour configurer correctement les routeurs lors du déclenchement du service.
2. **Session Service :** Correspond à un type de service ou bien à une classe de service DiffServ dans un "Network Service" donné. Une bande passante doit être assignée à chaque session et sera allouée par le service réseau lors du déclenchement du service "Trigger" de Velox.

1. `http://www.w3.org/`

La Figure 5.9 illustre l'utilisation d'un tunnel MPLS entre les deux routeurs *Posets et Montperdu* pour la réservation des ressources pour PlatSim_QoS.

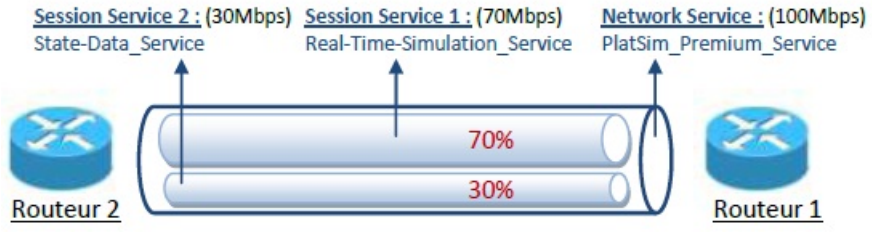


Figure 5.9: Configuration du lien entre les routeurs Posets et Montperdu

Velox utilise aussi un service que nous utilisons pour le déclenchement de la réservation de la bande passante (voir Figure 5.10). Pour chaque "user channel" que nous créons avec le middleware DDS nous créons aussi un service de session (session service) pour la publication du profil de production de Platsim_QoS dans le service Trigger. Cela permet à l'application (Platsim_QoS) de spécifier ses besoins en QoS au réseau sous-jacent.

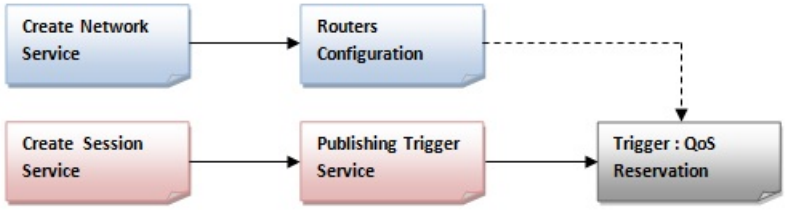


Figure 5.10: Service de déclenchement de la réservation des ressources

5.3.3.3 Définition de la politique de QoS réseau

Dans cette partie nous nous focalisons sur la définition d'une politique de QoS réseau adaptée à l'architecture DiffServ. La définition de cette politique se base essentiellement sur les mécanismes de QoS appliqués au niveau des routeurs de bordure (Posets et Montperdu dans notre expérimentation). Les mécanismes mis en oeuvre sont : la classification de trafic, la gestion des files d'attente, la politique de rejet, le " policer " et l'ordonnancement. La configuration suivante a été mise en place :

```
Class-of-service {
Classifier {
```

D'abord un classificateur DSCP est défini et permet d'assigner chaque paquet entrant à une classe de service appelée "Forwarding Class". Dans chaque classe, différentes valeurs possibles du champ DSCP ont été réparties avec une valeur significative de "Loss Priority".

```
dscp platsim_dscp_class #Classes de données PlatSim
  {
    forwarding-class best-effort #Built-in Topic Data
      { loss-priority high code-points [000000];# BE   DSCP 00 }
```

```
    forwarding-class expedited-forwarding #Periodic Real-Time Simulation Data
      { loss-priority low code-points [101110]; # EF   DSCP 46 }
    forwarding-class assured-forwarding #State Data/Control Data /Audio-video Data
      { loss-priority low code-points    [001010];# AF11 DSCP 10
        loss-priority low code-points    [001100];# AF12 DSCP 12
        loss-priority low code-points    [001110];# AF13 DSCP 14
loss-priority medium code-points [010010];# AF21 DSCP 18
        loss-priority medium code-points [010100];# AF22 DSCP 20
        loss-priority medium code-points [010110];# AF23 DSCP 22
        loss-priority high code-points   [011010];# AF31 DSCP 24
        loss-priority high code-points   [011100];# AF32 DSCP 26
        loss-priority high code-points   [011110];# AF33 DSCP 28
      }
     forwarding-class network-control
      { loss-priority low code-points [110000]; # NC1/CS6 DSCP 48
        loss-priority low code-points [111000]; # NC2/CS7 DSCP 56
      }
   }
```

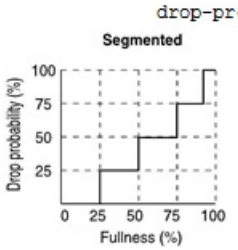
Les files d'attente du routeur sont créées à partir du classifier. Pour pouvoir les utiliser, il est nécessaire de les mapper dans la section " Forwarding-classes " avec les quatre files existantes dans la configuration du routeur (Le routeur Juniper M7i supporte des files dans l'ordre de 0 à 3). Si le classificateur ne possède qu'une file d'attente, alors il suffit de remplir uniquement la file 0 et les autres files sont toujours présentes mais non utilisées. Il est à noter que la file Network-Control est appropriée pour le routeur Juniper et est responsable du transfert des données de signalisation et des messages d'état des liens des routeurs.

```
forwarding-classes
  {   class best-effort queue-num 0;
      class expedited-forwarding queue-num 1;
      class assured-forwarding queue-num 2;
      class network-control queue-num 3;
    }
```

Il est nécessaire ensuite de fixer le classificateur défini à l'interface d'entrée du routeur et l'ordonnanceur à l'interface de sortie. Prenant l'exemple sur Montperdu :

```
interfaces {
ge-0/0/0 #vers Vlan 203 {
    unit 0 {
        classifiers {
            dscp platsim_dscp_class;
        }
    }
}
ge-1/3/0 #vers Posets {
    scheduler-map platsim-sched;
    }      }
}
```

La section " scheduler-maps " permet d'appliquer une politique de gestion particulière à chaque file d'attente en termes de bande passante et de taille de la file.




```
drop-profiles
    {
        Tail-Drop
            {       fill-level 90 drop-probability 0;
                    fill-level 95 drop-probability 100; }

        segmented-RED
            {       fill-level 25 drop-probability 25;
                    fill-level 50 drop-probability 50;
                    fill-level 75 drop-probability 75;
                    fill-level 95 drop-probability 100; }
    }
```

Figure 5.11: Politiques de gestion des files d'attentes

```
    scheduler-maps
        {  platsim-sched
             {
               forwarding-class expedited-forwarding scheduler EF;
               forwarding-class assured-forwarding scheduler AF;
               forwarding-class best-effort scheduler BE;
               forwarding-class network-control scheduler NC;   }
        }
```

Les caractéristiques de chaque politique de gestion sont définies dans la section "schedulers" où est précisé pour chaque queue la bande passante de sortie avec un "Transmit-rate" et une priorité entre (low, high et strict-high). Ces priorités permettent de spécifier le poids de chaque queue pour l'ordonnanceur WFQ et finalement une politique de rejet. Il est possible d'appliquer les politiques Tail-Drop ou RED. Cette caractéristique est montrée sur la Figure 5.11.

```
schedulers{
BE { transmit-rate 400K exact;
    priority low;
    drop-profile-map loss-priority high protocol any
                     drop-profile segment-RED; }
EF { transmit-rate 300K exact;
    priority strict-high;
    drop-profile-map loss-priority low protocol any
                     drop-profile segment-RED; }
AF { transmit-rate 200k exact;
    priority high;
    drop-profile-map loss-priority medium protocol any
                      drop-profile segment-RED; }
NC { transmit-rate 100k exact;
    priority high;
    drop-profile-map loss-priority low protocol any
                     drop-profile segment-RED; }
    }
```

La Figure 5.11 décrit la dernière section "Drop-profiles" qui permet de fournir les caractéristiques de chacune des politiques de gestion des files d'attente. "fill-level" définit le pourcentage d'occupation de la file et " drop-probability " la probabilité de suppression d'un paquet.
La politique Tail-Drop est configurée de telle sorte qu'entre 0 et 90% d'occupation de la file aucun paquet n'est supprimé puis à partir de 95% tous les paquets sont supprimés. La politique "Segmented-RED" met en place une politique de suppression des paquets par palier.

5.3.4 Mise en oeuvre de la QoS pour PlatSim_QoS dans un réseau multi-domaines à QoS

5.3.4.1 Signalisation de la QoS

PlatSim_QoS tire alors profit d'EuQoS pour faciliter la signalisation de la QoS pour le gestionnaire de ressources réseau qui crée des chemins à QoS garantie (EQ-PATH) associés à un ensemble de paramètres de QoS.

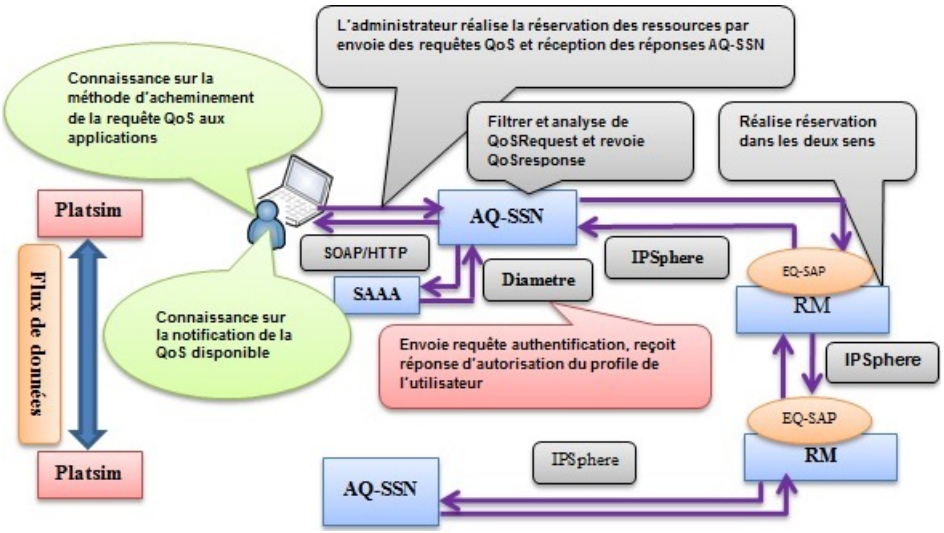


Figure 5.12: Signalisation de la QoS à la demande de bout-en-bout pour PlatSim_QoS

L'application PlatSim_QoS peut établir la session en envoyant une requête de QoS à travers les mécanismes de signalisation fournis à l'utilisateur par une interface web, qui se charge de négocier les paramètres de QoS avec le gestionnaire de ressources dans son domaine IP. La Figure 5.12 montre que l'application Platsim_QoS réalise la signalisation par un service de gestion de la QoS à la demande. Le composant AQ-SSN est présenté à l'administrateur du domaine pour établir, modifier, mettre à jour et libérer la session QoS en échangeant des requêtes de QoS au nom de l'application Platsim_QoS, ceci à travers une interface HTTP basée sur des communications SOAP.
Rappelons ici qu'il est possible d'utiliser un protocole de signalisation spécifique (SIP, H323, etc.) couplé à l'application, mais comme nous l'avons montré dans la section précédente, le couplage de l'application avec la signalisation engendre moins de flexibilité lors de l'établissement de la session QoS car cela nécessite plus temps pour mettre en place une requête de QoS sans pour autant permettre l'acceptation d'un nombre élevé de clients envoyant simultanément plusieurs requêtes de contrôle d'admission.

5.3.4.2 Établissement de la session de QoS pour PlatSim_QoS

La Figure 5.13 illustre les requêtes de signalisation envoyées par PlatSim_QoS arrivant au composant AQ-SSN. Ce dernier se charge de vérifier l'authentification de l'utilisateur envoyant la requête pour l'autoriser à échanger des messages pour invoquer le service auprès du serveur de QoS (RM) en utilisant le protocole Diameter [18]. Ensuite, la requête de l'utilisateur est

autorisée, et l'AQ-SSN utilise l'interface EQ-SAP pour envoyer la demande de l'utilisateur au plan de contrôle pour réserver les ressources par un nouvel appel au niveau du RM. Enfin, l'AQ-SSN demande l'autorisation au SAAA pour commencer la comptabilité associée à l'appel en cours. La signalisation de QoS est ensuite présentée au niveau réseau, c'est à dire sur le

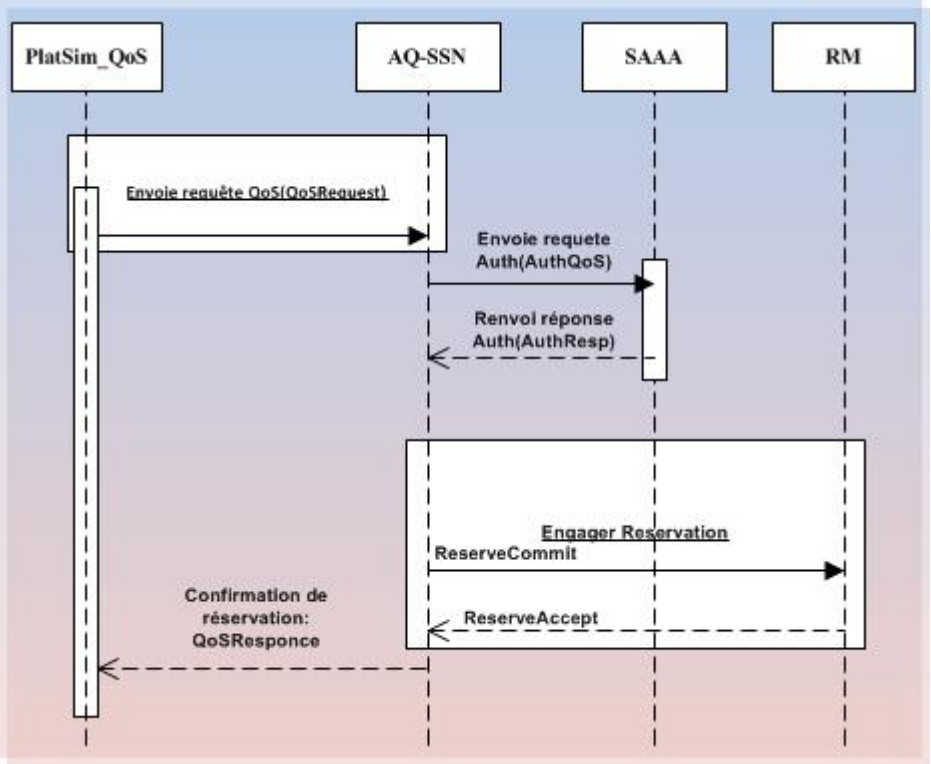


Figure 5.13: Mécanisme d'établissement de la signalisation pour PlatSim_QoS

plan de contrôle, par l'intermédiaire d'un point d'accès au service (SAP) du réseau. Le SAP permet au gestionnaire de resource (composant RM) d'établir, de mettre à jour, de modifier et de libérer les requêtes de QoS. Le RM se charge de fournir tous les mécanismes nécessaires à la signalisation découplée du chemin de données et à la mise en place de la réservation dans tous les domaines.

5.3.4.3 Provisionnement de la QoS

Dans ce qui précède, le serveur de QoS (RM) qui se trouve au niveau du plan de contrôle communique avec le plan de service par l'intermédiaire du SAP pour établir le chemin de données de bout-en-bout et gérer l'admission des nouveaux clients qui demandent la réservation des ressources. Le mécanisme de contrôle d'admission supporte un mécanisme d'allocation de ressources en intra-domaine et inter-domaine. Le serveur de QoS offre aussi la possibilité de négocier et déployer des services entre domaines hétérogènes via des APIs spécifiques.

Contrôle d'admission : Le CAC utilise les différents paramètres de QoS pour réaliser le traitement que chaque classe de trafic requiert (délai de transfert toléré, taux de perte toléré, gigue

tolérée, bande passante tolérée, etc.) en se basant sur le descripteur de trafic de l'application émettrice (qui se traduit en termes de taille de rafales, de débit crête, de taille du(des) seau(x) à jetons, etc.) pour lui affecter le traitement correspondant au service défini par son champ DSCP. Le descripteur de trafic est traduit au niveau des routeurs *Posets et Montperdu* à la bordure du réseau de la plateforme LaasNetExp en termes de classification de trafic, de gestion de files d'attentes, de lissage, de mise en conformité du trafic (traffic policing) et d'ordonnancement de paquets en tenant compte de la charge instantanée.

Provisionnement des ressources : Le processus de provisionnement consiste alors à définir le chemin de données entre différents AS pour garantir la QoS. Le modèle est basé sur le concept de chemin de bout-en-bout illustré sur la Figure 5.14, appelé EQ-Link. L'EQ-Link crée un chemin virtuel entre les routeurs de bordure des différents AS, ce chemin ayant dans chaque AS des caractéristiques de QoS connues (le chemin dans l'AS est facilement caractérisé entre deux noeuds de bordure quelconques du l'AS). La mise en place du chemin de bout-en-bout peut aussi se réaliser alors en mettant en place, entre plusieurs AS, par le protocole PCE de l'IETF, par exemple en établissant un tunnel MPLS sur DiffServ [45], tunnel spécifique à une CoS bien définie, qui peut alors s'étendre sur plusieurs AS/domaines non adjacents.

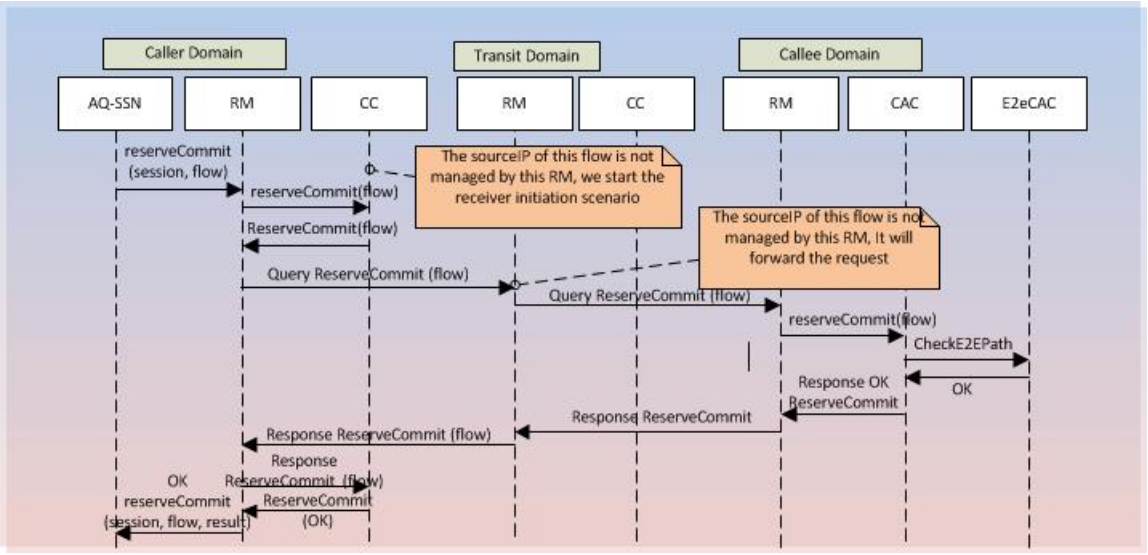


Figure 5.14: Mécanisme de provisionnement pour PlatSim_QoS

5.3.4.4 Libération de la session

Lorsque l'application demande via l'interface web service à l'AQ-SSN de libérer la signalisation et d'arrêter la réservation des ressources, elle envoie une requête de libération de QoS comme le montre la Figure 5.15. Le SAAA vérifie si la demande de l'utilisateur remplit toutes les conditions requises, et l'AQ-SSN utilise l'interface EQ-SAP pour la demande de libération des ressources. Lorsque la session se termine, le SAAA envoie les documents comptables au module charge (CHAR) pour générer les factures associées à la session.

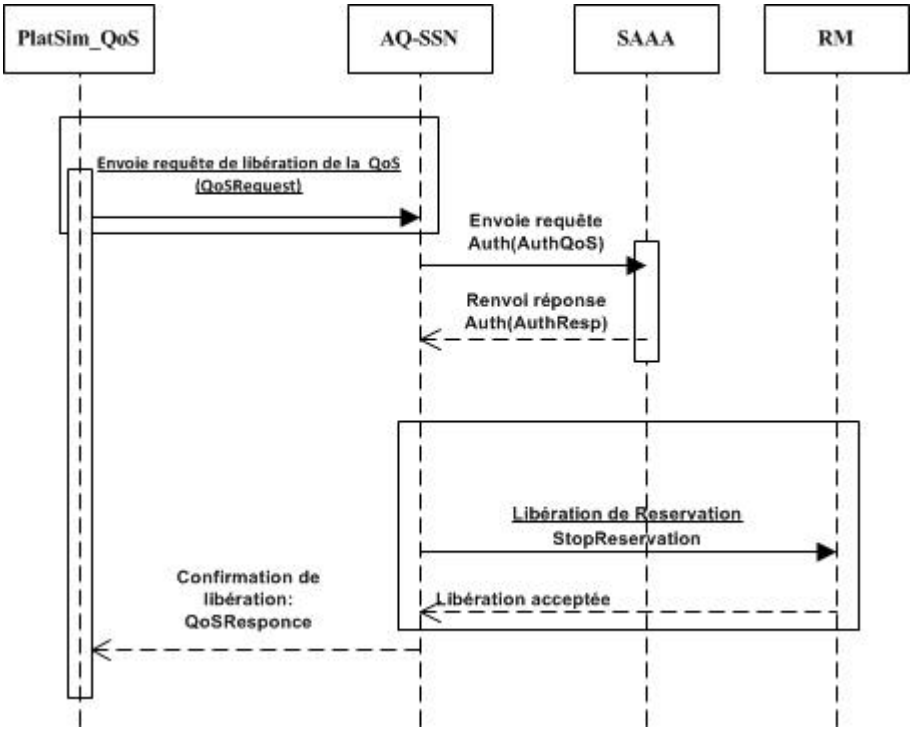


Figure 5.15: Libération d'établissement de signalisation pour PlatSim_QoS

5.3.5 Conclusion sur la QoS avec EuQoS

Dans cette partie, nous avons décrit le mapping de la QoS DDS et de la QoS réseau pour Platsim_QoS sur des réseaux hétérogènes implémentant une architecture semblable à celle d'EuQoS. En effet, la prise en compte des politiques de QoS définies par le middleware DDS n'est effective que si le réseau qui va supporter la distribution des données DDS est capable de prendre en compte cette QoS. L'utilisation du serveur de QoS Velox a permis de prendre en compte les caractéristiques de l'application DDS pour réaliser la signalisation et la réservation des ressources de bout-en-bout. Dans la prochaine Section, nous allons évaluer les performances des implémentations réalisées dans ce chapitre et nous finirons par quelques retours d'expériences.

5.4 Evaluation des solutions dans les réseaux distants sans QoS

Dans cette section, nous allons évaluer les performances des différentes implémentations des mécanismes de gestion de la simulation distribuée DDS que nous avons décrits dans ce chapitre.

5.4.1 Evaluation des performances en réseaux distants sans QoS

Pour comparer les résultats obtenus avec les différentes solutions de gestion de QoS dans les paragraphes qui vont suivre, nous présenterons successivement dans ce paragraphe, l'évaluation de performance des solutions de PlatSim_QoS que nous avons réalisée sur un réseau WAN

n'implémentant aucun mécanisme de gestion de QoS, avec trois architectures différentes : (i) la solution initiale de PlatSim, (ii) PlatSim_QoS avec HLA et (iii) PlatSim_QoS avec DDS.

Condition des expérimentations : Pour ces expérimentations, nous avons utilisé les implémentations de Platsim_QoS que nous avons présentées dans le chapitre 3. Pour l'implémentation sous HLA, le flux généré par chaque simulateur est composé de 10 classes d'objets et de 7 classes d'interactions de HLA, générant ensemble un flux à un débit moyen de 500kbps.
Concernant l'implémentation sous DDS, l'application est composée des 17 Topics que nous avons décrits dans le paragraphe 3.4, et générant aussi un débit moyen de 500kbps.
Pour la solution initiale de Platsim sans QoS, il s'agit d'une architecture client/serveur sans aucun middleware. Chaque simulateur génère le trafic global de Platsim sans aucune décomposition des flux à un débit constant de 554kbps. Pour étudier l'impact de la gestion des files

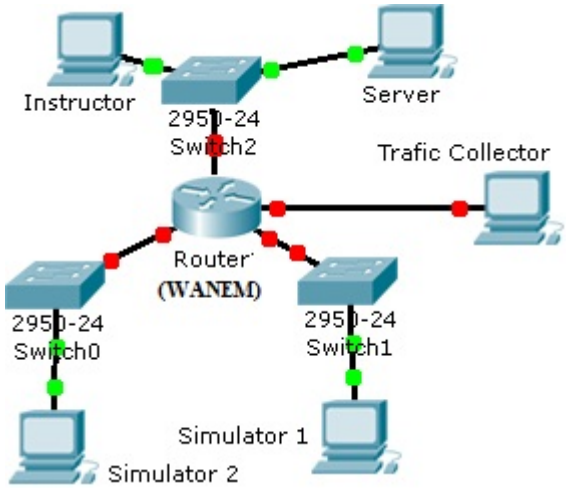
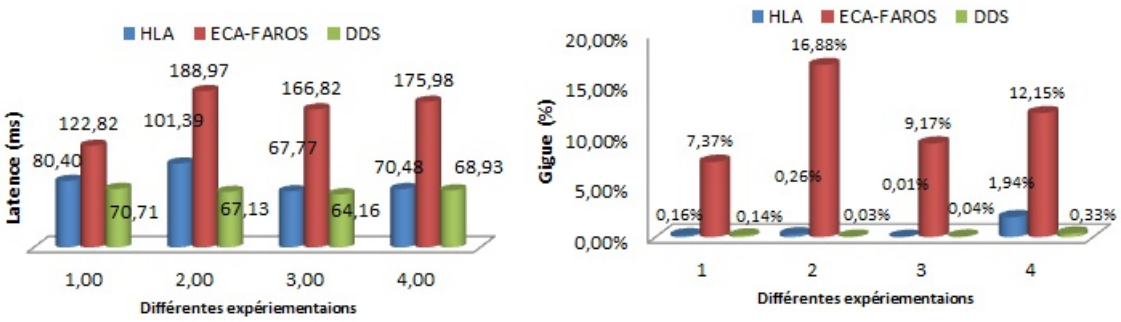


Figure 5.16: Architecture de Test avec l'émulateur WANem

d'attentes sur la voie aller des routeurs reliant deux simulateurs situés sur deux réseaux distants sans QoS, nous avons utilisé l'émulateur WANem [1] pour émuler un réseau WAN (Figure 5.16). WANem [105] permet d'installer un routeur Linux pour simuler des qualités de liens spécifiques comme la latence, la bande passante, le taux de perte de paquets, la gigue, etc. Il implémente l'outil TC (Traffic Control) fonctionnant sous Linux. Cet outil utilise aussi un gestionnaire de file d'attente par flux réseau, un choix des règles de classification des paquets avant leur mise en file d'attente. Dans ces expérimentations aucun mécanisme de gestion de QoS n'est configuré.
Cet émulateur est configuré pour avoir une capacité totale de 1Mbps dont 500kbps sont alloués pour les flux envoyés par chaque simulateur, un délai de transfert de 60ms et un taux de perte de paquets de 10%, sachant que nous tenons compte du débit moyen envoyé par chaque simulateur.
Pour ces tests, nous démarrons les expérimentations pour chaque architecture indépendamment l'une de l'autre, et nous répétons la manipulation plusieurs fois pour s'assurer que les résultats

1. `http://wanem.sourceforge.net/`

obtenues sont constants. Nous utilisons aussi le générateur de trafic UDP concurrent "Jperf" [1] à un débit de 1Mbps afin d'émuler ce qui pourrait être un réseau Internet surchargé sans QoS.



(a) Evaluation de la Latence (ms) **(b)** Evaluation de la Gigue (%)

Figure 5.17: Performance sur WAN sans QoS pour les implémentations avec HLA, DDS et PlatSim ECA

Analyse : Pour analyser les résultats des expérimentations, nous présentons les graphiques de la Figure 5.17 correspondant au délai moyen (Moving average Delay) de transfert du trafic des trois solutions et la gigue correspondant à la variation de délai de transmission entre l'émetteur et le récepteur.

L'inspection de la Figure 5.17a et la Figure 5.17b montre que la solution Platsim sur une architecture client/serveur présente les valeurs de latence les plus longues pendant les expérimentations. Par exemple, l'interconnexion des simulateurs de Platsim sur une architecture client/serveur montre une latence entre 122,82ms, l'implémentation sur HLA montre un délai de 80,40ms et pour l'implémentation sur DDS la latence est égale à 70,71ms. Concernant la gigue, l'architecture client/serveur admet une gigue de 7,37%, supérieure aux deux autres solutions ; la gigue pour l'architecture sur HLA qui est 0,16% et celle sur DDS est égale à 0,14%. Ceci parce que le serveur de simulation doit attendre la réception des paquets depuis tous les simulateurs pour régénérer le trafic correspondant. En plus, la solution initiale utilise un environnement réel qui consomme beaucoup de temps CPU, ce qui influence le délai de transfert .

La deuxième constatation que nous avons pu retenir des graphiques c'est que, en implémentant des mécanismes de synchronisation de flux sur la solution HLA pour combler le manque de mécanismes de gestion de la QoS de HLA, ces résultats sont très proches pour HLA et DDS, et que la différence apparaît très étroite lors de l'évaluation de leur performance. En plus, l'utilisation du middleware HLA & DDS permet une réduction de bande passante pour les flux applicatifs décomposés par rapport à la solution initiale de ECA-FROS qui se contente d'envoyer les données applicatives ensemble sans les décomposer en éléments applicatifs. Cependant, DDS présente toujours une meilleure performance. Ces résultats justifient encore une fois notre choix d'implémenter PlatSim_QoS sur un middleware DDS plutôt que sur HLA.

1. https ://www.projet-plume.org/fr/fiche/iperf

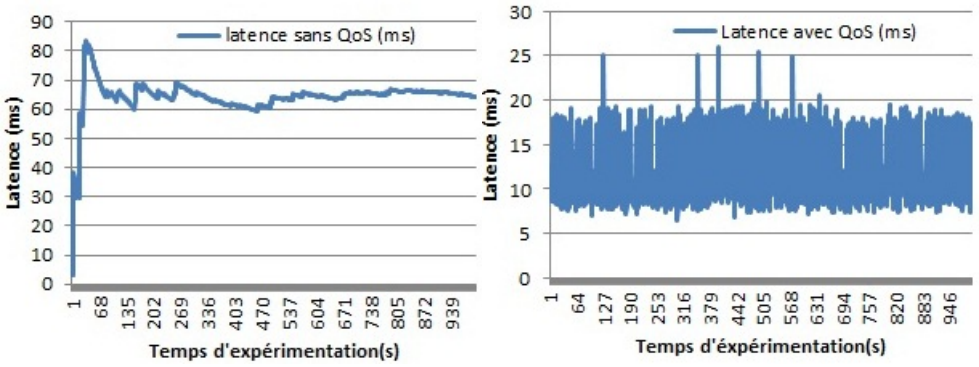
5.4.2 Evaluation de l'architecture couplant SIP et DDS

Dans cette partie, nous allons présenter les expérimentations qui nous ont permis de valider le fonctionnement de l'architecture présentée dans le paragraphe 5.2 permettant la gestion de la QoS pour la solution Platsim_QoS sur DDS.

Conditions des expérimentations : Les expériences ont aussi été réalisées sur la plateforme en déployant un routeur Linux basé sur l'outil TC. Nous considérons une session SIP/DDS (données de Platsim_QoS avec 4 classes de Topics). Pour vérifier que le traitement du flux DDS est prioritaire, nous l'avons soumis à un flux UDP concurrent à un débit moyen de 1Mbps pour perturber son comportement.
Pour les tests sans QoS nous considérons une seule file d'attente pour le service Best Effort pour forcer tous les flux à passer par cette file. Pour ce qui concerne les tests avec QoS, nous favorisons le traitement prioritaire du trafic DDS au niveau du routeur et nous n'associons aucune priorité au trafic UDP concurrent. Les courbes obtenues sont issues de l'analyse de fichiers traces qui ont été obtenues lors des mesures de performances.
Si nous reprenons l'architecture de LaasNetExp, nous trouvons les éléments suivants :

- une communication SIP entre deux clients SIP/DDS, chacun situé sur une machine. Chaque Client SIP/DDS est formé par un seul DataWriter et un seul DataReader qui gèrent tous les Topics du client.
- Les QoS policies de DDS configurées dans le fichier XML sont les suivantes : pour le "Transport_Priority" la valeur est 46, ce qui donne une valeur 46 du champ DSCP pour un service EF de DiffServ. La fiabilité au niveau DDS est choisie "DDS_BEST_EFFORT_RELIABILITY_QOS" pour le DataWriter et le DataReader. Les valeurs des paramètres "Latency_Budget" et "Deadline" sont $0s, 15000000ns$ pour spécifier le temps de transfert admissible à 15ms.
- Un proxy SIP conscient de la QoS DDS est présent sur chaque machine à côté du client SIP/DDS.
- Un serveur de QoS basé sur COPS-DRA se situe au niveau de chaque machine à côté du Proxy SIP.

Analyse : Pour étudier l'impact des mécanismes de la QoS SIP sur le délai moyen de transfert des données, nous allons analyser les graphiques de la Figure 5.18 correspondant au délai moyen (Moving average Delay) de transmission entre le Publisher et le Subscriber.
Nous avons constaté aussi que le délai moyen ressenti par le mécanisme de gestion de la QoS avec SIP (voir Figure 5.21b) est inférieur au délai sans QoS (Figure 5.21a). En effet, lorsque la QoS n'est pas mise en place, le trafic DDS passe dans la même file d'attente que le flux UDP concurrent. Puisque le réseau interconnectant les deux participants DDS ne supporte pas la QoS réseau, la spécification des QoS policies de DDS sur chaque noeud n'offre pas de garanties pour traiter le flux DDS en priorité par rapport aux autres flux concurrent au niveau du réseau. L'ajout des mécanismes de QoS réseau améliore la latence de bout-en-bout pour le trafic DDS et dégrade aussi le flux UDP concurrent. En plus, le délai moyen du flux DDS avec QoS est aux alentours des 15ms spécifiés dans le message SDP envoyé par le publisher vers le proxy SIP amélioré. Ceci nous permet de confirmer que les QoS policies de DDS permettent l'amélioration de la communication sur les machines locales. Ils permettent d'améliorer la distribution des données de Platsim_QoS sur DDS entre le publisher et le subscriber.



(a) Latence SIP/DDS sans QoS **(b)** Latence SIP/DDS avec QoS

Figure 5.18: Impact de la gestion de QoS par SIP/DDS sur le délai moyen de transfert

Cependant, la mise en place de la session DDS ne peut se faire que si le proxy SIP délivre le message "200Ok" vers le publisher indiquant que l'allocation de ressources est réalisée, ce qui implique qu'un ensemble de message de négociation de la session a été échangé entre le publisher et le subscriber. Le délai cumulatif obtenu par les messages de négociation de la session illustre l'impact potentiel de l'addition des mécanismes de contrôle QoS SIP dans l'établissement de session. Le processus de négociation a l'inconvénient d'augmenter le délai total de bout-en-bout, qui en plus du délai de transfert ajoute le délai d'établissement de session, et ajouté au temps de mise en place de la QoS, le délai total atteint rapidement des valeurs proches de 300ms. Ceci nous amènera dans le prochain paragraphe à évaluer les mécanismes de gestion de QoS avec l'approche EuQoS.

5.4.3 Evaluation des performances de PlatSim_QoS à travers un réseau multi-domaine à QoS

Dans cette partie, nous allons évaluer les performances de la solution basée sur l'architecture de type EuQoS. Nous considérons le test expérimental effectué pour la gestion de la QoS du middleware DDS à travers un réseau multi-domaine à QoS. La gestion de la QoS est réalisée à l'aide du serveur de QoS Velox.
De la même manière que dans les deux cas précédents, nous allons étudier l'impact de la gestion des files d'attentes sur le délai de transfert entre les routeurs de la plateforme et nous nous plaçons dans les mêmes conditions de test que celles nous avons données dans les expérimentations précédentes. Pour cela, nous commençons par la description de la configuration du serveur Velox ; ensuite nous étudions l'impact de cette configuration sur le délai de transfert, la bande passante et aussi sur l'augmentation du nombre de simulateurs.

5.4.3.1 Configuration du gestionnaire de ressources

La mise en place de la stratégie de gestion de la QoS sur un réseau distant hétérogène nécessite l'intervention de l'administrateur de chaque domaine pour définir le politique de QoS à mettre en place en tenant compte du trafic échangé par son domaine. Pour cela, notre stratégie

nécessite la définition de la topologie des routeurs de bordure (définissions des liens d'interconnexions, les noms les adresses IP, les ports de communications avec le serveur Velox, etc.) et la définition des droits d'accès de l'utilisateur pour vérifier son authentification afin de lui accorder l'autorisation de créer la session QoS comme nous l'avons décrit dans le paragraphe 5.3.4. Cette configuration est montrée à la Figure 5.19 : les routeurs de bordure sont Posets avec l'adresse réseau $X.Y.Z.61/26$ et Montperdu avec l'adresse réseau $X1.Y1.Z1.80/28$. Les machines Euqos6 et Euqos7 échangeant les données sont déclarées dans le gestionnaire de ressources réseau pour qu'il puisse créer la session de bout-en-bout. La deuxième étape consiste à ajouter un service

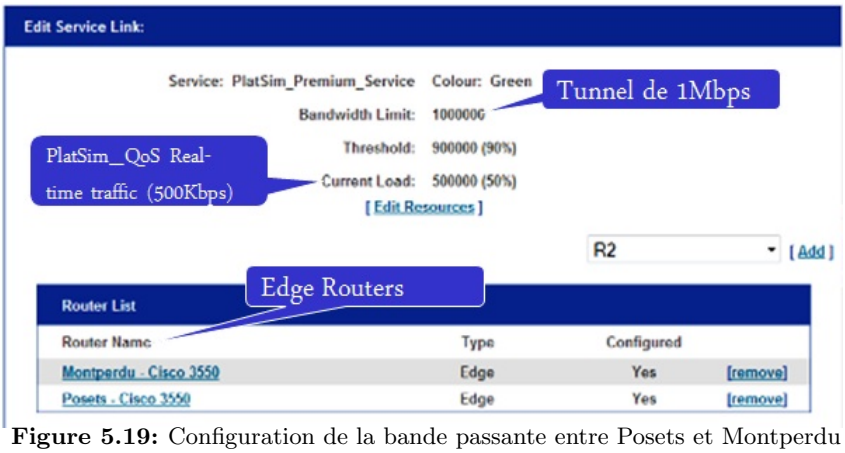


Figure 5.19: Configuration de la bande passante entre Posets et Montperdu

réseau (Network service) qui sera utilisé par l'EQ-SAP. Ce dernier va traduire cette session vers le gestionnaire des ressources réseau (RM) qui se chargera de réserver les ressources requises à l'application DDS demandant la QoS. Le RM va alors utiliser plusieurs scripts (rédigés en langage CLI) pour réaliser les taches d'administration des routeurs. La configuration de Velox nécessite les quatre fichiers suivant :

- Fichier Configuration : Script permettant d'appliquer la politique de QoS sur le routeur.
- Fichier Cleanup : Script permettant de réinitialiser la configuration du routeur et le remettre à défaut.
- Fichier Start : Script permettant de déclencher une réservation de bande passante suite à une requête du service Trigger.
- Fichier Stop : Script permettant de libérer la bande passante réservé suite à un arrêt du service Trigger

Une fois la configuration est réalisée, le chargement de la politique de QoS dans le serveur Velox retourne une notification de la réussite de l'opération comme le montre la Figure 5.20.

5.4.3.2 Impact des mécanismes de gestion de la QoS sur la bande passante

Dans cette configuration, les flux concurrents sont un flux UDP issu d'Iperf pour congestionner les files d'attente des routeurs et perturber le comportement du flux DDS qui a pour marquage DSCP 46, correspondant au service EF de DiffServ. Notre objectif est d'étudier l'impact de la signalisation générique réalisée par Velox sur la bande passante. Nous proposons alors d'injecter un flux DDS en concurrence avec le flux UDP initial avant de mettre en place la QoS et refaire cette expérience sur le réseau à QoS.

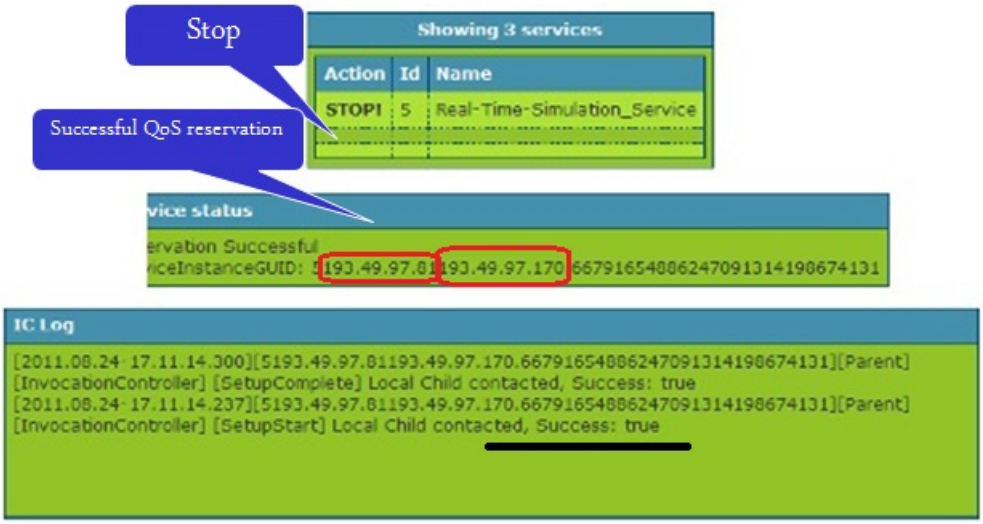
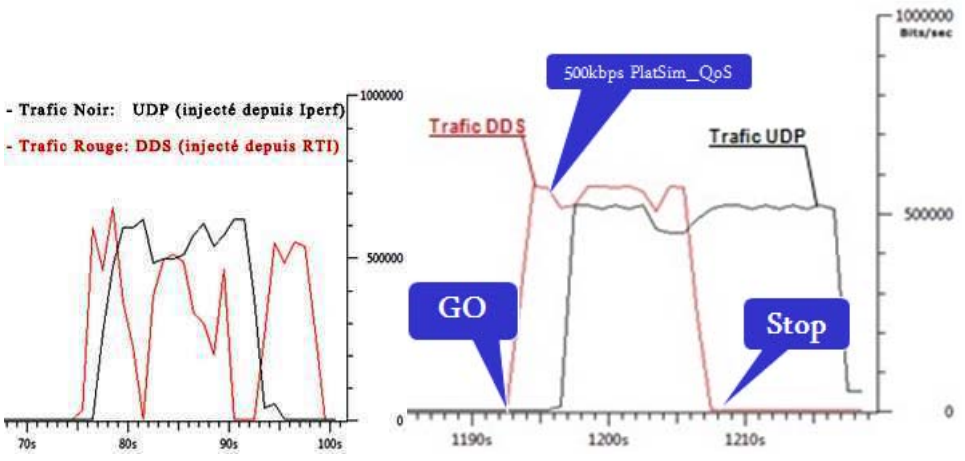


Figure 5.20: Résultat de l'invocation du service Trigger du gestionnaire de ressources

Dans le premier scénario, nous considérons la configuration des routeurs de la plateforme sans QoS (par défaut les routeurs utilisent deux files uniquement : 95% Best-effort et 5% Network-Control) où tous les flux qui traversent le réseau passent par une seule file d'attente BE. Pour la première expérience, nous démarrons à t=0s un flux DDS à 500Kbps suivie d'un flux UDP à 600Kbps injecté par Iperf pour congestionner la file d'attente et observer le comportement du flux DDS.

Analyse : Pour l'évaluation de cette configuration, nous allons analyser les graphiques de la Figure 5.22 correspondant à la bande passante du flux PlatSim_QoS sans QoS envoyé depuis EuQoS7 vers EuQoS6 (courbe 5.21a : tous les flux sont en BE) et avec QoS (courbe 5.21b, le flux DDS est en EF et les autres flux sont BE).



(a) Sans QoS **(b)** Avec QoS

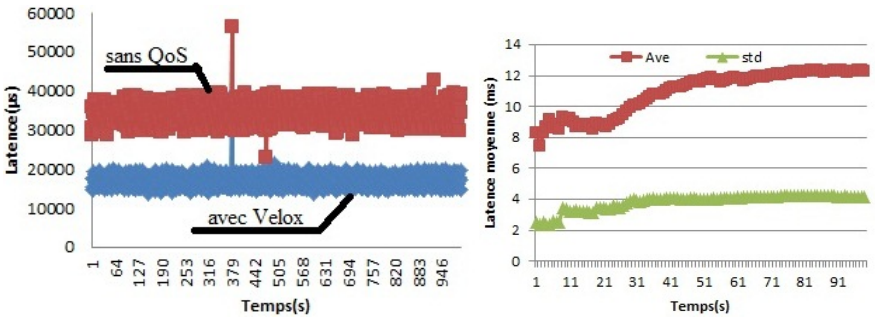
Figure 5.21: Impact de l'utilisation du Ressource Manager sur un flux DDS

D'après la Figure 5.21a, nous constatons une dégradation du comportement du flux DDS, qui n'arrive pas à maintenir une bande passante constante dût à une perturbation par le flux UDP concurrent.
Le deuxième scénario, consiste alors à mettre en place une signalisation générique de bout-en-bout en utilisant le gestionnaire de bande passante Velox et appliquer la stratégie de gestion de la QoS définie précédemment. Pour cela, nous mettons le flux UDP perturbateur en BE à 600kbps en concurrence avec un flux PlatSim_QoS en EF à 500kbps.
L'inspection de la Figure 5.21b nous permet de constater que le flux PlatSim_QoS présente une bande passante constante par rapport au premier scénario malgré l'existence du flux de perturbation UDP mis en concurrence avec le premier flux. Lorsque la QoS est activée, à l'initiation de la session, le serveur Velox a envoyé un message de réservation pour le flux DDS pour configurer les files d'attentes. Les paquets du flux DDS passent alors par la file EF prioritaire par rapport à celle d'UDP BE. Il en découle que les flux sont protégés et leurs délais moyens n'augmentent que très faiblement (on le verra dans le paragraphe suivant).

5.4.3.3 Impact des mécanismes de gestion de la QoS sur le délai moyen

Dans ce scénario, nous utilisons la configuration précédente pour évaluer l'effet de la mise en place de la QoS sur le délai moyen de transfert en fonction du nombre de simulateurs déployés sur deux réseaux d'accès distants. Le premier cas comporte un seul publisher Platsim_QoS situé dans le domaine IP1 envoyant du trafic vers un subscriber Platsim_QoS sur le domaine IP2.

Analyse : Pour cela, nous allons analyser les graphiques de la Figure 5.22 correspondant à la latence de transfert du flux PlatSim_QoS envoyé depuis EuQoS7 vers EuQoS6 (courbe 5.22a, tous les flux sont en EF) et la distribution de la variabilité du délai (écart-type ou STD sur la courbe 5.22b, le flux DDS est en EF et les autres flux sont BE). D'après la Figure 5.21a, nous



(a) délai de transfert **(b)** variabilité du délai

Figure 5.22: Impact de l'utilisation du Ressource Manager sur le délai de transfert de bout-en-bout

constatons que le délai moyen de transfert présente des résultats meilleurs que ceux trouvés sans QoS et même un peu meilleurs que les délais obtenus avec la gestion de la QoS en utilisant SIP. Ceci s'explique par le fait que la signalisation générique réalisée par Velox n'ajoute pas de délai supplémentaire pour la mise en place de la QoS. Une fois le chemin de bout-en-bout établi, Velox se charge de réserver les ressources à l'application Platsim_QoS sur les routeurs Posets et Montperdu.

5.4.3.4 Impact du nombre de simulateurs sur le délai moyen

Dans cette expérimentation, nous avons considérés les mêmes conditions d'expérimentations sur la même plateforme, sauf que les flux envoyés sont issus de plusieurs applications Platsim_QoS/DDS qui distribuent les Topics entre EuQoS6 dans le réseau IP 1 de LaasNetExp ayant "Montperdu" comme routeur de sortie (publisher) et EuQoS7 dans le réseau IP 2 qui a "Posets" comme routeur de sortie (Subscribers) utilisant le service EF de DiffServ. En plus, nous avons utilisé une bande passante entre les producteurs et les consommateurs de 100Mbps, et nous avons injecté un flux UDP issu d'Iperf concurrent aux flux DDS.
La Figure 5.23a illustre la latence moyenne entre un publisher qui envoie du trafic DDS vers plusieurs subscribers. La deuxième configuration concerne plusieurs publishers qui distribuent leur trafic DDS vers un seul subscriber (Figure 5.23b). La Figure 5.23c concerne la communication entre plusieurs publishers et plusieurs subscribers. Enfin, la Figure 5.23d compare la latence moyenne observée lors de la distribution des Topics DDS de : i) 1 publisher vers 8 subscribers ; ii) 8 publishers vers 1 subscriber et iii) 8 publishers vers 8 subscribers.

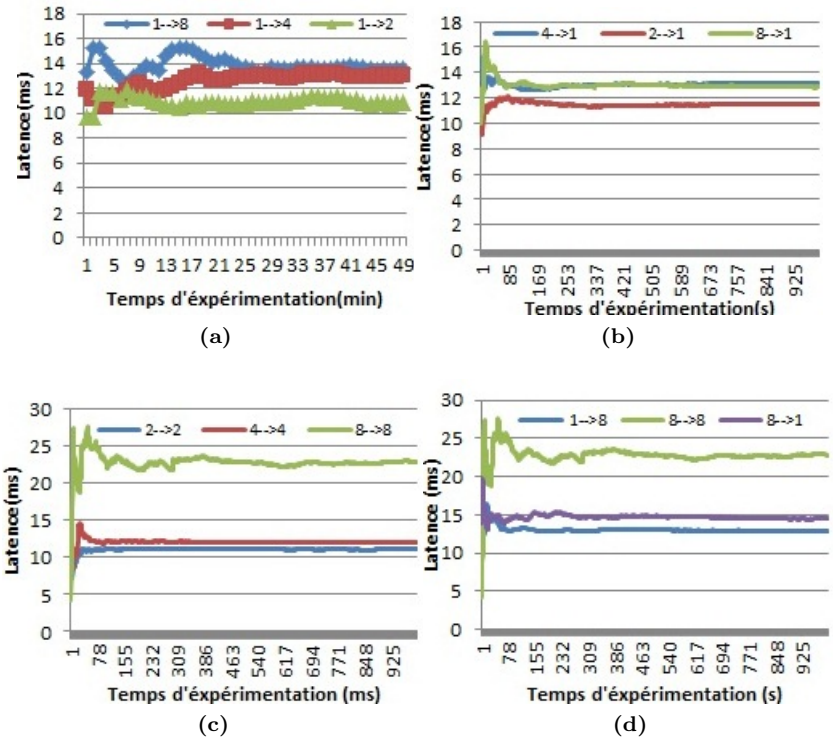
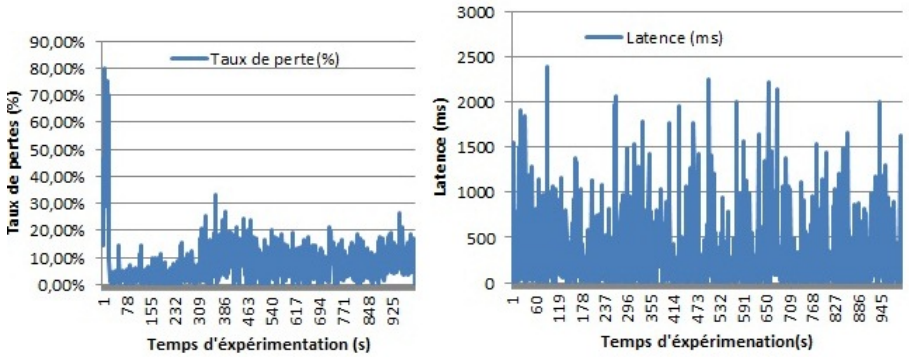


Figure 5.23: Impact de l'augmentation du nombre de simulateurs sur le délai de transfert de bout-en-bout

Analyse : L'inspection de la Figure 5.23 montre que le délai moyen de transfert pour la configuration "one-to-many" et "many-to-one" est entre 12ms et 15ms. Le délai pour la configuration "many-to-many" augmente considérablement pour atteindre 22ms lorsque nous avons utilisé 16 participants. Ceci s'explique par le fait que chaque subscriber doit traiter toutes les données qui

lui sont envoyées par tous les publishers, et cela augmente le temps de traitement sur chaque machine. Ces données sont stockées dans le cache du DataReader du subscriber dans l'ordre de leur arrivée. Le DataReader dispose d'un seul "DDS_Listener" qui doit récupérer tous les Topics DDS reçus à partir tous les publishers depuis son cache pour les acheminer à l'application qui va afficher ces Topics.
Concernant le flux UDP concurrent, nous présentons dans la Figure 5.24 l'impact de la priorisation du flux DDS par rapport au flux UDP. En effet, le délai ressenti pour le flux UDP concurrent augmente considérablement à 500ms lorsque le nombre de publishers augmente et nous remarquons que le taux de perte des paquets atteint 20%.



(a) Taux de perte flux UDP Iperf **(b)** Latence flux UDP Iperf

Figure 5.24: Latence et taux de perte des flux UDP concurrents

5.5 Conclusion

Dans ce chapitre, nous avons présenté deux contributions pour la gestion de la QoS dans les réseaux étendus. Pour cela, la première partie a été consacrée au couplage de la signalisation SIP avec le middleware DDS. Nous avons montré l'utilité de la mise en place d'un Framework pour l'automatisation de la réservation de ressources en se basant sur les mécanismes de négociation qu'offre le protocole de gestion SIP. Nous avons décrit les différentes étapes de la signalisation : a) l'établissement des sessions entre les proxies SIP que nous avons améliorés pour la prise en compte de la QoS DDS ; b) la conception du serveur COPS amélioré pour qu'il puisse supporter les spécifications DDS.
Par la suite, nous avons introduit les mécanismes de négociation entre ce dernier et le routeur de bordure de son domaine IP.
Toutefois, SIP montre un certain nombre de faiblesses. Il est à évaluer pour aborder des applications qui nécessitent de supporter la gestion de la QoS pour des milliers de Topics. La signalisation SIP avec COPS pour DDS engendre un couplage entre le plan de service et le plan de contrôle qui est encore à étudier.
Dans la deuxième partie de ce chapitre, nous avons présenté le projet européen IST EuQoS auquel le groupe OLC a participé. La contribution a ici porté sur des modifications permettant de développer et d'intégrer son architecture dans le cadre du projet PlatSim. L'approche préconisée par EuQoS est innovante en matière de concepts et de fonctionnalités à utiliser dans

les réseaux et l'internet nouvelle génération. Nous nous sommes alors investi autant dans la ré-implémentation de ses briques logicielles pour réutiliser leurs mécanismes et fonctionnalités que dans leur réintégration, déploiement et test dans le cadre du projet PlatSim.
Les résultats des expérimentations confirment que la configuration du réseau peut être automatisée ou simplement administrée par l'utilisateur ou l'administrateur. Nous soulignons enfin que la garantie des politiques de QoS du middleware DDS et donc des applications n'est effective que si le réseau qui va implémenter la distribution des données est capable de prendre en compte et d'implémenter une vrai garantie de qualité de service.

Chapitre 6

Conclusion et Perspectives

La problématique adressée dans nos travaux a concernée la mise en oeuvre d'architectures permettant l'intégration de simulateurs distribués en réseaux locaux et à grandes distances avec des mécanismes de qualité de service dans la cadre du projet national Platsim. Ces simulateurs permettaient alors l'entraînement d'équipes dans des situations critiques et à risque.

Bilan :

Les contributions apportées par nos travaux ont été présentés en plusieurs étapes :

1. Nous avons proposé une recomposition des différents flux de Platsim en les décomposant en plusieurs classes de "sous-flux". Chaque classe a des caractéristiques propres qui permettent d'exprimer le profil trafic applicatif et les exigences applicatives. Ces exigences nous ont permis d'étudier les contraintes de synchronisation des flux sur HLA, en utilisant un modèle formel basé sur les réseaux de Petri à flux hiérarchique, que nous avons implémenté au sein de l'application afin de combler le manque de QoS applicative de HLA.
   Ensuite, nous avons implémenté une architecture de simulation distribuée sur DDS en profitant de sa richesse en termes de paramètres de QoS. Cette étape nous a permis d'assurer une simulation synchronisée entre les différents participants en leurs assurant la QoS requise dans le cadre d'un réseau local. Les évaluations de performances nous ont permis de choisir DDS comme solution pour les prochaines générations des simulateurs de Platsim_QoS.
2. Nous avons proposé par la suite une approche de navigation à l'estime (Dead Reckoning) pour l'anticipation de position des objets simulés. Cette solution permet, à des applications distribuées dont le support réseau n'offre pas de mécanismes permettant de garantir la QoS dédiée, des performances suffisantes aux différents simulateurs et équipements pour permettre le déroulement quasi-normal de l'exercice. Nous avons également présenté deux solutions pour l'interconnexion de simulateurs DDS en réseaux distants sans QoS : (1) l'utilisation du service de routage fournit par DDS pour élaborer un "pond-fédéré" et (2) le proxy DDS. Ces solutions permettaient l'optimisation de l'échange de flux transmis à travers un réseau à grande envergure.
3. Enfin, nous avons proposé de gérer la QoS dans un environnement grande distance à QoS garantie avec deux approches distinctes : la première est une extension SIP/SDP pour l'intégration de la QoS DDS. La deuxième est basée sur l'architecture du projet EuQoS qui inspire ce que pourrait être l'internet de nouvelle génération. Nous avons pu

évaluer l'efficacité des mécanismes de QoS que nous avons proposés en concluant que le traitement différencié du trafic depuis l'application jusqu'au réseau permet une utilisation plus efficace des ressources et des équipements de la simulation.

Perspectives :

Les différents travaux réalisés ouvrent la voie à diverses perspectives de recherche liées à l'extension de ces contributions. Ces perspectives se situent sur le court terme et sont liées directement à ces travaux d'une part. L'utilisation de réseaux sémantiques à base d'ontologies nous semble une possibilité d'extension du pouvoir d'expression des mécanismes de spécification de la QoS et des besoins applicatifs. A moyen et à long terme, elle concerne l'intégration et l'adaptation de nos travaux à d'autres architectures.

Les perspectives à court termes sont les suivants :

- Certaines applications basées sur HLA devront communiquer avec des applications DDS sans modifier leur code. Pour l'amélioration de l'interopérabilité des travaux actuels, il sera donc intéressant de proposer une passerelle entre HLA et DDS, pour profiter de la spécificité de chaque architecture et la réutilisation du code des anciennes fédérations.
- Ensuite, la prise en compte des QoS policies de DDS sont spécifiées une seule fois dans des profils de QoS différents. Afin d'automatiser cette opération, nous avons actuellement commencé à travailler sur l'utilisation des ontologies pour l'auto-configuration des QoS DDS en fonction de la charge du réseau, par exemples pour adapter dynamiquement le débit de transmission. L'intérêt de cette approche est qu'elle pourrait permettre la reconfiguration automatique du commutateur LAN.

A long terme, les principales perspectives envisagées concernent :

- l'étude de l'intégration de la QoS DDS aux réseaux sans fil. Certains travaux ont été faits pour les réseaux de capteurs sans fil, en intégrant les QoS DDS au niveau applicatif. Ces approches ne prennent pas en considération les propriétés intrinsèques de ces systèmes, notamment au niveau de l'optimisation de la consommation de l'énergie permettant d'aller vers des systèmes écologiques. Ces directions pourront créer un axe de recherche complet pour l'intégration de DDS dans des architectures multicouches (Cross-Layer) à faible consommation d'énergie.
- Sur la base de ce que nous venons de développer, il est intéressant de penser augmenter l'interaction entre les différentes couches de l'architecture (de la couche application vers la couche physique en passant par le middleware) en associant les informations des couches hautes avec celles des couches basses (ex. réseaux sans fil, Cloud Computing, etc.). Il serait donc envisageable d'intégrer de nouveau services de transport au middleware DDS à côté des services de base. Cette approche nécessite l'intégration de la QoS au niveau transport pour permettre une bonne gestion de la communication avec des protocoles à fiabilité partielle et ordre partiel comme SCTP et DCCP.

References

[1] M. Aaron, W. Tomas, and M. Seamus. Multistep-ahead neural-network predictors for network traffic reduction in distributed interactive applications. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, Septembre 2007. 91

[2] J. H. Ahn and T. G. Kim. Design and implementation of hierarchical rti (hrti). *Spring Simulation Interoperability Workshop*, June 1999. 14

[3] G. Almes, S. Kalidindi, and M. Zekauskas. A One-way Delay Metric for IPPM. RFC 2679 (Proposed Standard), Sept. 1999. 27

[4] G. Almes, S. Kalidindi, and M. Zekauskas. A One-way Packet Loss Metric for IPPM. RFC 2680 (Proposed Standard), Sept. 1999. 28

[5] P. Amsden, J. Amweg, P. Calato, S. Bensley, and G. Lyons. Cabletron's Light-weight Flow Admission Protocol Specification Version 1.0. RFC 2124 (Informational), Mar. 1997. 38

[6] D. Andrea and G. Daniele. Using corba to enhance hla interoperability in distributed and web-based simulation. *Lecture notes in computer science*, 2004. 14

[7] J. Aronson. Dead reckoning : Latency hiding for networked games. September 1997. 91

[8] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. RFC 2702 (Informational), Sept. 1999. 31

[9] J. Babiarz, K. Chan, and F. Baker. Configuration Guidelines for DiffServ Service Classes. RFC 4594 (Informational), Aug. 2006. Updated by RFC 5865. 69

[10] M. Bassiouni and J. W. M. Chiu. Performance results of atm backbones for dis networks. *Proceedings of Southeastern Simulation Conference.* 7, 15

[11] T. Bates and R. Chandra. BGP Route Reflection An alternative to full mesh IBGP. RFC 1966 (Experimental), June 1996. Obsoleted by RFC 4456, updated by RFC 2796. 30

[12] B. Bouchon-Meunier. *La logique floue et ses applications.* Addison-Wesley France Eds, janvier 1996. 92

[13] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture : an Overview. RFC 1633 (Informational), June 1994. 32

[14] R. Braden and L. Zhang. Resource ReSerVation Protocol (RSVP) – Version 1 Message Processing Rules. RFC 2209 (Informational), Sept. 1997. 39

[15] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC 2205 (Proposed Standard), Sept. 1997. Updated by RFCs 2750, 3936, 4495, 5946. 39

[16] T. Braun, M. Diaz, J. E. Gabeiras, and T. Staub. *End-to-End Quality of Service Over Heterogeneous Networks*. springer Co, 2008. 43

[17] B. Bréholée and P. Siron. Design and implementation of a hla interfederation bridge. *European Simulation Interoperability Workshop*, June 2003. 14

[18] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko. Diameter Base Protocol. RFC 3588 (Proposed Standard), Sept. 2003. Updated by RFCs 5729, 5719. 134

[19] M. Callejo-Rodrigitez, J. Enriquez-Gabeiras, W. Burakowski, A. Beben, J. Sliwinski, O. Dugeon, E. Mingozzi, G. Stea, M. Diaz, and L. Baresse. Euqos : End-to-end qos over heterogeneous networks. In *Innovations in NGN : Future Network and Services, 2008. K-INGN 2008. First ITU-T Kaleidoscope Academic Conference*, may 2008. 126

[20] M. Callejo-rodriguez and J. Enriquez-Gabeiras. Bridging the standardization gap to provide qos in current ngn architectures. *Communications Magazine, IEEE*, october 2008. 126

[21] R. Callon. Use of OSI IS-IS for routing in TCP/IP and dual environments. RFC 1195 (Proposed Standard), Dec. 1990. Updated by RFCs 1349, 5302, 5304. 30

[22] G. Camarillo, B. Marshall, and J. Rosenberg. Integration of resource management and sip. *IETF Internet Draft, draft-ietf-sip-manyfolks-resource-07.txt*, April 2002. 41

[23] G. Camarillo, W. Marshall, and J. Rosenberg. Integration of Resource Management and Session Initiation Protocol (SIP). RFC 3312 (Proposed Standard), Oct. 2002. Updated by RFCs 4032, 5027. 117, 119

[24] K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar, and A. Smith. COPS Usage for Policy Provisioning (COPS-PR). RFC 3084 (Proposed Standard), Mar. 2001. 40

[25] C. Chassot, A. Lozes, M. Diaz, L. Dairaine, and L. Rojas. Qos requise par une application de dis distribuée dans un environnement réseau grande distance. *Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'99) Nancy*. 7

[26] D. Chen, B.-S. Lee, W. Cai, and S. J. Turner. Design and development of a cluster gateway for cluster-based hla distributed virtual simulation environments. *ANSS '03 Proceedings of the 36th annual symposium on Simulation*, 2003. 14

[27] L. Chen and G. Chen. A fuzzy dead reckoning algorithm for distributed interactive applications. *In Proceedings of the 2nd International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*. 91

[28] D. Chirieleison, L. Cunningham, D. Scott, and S. Tarquinio. Dis communication service interface to atm. *MITRE Corporation*. 7

[29] B. Christos and P. Dimitris. Architectures and performance evaluation of bandwidth brokers. *Int. J. Netw. Manag.*, 2009. 36

[30] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QoS-based Routing in the Internet. RFC 2386 (Informational), Aug. 1998. 30

[31] P. K. Davis. Distributed interactive simulation in the evolution of dod warfare modeling and simulation. *Proceedings of the IEEE*, August 1995. 6

[32] S. Deering. Host extensions for IP multicasting. RFC 1112 (Standard), Aug. 1989. Updated by RFC 2236. 101

[33] C. Demichelis and P. Chimento. IP Packet Delay Variation Metric for IP Performance Metrics (IPPM). RFC 3393 (Proposed Standard), Nov. 2002. 28

[34] M. Diaz. *Petri Nets : Fundamental*. Wiley and ISTE, 2009. 52

[35] M. Diaz and P. Senac. Time stream petri nets : A model for timed multimedia information. In *Proceedings of the 15th International Conference on Application and Theory of Petri Nets*. Springer-Verlag, 1994. 53

[36] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry. The COPS (Common Open Policy Service) Protocol. RFC 2748 (Proposed Standard), Jan. 2000. Updated by RFC 4261. 40

[37] R. E.800. terms and definitions related to quality of service and network performance including dependability. *ITU-T*, August 1994. 27

[38] B. M. et Al. SIP Extensions for Media Authorization. Internet-Draft draft-ietf-sip-call-auth-01, Internet Engineering Task Force, 2000. 41

[39] ETSI WG. Telecommunications and internet converged services and protocols for advanced networking, 2004. `http://portal.etsi.org/tispan/TISPAN-ToR.asp`. 42

[40] ETSI WG. Telecommunications and internet converged services and protocols for advanced networking (tispan) ; ngn functional architecture release 1, August 2005. 42

[41] ETSI WG. Telecommunications and internet converged services and protocols for advanced networking (tispan) ; ngn functional architecture ; network attachment subsystem (nass), June 2006. 42

[42] ETSI WG. Telecommunications and internet converged services and protocols for advanced networking (tispan) ; resource and admission control sub-system (racs) ; functional architecture, March 2006. 42

[43] ETSI WG. TISPAN : IP Multimedia Subsystem (IMS) : Functional Architecture, June 2006. 42

[44] A. Farrel, J.-P. Vasseur, and J. Ash. A Path Computation Element (PCE)-Based Architecture. RFC 4655 (Informational), Aug. 2006. 128

[45] F. L. Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, and J. Heinanen. Multi-Protocol Label Switching (MPLS) Support of Differentiated Services. RFC 3270 (Proposed Standard), May 2002. Updated by RFC 5462. 31, 136

[46] M. C. Fischer. Aggregate level simulation protocol (alsp) managing confederation development. In *Proceedings of the 26th conference on Winter simulation*, pages 775–780, 1994. 13

[47] S. Giordano, M. Listanti, F. Mustacchio, S. Niccolini, S. Salsano, and L. Veltri. Sip originated dynamic resource configuration in diffserv networks : Sipcopstraffic control mechanisms. *QoS-IP 2003 : quality of service in multiservice IP networks*. 41

[48] D. Goderis. Internet design for sla delivery from service level agreement to per-hop behavior : a report on the 1st tequila workshop. June 2001. 43

[49] H. Granzer, B. Koch, M. Winter, P. Sampatakos, I. Venieris, H. Hussmann, F. Ricciato, and S. Salsano. Aquila : Adaptive resource control for qos using an ip-based layered architecture. January 2003. 43

[50] G. Gross, H. Sinnreich, D. Rawlins, and S. Thomas. QoS and AAA Usage with SIP based IP Communications. *http ://tools.ietf.org/html/draft-gross-sipaq-00*. 41

[51] P. Gross. Choosing a Common IGP for the IP Internet. RFC 1371 (Informational), Oct. 1992. 29

[52] P. A. A. Gutierrez, I. Miloucheva, D. Wagner, C. Niephaus, A. Flizikowski, N. V. Wambeke, F. Armando, C. Chassot, and S. P. Romano. NETQOS Policy Management Architecture for Flexible QoS Provisioning in Future Internet. September 2008. 43

[53] J. F. Halpin. Completeness conditions for indeterministic theories. *Philosophical Studies*, 49 :373–384, 1986. 99

[54] L. N. Hamer, B. Gage, and H. Shieh. Framework for Session Set-up with Media Authorization. RFC 3521 (Informational), Apr. 2003. 41

[55] R. Hancock, G. Karagiannis, J. Loughney, and S. V. den Bosch. Next Steps in Signaling (NSIS) : Framework. RFC 4080 (Informational), June 2005. 42, 127

[56] M. Handley and V. Jacobson. SDP : Session Description Protocol. RFC 2327 (Proposed Standard), Apr. 1998. Obsoleted by RFC 4566, updated by RFC 3266. 39

[57] M. Handley, V. Jacobson, and C. Perkins. SDP : Session Description Protocol. RFC 4566 (Proposed Standard), July 2006. 39, 117

[58] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. SIP : Session Initiation Protocol. RFC 2543 (Proposed Standard), Mar. 1999. Obsoleted by RFCs 3261, 3262, 3263, 3264, 3265. 39

[59] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. RFC 2597 (Proposed Standard), June 1999. Updated by RFC 3260. 35

[60] S. Herzog, J. Boyle, R. Cohen, D. Durham, R. Rajan, and A. Sastry. COPS usage for RSVP. RFC 2749 (Proposed Standard), Jan. 2000. 40

[61] D. J. V. Hook and J. O. Calvin. A protocol independent compression algorithm (pica). 1994. 6, 15

[62] IEEE 1278.1a-1998 STD. 1278.1a-1998 - IEEE Standard for Distributed Interactive Simulation Application Protocols. *DIS WG - Working Group for Distributed Interactive Simulation*. 5

[63] IEEE-1516. IEEE standard for modeling and simulation high level architecture (hla). *http ://standards.ieee.org*, Janvier 2000. 7

[64] IEEE 1516-2000. Standard for modeling and simulation high level architecture - framework and rules. *http ://standards.ieee.org/findstds/standard/1516-2010.html*. 8

[65] IEEE Std 802.1D. IEEE standards for local and metropolitan area networks ; media access control (mac) bridges. *IEEE Standards*. 28

[66] IEEE Std 802.1Q. IEEE standards for local and metropolitan area networks ; virtual bridged local area networks. *IEEE Standards*. 28

[67] IEEE1516.1-2000. Standard for modeling and simulation high level architecture, federate interface specification. *http ://standards.ieee.org/findstds/standard/1516-2010.html*. 12, 57

[68] IEEE1516.2-2000. Standard for modeling and simulation high level architecture - object model template (omt) specification. *http ://standards.ieee.org/findstds/standard/1516-2010.html*. 9

[69] IEEE1516.3-2003. Recommended practice for high level architecture federation development and execution process (fedep). *http ://standards.ieee.org/findstds/standard/1516-2010.html*. 8

[70] ITU-T. Packet-based multimedia communications systems. Recommendation H.323, International Telecommunication Union, Geneva, Nov. 2009. 38

[71] V. Jacobson, K. Nichols, and K. Poduri. An Expedited Forwarding PHB. RFC 2598 (Proposed Standard), June 1999. Obsoleted by RFC 3246. 35

[72] R. Johnson. Open Source POSIX Threads for Win32, 2006. 58

[73] R. Joshi and G.-P. Castellote. A comparison and mapping of data distribution service and high-level architecture. *http ://www.rti.com/whitepapers*. 16

[74] A. Kantawala, G. Parulkar, and J. DeHart. Supporting dis applications using atm multipoint connection caching. *Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*. 7

[75] S. W. Kim and K. H. Ko. Kalman filter based dead reckoning algorithm for minimizing network traffic between mobile nodes in wireless grid. *Proceedings of the 9th Pacific Rim international conference on Artificial intelligence*, Octobre 2006. 91

[76] F. Kuhl, R. Weatherly, and J. Dahmann. *Creating computer simulation systems : an introduction to the high level architecture*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999. 7

[77] B. S. Lee, W. Cai, S. J. Turner, and L. H. Chen. Adaptive dead reckoning algorithms for distributed interactive simulation. *Proceedings of the thirteenth workshop on Parallel and distributed simulation*, 1999. 91, 98, 99

[78] R. Levenshteyn, I. Fikouras, S. Loreto, and G. Camarillo. Addressing & invocation of ims-attached services. In *Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications*, IPTComm '07, 2007. 42

[79] N. Li, X. Y. Peng, M. H. Zhang, M. Wang, and G. H. Gong. Multimedia communication over hla/rti. *Simulation Modelling Practice and Theory*, 14, February 2006. 54

[80] E. S. Liu, M. K. Yip, and G. Yu. Scalable interest management for multidimensional routing space. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '05, pages 82–85, 2005. 15

[81] K. Lougheed and Y. Rekhter. Border Gateway Protocol (BGP). RFC 1105 (Experimental), June 1989. Obsoleted by RFC 1163. 30

[82] J. W. M. McAuliffe, R. Long and D. Nocera. Testing the implementation of dis dead reckoning algorithms. *SISOSTD*, Octobre 2004. 6, 86

[83] G. Malkin and R. Minnear. RIPng for IPv6. RFC 2080 (Proposed Standard), Jan. 1997. 30

[84] A. Mankin, F. Baker, B. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib, and L. Zhang. Resource ReSerVation Protocol (RSVP) – Version 1 Applicability Statement Some Guidelines on Deployment. RFC 2208 (Informational), Sept. 1997. 39

[85] D. Miller and J. Thorpe. SIMNET : the advent of simulator networking. *Proceedings of the IEEE*, August 1995. 5

[86] D. Mills, J. Martin, J. Burbank, and W. Kasch. Network time protocol version 4 : Protocol and algorithms specification. *Request for Comments : 5905*. 74

[87] M. Morganthaler and J. Shockley. Performance in sending high volume data. *Simulation Interoperability Workshop, 98F-SIW-214*, September 1998. 54

[88] M. Morgenthaler, J. W. Shockley, and A. Vaiman. Rti performance in sending high volume data : An update–the development of an rti vtc capability. *Simulation Interoperability Workshop-99S-SIW-102*, 1999. 54

[89] B. Mostafa, C. Ming-Hsing, L. Margaret, G. Michael, and W. Jim. Performance and reliability analysis of relevance filtering for scalable distributed interactive simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 7 :293–331, July 1997. 15

[90] J. Moy. OSPF specification. RFC 1131 (Proposed Standard), Oct. 1989. Obsoleted by RFC 1247. 30

[91] J. Moy. OSPF Protocol Analysis. RFC 1245 (Informational), July 1991. 30

[92] J. Moy. OSPF Version 2. RFC 1247 (Draft Standard), July 1991. Obsoleted by RFC 1583, updated by RFC 1349. 30

[93] J. Moy. OSPF Version 2. RFC 1583 (Draft Standard), Mar. 1994. Obsoleted by RFC 2178. 30

[94] K. Muthukrishnan and A. Malis. A Core MPLS IP VPN Architecture. RFC 2917 (Informational), Sept. 2000. 31

[95] T. Nadeau and S. Hegde. Multiprotocol Label Switching (MPLS) Label-Controlled Asynchronous Transfer Mode (ATM) and Frame-Relay Management Interface Definition. RFC 4368 (Proposed Standard), Jan. 2006. 31

[96] K. Nichols, V. Jacobson, and L. Zhang. A Two-bit Differentiated Services Architecture for the Internet. RFC 2638 (Informational), July 1999. 36

[97] NIST. JAIN-SIP Presence Proxy. `http://dns.antd.nist.gov/proj/iptel/nist-sip-downloads.html`, Year = 2011. 120

[98] NIST. JAIN-SIP Project. `https://jain-sip.dev.java.net/`, Year = 2011. 119

[99] Object Computing, Inc. (OCI). *OCI's TAO Developer's Guide*. 2011. 16

[100] Object Management Group. Data distribution service for real-time systems (dds). *http ://www.omg.org/technology/documents/*, Janvier 2007. 15

[101] OMG. The real-time publish-subscribe wire protocol dds interoperability wire protocol specification, 2009. 23

[102] D. Oran. OSI IS-IS Intra-domain Routing Protocol. RFC 1142 (Informational), Feb. 1990. 30

[103] C. O'Ryan, D. C. Schmidt, and J. R. Noseworthy. 14

[104] P. Owezarski, P. Berthou, Y. Labit, and D. Gauchard. Laasnetexp : a generic polymorphic platform for network emulation and experiments. *4th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*. 74

[105] Perf. Eng. Research Center. wanem 2.0 User GuidePerformance, 2008. 138

[106] E. Phillip. The internet 2 qbone project architecture and phase 1 implementation. Springer-Verlag, 1999. 36, 42

[107] M. Poikselka and G. Mayer. *The IMS : IP Multimedia Concepts and Services, 3rd Edition*. Wiley, 2006. 42

[108] S. Poretsky and I. Property. Considerations for Benchmarking Link-State IGP Data Plane Route Convergence. Internet-Draft draft-ietf-bmwg-igp-dataplane-conv-app-15, Internet Engineering Task Force, Feb. 2008. Work in progress. 29

[109] C. Raczy, G. Tan, and J. Yu. A sort-based ddm matching algorithm for hla. *ACM Trans. Model. Comput. Simul.*, 15(1) :14–38, January 2005. 15

[110] W. Roberto, S. Patrick, D. Michel, and P. de Saqui-Sannes. A formal framework for the specification, analysis and generation of standardized hypermedia documents. *Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems*, Juin 1996. 55

[111] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP : Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141. 119

[112] RTI Innovation. Real-time innovations offers first commercial product based on data distribution service standard, 2004. 16, 23, 104

[113] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP) : Core. RFC 6120 (Proposed Standard), Mar. 2011. 125

[114] S. Salsano. Cops usage for diffserv resource allocation (cops-dra). *IETF Draft, draft-salsano-cops-dra-00*. 41

[115] S. Salsano and L. Veltri. Qos control by means of cops to support sip-based applications. *IEEE Network Magazine*. 41

[116] H. Satou. AAA and Admission Control Framework for Multicasting. Internet-Draft draft-ietf-mboned-multiaaa-framework-07, Internet Engineering Task Force, July 2008. Work in progress. 38

[117] D. C. Schmidt and H. van t Hag. Addressing the challenges of mission-critical information management in next-generation net-centric pub/sub systems with opensplice dds. April 2008. 16, 23

[118] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP : A Transport Protocol for Real-Time Applications. RFC 1889 (Proposed Standard), Jan. 1996. Obsoleted by RFC 3550. 54

[119] H. Schulzrinne and R. Hancock. GIST : General Internet Signalling Transport. Internet-Draft draft-ietf-nsis-ntlp-16, Internet Engineering Task Force, July 2008. Work in progress. 42

[120] S. Shenker, C. Partridge, and R. Guerin. Specification of Guaranteed Quality of Service. RFC 2212 (Proposed Standard), Sept. 1997. 32

[121] J. Shing, R. Jang, and C. T. Sun. Neuro-fuzzy modeling and control. *Proceedings of the IEEE*, Mars 1995. 91

[122] J. W. Shockley, K. Parsons, and M. Morgenthaler. Developing an hla virtual command post, 1999. 54

[123] S. K. Singhal. Effective remote modeling in large-scale distributed simulation and visualization environments. Technical report, Stanford, CA, USA, 1996. 98

[124] S. K. Singhal and D. R. Cheriton. Using a position history-based protocol for distributed object visualization. Technical report, Stanford, CA, USA, 1994. 98

[125] A. Smith, D. Partain, and J. Seligson. Definitions of Managed Objects for Common Open Policy Service (COPS) Protocol Clients. RFC 2940 (Proposed Standard), Oct. 2000. 40

[126] P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663 (Informational), Aug. 1999. 102

[127] R. Stewart. Stream Control Transmission Protocol. RFC 4960 (Proposed Standard), Sept. 2007. Updated by RFCs 6096, 6335. 39

[128] S. Symington, D. Wood, and M. Pullen. Modeling and Simulation Requirements for IPng. RFC 1667 (Informational), Aug. 1994. 87

[129] S. Taylor. Using determinism to improve the accuracy of dead reckoning algorithms. In *in Proc. of Simulation Technologies and Training Conference*, 2000. 91

[130] N. Thi-Mai-Trang, P. Guy, and B. Nadia. COPS Usage for SLS negotiation (COPS-SLS). *http ://www.ist-tequila.org/standards/draft-nguyen-rap-cops-sls-00.txt*, June 2001. 40

[131] W. C. Turner and B. P. Gan. Hierarchical federations : an architecture for information hiding. *Workshop on Parallel and Distributed Simulation archive*, May 2001. 14

[132] C. Van Ham and T. Pearce. The sip-rti : An hla rti implementation supporting interoperability. In *Proceedings of the 10th IEEE-DS-RT '06*, DS-RT '06, pages 227–234, 2006. 14

[133] L. Veltri, S. Salsano, and D. Papalilo. Sip extensions for qos support in diffserv networks. *IETF Draft, draft-veltri-sip-qsip-00.* 41

[134] D. Wood and M. Petty. HLA Gateway. *Proceedings of the Spring Simulation Interoperability Workshop*, March 1999. 14

[135] J. Wroclawski. Specification of the Controlled-Load Network Element Service. RFC 2211 (Proposed Standard), Sept. 1997. 32

[136] J. Wroclawski. The Use of RSVP with IETF Integrated Services. RFC 2210 (Proposed Standard), Sept. 1997. 39

[137] R. Yavatkar, D. Hoffman, Y. Bernet, F. Baker, and M. Speer. SBM (Subnet Bandwidth Manager) : A Protocol for RSVP-based Admission Control over IEEE 802-style networks. RFC 2814 (Proposed Standard), May 2000. 38

[138] R. Yavatkar, D. Pendarakis, and R. Guerin. A Framework for Policy-based Admission Control. RFC 2753 (Informational), Jan. 2000. 38

[139] H. Zhao and N. D. Georganas. An approach for stream retrieval over hla-rti in distributed virtual environments. In *Proc. IEEE DS-RT*, 2000. 54

Bibliographie de L'auteur

Publications

Revue

1. coming soon
2. coming soon

Publications à des conférences avec comité de lecture

1. **Akram Hakiri**, Aniruddha Gokhale, Douglas C. Schmidt, Berthou Pascal, Joe Hoffert, and Gayraud Thierry, A SIP-based Network QoS Provisioning Framework for Cloud-hosted DDS Applications, 1st International Symposium on Secure Virtual Infrastructures (DOA-SVI'11), Oct 17-19, 2011, Crete, Greece, pp.507-524
2. **Akram Hakiri**, Pascal Berthou,and Thierry Gayraud, Design of low cost PC-based simulators for education and training purpose using DDS International Conference on Computer as a Tool (EUROCON 2011), Lisbonne (Portugal), 27-29 Avril 2011, 4p.
3. **Akram Hakiri** and Michel Diaz, A formal model for the specification and analysis of HLA based distributed multimedia interactive simulation using hierarchical time stream petri nets International Conferences on Advances in Multimedia, MMEDIA 2011, Budapest (Hongrie), 17-22 Avril 2011, pp.23-29
4. **Akram Hakiri**, Pascal Berthou,and Thierry Gayraud, Controlled stochastic petri net model for end-to-end network QoS provisioning in middleware-based multimedia and real-time systems Spring Simulation Multi Conférences, Annual Simulation Symposium (ANSS11), Boston (USA), 4-9 Avril 2011, pp.118-125
5. **Akram Hakiri**, Pascal Berthou,and Thierry Gayraud, QoS-enabled ANFIS dead reckoning algorithm for distributed interactive simulation IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2010), Faifax (USA), 17-20 Octobre 2010, pp.33-42
6. **Akram Hakiri**, Pascal Berthou,and Thierry Gayraud, Survey study of the QoS management in distributed interactive simulation through dead reckoning algorithms Euro Simulation Interoperability Workshop (Euro SIW 2010), Ottawa (Canada), 12-15 Juillet 2010, 10p.
7. **Akram Hakiri**, Pascal Berthou,and Thierry Gayraud,Addressing the challenge of distributed interactive simulation with data distribution service Euro Simulation Interoperability Workshop (Euro SIW 2010), Ottawa (Canada), 12 - 15 Juillet 2010, 9p.

8. **Akram Hakiri**, Michel Diaz, Slim Abdellatif, Pascal Berthou, and Thierry Gayraud, Multi-level model for synchronizing temporal streams on HLA based distributed multimedia applications using HTSPN International Conference on Advances in Multimedia (MMEDIA), Athènes (Grèce), 13-19 Juin 2010, 8p.
9. **Akram Hakiri**, Pascal Berthou,Julien Henaut, Daniella Dragomirescu, and Thierry Gayraud, Performance Evaluation of Wireless Sensor Network for Spacial and Aeronautic Systems International Conference on Telecommunications (IEEE ICT 2010), Doha (Qatar), 5-7 Avril 2010, 8p.
10. Julien Henaut, **Akram Hakiri**, Pascal Berthou, Daniella Dragomirescu, Thierry Gayraud, and Robert Plana, Wireless field buses for aerospace ground and in-flight testing : an experiment, 8th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems (FeT'2009), Ansan (Corée), 20-22 Mai 2009, pp.89-96

Papiers courts, Posters et Demos avec comité de lecture

1. **Akram Hakiri**, Berthou Pascal, Gayraud Thierry, Aniruddha Gokhale, Joe Hoffert and Douglas C. Schmidt, Poster : SIP-based QoS Support and Session Management for DDS-based Distributed Real-time and Embedded Systems, Poster Proceedings of the 5th ACM International Conference on Distributed Event-based Systems (DEBS' 11), Yorktown Heights, NY, USA, July 11-15, 2011.
2. **Akram Hakiri**, Etude des applications de simulation distribuée interactive et l'algorithme dead reckoning EDSYS 2010. 11ème Congrès de Doctorants, Toulouse (France), 6-7 Mai 2010, 8p.
3. **Akram Hakiri**, Berthou Pascal, and Gayraud Thierry, Architecture de communication pour l'interconnexion temps réel de simulateurs distribuée EDSYS 2009. 10ème Congrès de Doctorants, Toulouse (France), 14-15 Mai 2009, pp.1-8
4. **Akram Hakiri**, Berthou Pascal, and Gayraud Thierry, Mise en oeuvre de simulateurs de conduite : de la conception à l'implémentation Pôle et École de recherche RESCOM2009, Palmyre (France), 7- 12 Juin 2009

Rapports Techniques

1. **Akram Hakiri**, Michel Diaz, Pascal Berthou, Slim Abdellatif, Thierry Gayraud, Le projet PLATSIM. Réseau pour interconnexion de simulateurs–analyse et approche de conception , 1^{st} Deliverable of PlatSim project, January 2010, 77p.
2. **Akram Hakiri**, Michel Diaz, Pascal Berthou, Slim Abdellatif, Thierry Gayraud, Le projet PLATSIM. Réseau pour interconnexion de simulateurs–interconnexion en réseaux locaux à QoS, 2^{nd} Deliverable of PLATSIM project, March 2011, 83p.
3. Slim Abdellatif, Amira Zammeli, Pascal Berthou, **Akram Hakiri**, Michel Diaz, Thierry Gayraud, Le projet PLATSIM. Réseau pour interconnexion de simulateurs–paramétrage automatique de la QoS DDS à l'aide des ontologies, 3^{rd} Deliverable of PlatSim project (an update), December 2011, 101p.

4. **Akram Hakiri**, Michel Diaz, Pascal Berthou, Slim Abdellatif, Thierry Gayraud, Le projet PLATSIM. Réseau pour interconnexion de simulateurs–interconnexion sur des réseaux étendus à QoS, 4^{nd} Deliverable of PLATSIM project, January 2012, 70p.
5. **Akram Hakiri**, Pascal Berthou, Thierry Gayraud, Publish/Subscribe middleware for Time-Critical and Large Scale Distributed Communication, 2010-02-18, Report–LAAS 10103
6. **Akram Hakiri**, Pascal Berthou, Julien Henaut, Daniella Dragomirescu, and Thierry Gayraud, Wireless Sensor Network for Spatial and Aeronautic Communications, 2009-09-21, Report-LAAS 09543

Presentations

Akram Hakiri, Mise en réseau de simulateurs de conduite : De la conception à la mise en oeuvre, Journée du pôle SINC, 28 avril 2009, Le mas des canelles, Castanet-Tolosan

Service Professionnel

Responsabilités éditoriales revue/livre

- **JZUS :** International Journal, Journal of Zhejiang University Science C (Computers & Electronics)
- **IARIA :** International Journal On Advances in Telecommunications
- **Wiley :** International Journal, Software : Practice and Experience
- **IEEE :** International Journal, IEEE Potentials Magazine.

Comités techniques du programme

- **IARIA :** The Third International Conferences on Advances in Multimedia, MMEDIA 2012
- **IARIA :** The Third International Conferences on Advances in Multimedia, MMEDIA 2011
- **EDSYS :** congrès des doctorants EDSYS, 2010

Participation à des Organisations de Normalisation

Membre de l'organisation SISO (Simulation Interoperability Standards Organization)

Membre de sociétés professionnelles

- Membre étudiant à l'IEEE (2007-2011) ;
- Membre à l'IEEE (depuis 2011-présent) ;
- IEEE Communications Society (depuis 2009) ;
- Membre ACM (depuis 2011).