Université Toulouse III – Paul Sabatier

École Doctorale Mathémathiques Informatique et Télécommunications

THÈSE

en vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE
délivré par l'Université Toulouse III - Paul Sabatier

Discipline : Informatique

présentée et soutenue par

Souad EL MERHEBI

le 15 avril 2008

La gestion d'effet : une méthode de filtrage pour les environnements virtuels répartis

Directeur de Thèse : Jean-Pierre JESSEL

JURY

Président :	Jean-Marc PIERSON
Rapporteurs :	Bruno ARNALDI Jacques TISSEAU
Examinateurs :	Jean-Marc PIERSON Nancy RODRIGUEZ
Encadrant :	Patrice TORGUET

Institut de Recherche en Informatique de Toulouse — UMR CNRS 5505

CNRS - INPT - UPS - UT1 - UTM

Remerciements

Pour commencer, je remercie Bruno Arnaldi et Jacques Tisseau, rapporteurs, Nancy Rodriguez et Jean-Marc Pierson, examinateurs, d'avoir accepté d'évaluer mon travail et d'avoir eu l'amabilité de participer au jury.

Je remercie sincèrement Jean-Pierre Jessel d'avoir accepté d'être mon directeur de recherche et de m'avoir soutenue et encouragée en étant toujours de bonne humeur même durant les temps de pluie.

Je tiens également à remercier Patrice Torguet d'avoir accepté d'être mon encadrant pendant mon stage de DEA et ma thèse. Je voudrais le remercier pour sa confiance, son écoute et ses conseils tout au long de ces années. Merci pour l'aide très active pendant la période de finalisation de la thèse.

Je souhaite aussi remercier Charles Tabet, Alexandre Sursock et Mouin Hamzé du CNRS Libanais qui se battent tous les jours pour permettre à des jeunes chercheurs, comme moi, de tracer leurs chemins malgré toutes les contraintes du pays.

Un grand merci aux membres de l'équipe Vortex pour leur hospitalité et leur bonne humeur quotidienne. Je remercie particulièrement Roger Pujado pour sa sérénité et son sourire contagieux. Merci à Anca pour ses encouragements et pour toutes nos petites discussions interculturelles passionnantes. Je voudrais aussi remercier Chaouki pour son amitié et son don de métamorphoser toute tragédie en une histoire qui finit bien. Merci à Samir et Vincent V. pour les pauses thésardes pleines de recherche (d'espoir, de rires et de gâteaux...). Merci à Vicent F. d'avoir participé à créer l'identité Voxarienne de recherche. Finalement, je voudrais remercier Wafaa pour nos discussions nostalgiques et créatrices qui n'ont malheureusement pas réussi à refaire le monde.

Je voudrais remercier mes parents pour leur amour et leur confiance chers et profonds. Merci de m'avoir appris tout ce que vous m'avez appris et de m'avoir toujours poussée à aller plus loin même lorsque les horizons sont étroits. Merci infiniement d'avoir traversé tant de kilomètres pour partager ce grand jour avec moi.

Je voudrais remercier mes amies qui m'ont soutenue de l'autre côté de la méditérannée. Un grand merci à Elvis pour sa grande amitié et pour m'avoir appris que la générosité n'a jamais de

limites. Je voudrais aussi remercier Aline pour ses mails quotidiens prouvant que l'amitié peut survivre et s'approfondir malgré la distance.

Je voudrais également remercier les amis que j'ai eu la chance de connaître à Toulouse. Je vais commencer par Rania que j'ai connue les premiers jours de mon arrivée. Merci Renno d'avoir toujours su me réconforter et merci pour toutes nos discussions philosophiques et gastronomiques. Un grand merci à Salam pour son amitié précieuse et sincère et pour ses petites attentions. Je voudrais aussi remercier Batoul pour sa disponibilité et sa bienveillance. Merci à Claire d'être une amie affectueuse et attentionnée. Merci aussi de m'avoir fait découvrir beaucoup sur la France et de m'avoir expliqué toutes ces expressions que je ne comprenais pas. Enfin, je voudrais remercier Laetitia pour les marchés excitants du Dimanche matin et merci de m'avoir toujours soutenue et remontée le moral à ta jolie façon.

Je voudrais aussi remercier toutes les personnes que j'ai rencontrées pendant mes années de thèse et qui m'ont beaucoup marquées. Merci à Mounira pour ces soirées pleines de foux rires. Merci à Ilham et Randa pour votre amitié simple et forte malgré la distance et le manque de temps.

Finalement, je voudrais remercier Jean-Christophe pour son aide et ses conseils qui ont permis à cette thèse de se finir paisiblement. Merci aussi d'avoir pris le temps de me relire et de m'aider. Enfin, merci pour ta générosité, ta patience et ton soutien pendant les moments les plus difficiles.

Résumé

Les environnements virtuels distribués (EVDs) sont destinés à fournir à leurs utilisateurs une expérience immersive au sein d'un environnement virtuel partagé. Pour cette raison, les EVDs essaient d'apporter aux différents participants des vues cohérentes du monde partagé. Ceci nécessite un échange intense de messages en particulier pour les EVDs fortement peuplés. Cet important échange de messages consomme beaucoup de ressources de calcul et réseau, ce qui ralentit le système et limite l'interactivité. Ainsi, la cohérence, l'interactivité et le passage à l'échelle sont trois besoins primordiales pour les EVDs. Par contre, ces besoins sont contradictoires : le besoin de cohérence requiert un échange plus important de messages alors que ceux d'interactivité et de passage à l'échelle demandent de diminuer au minimum ces échanges.

Pour gérer l'échange de messages d'une manière intelligente, les systèmes d'EVDs utilisent des méthodes de filtrage différentes. Parmi ces méthodes, les méthodes de gestion d'intérêt filtrent les messages en se basant sur les intérêts des utilisateurs dans le monde. Dans ce document, nous présentons notre méthode de gestion d'intérêt, la gestion d'effet. Cette méthode exprime les intérêts et les manifestations des participants dans les différents média à travers les zones de conscience et d'effet. Lorsque la zone de conscience d'un participant chevauche la zone d'effet d'un autre dans un média, le premier devient conscient du second dans ce média. De plus, pour un passage à l'échelle continu, la gestion d'effet a été développée au sein d'une architecture client/multi-serveurs qui gère les intérêts des participants à travers les serveurs.

La principale originalité de notre méthode réside dans le fait qu'elle répond aux différents besoins des applications d'EVDs comme l'expression de la manifestation, les relations asymétriques, l'adaptation aux environnements ouverts et aux architectures emboîtées, l'utilisation du multicast... Ces aspects permettent à la gestion d'effet d'effectuer un filtrage réaliste et efficace et d'être applicable dans différents types d'environnements.

Mots-clés: Environnements virtuels distribués, passage à l'échelle, interactivité, méthodes de filtrage.

Abstract

Distributed virtual environments (DVEs) are intended to provide an immersive experience to their users within a shared virtual environment. For this purpose, DVEs try to supply participants with coherent views of the shared world. This requires a heavy message exchange between participants especially with the increasing popularity of massively multiplayer DVEs. This heavy message exchange consumes a lot of processing power and bandwidth, slowing down the system and limiting interactivity. Indeed, coherence, interactivity and scalability are basic requirements of DVEs. However, these requirements are conflicting because coherence requires the more important exchange of messages that we can have while interactivity and scalability demand to decrease this exchange to minimum. For this reason, the management of message exchange is essential for distributed virtual environments.

To manage message exchange in an intelligent way, DVE systems use various filtering techniques. Among them, interest management techniques filter messages according to users' interests in the world. In this document, we present our interest management technique, the effect management. This technique expresses the interests and manifestations of participants in various media through conscience and effect zones. When the conscience zone of a participant collides the effect zone of another participant in a given medium, the first one becomes conscious of the second. This principle allows an efficient and realistic decrease of message exchange. Furthermore, in order to extend scalability, effect management is used within a client/multi-server architecture that manages the interests of participants between servers.

The main originality of our technique relies in the fact that it satisfies varied requirements of DVE applications such as the expression of manifestation, asymmetrical relations, adaptation to open and closed environments, the use of multicast... These aspects allow effect management to be applied in various kinds of virtual environments in a simple and efficient way.

Keywords: Distributed virtual environments, scalability, interactivity, filtering techniques.

Table des matières

Table des figures

1

Introduction

La réalité virtuelle est la technologie qui permet à un utilisateur d'interagir au sein d'un environnement tridimensionnel réel ou imaginaire et de se sentir immergé dedans. La réalité virtuelle distribuée est la suite logique de la réalité virtuelle avec la révolution de la communication. Elle permet à un utilisateur de partager cette expérience immersive avec d'autres participants au sein du même environnement virtuel. La réalité virtuelle et en particulier la réalité virtuelle distribuée deviennent de plus en plus importantes dans le monde de l'informatique, surtout avec la popularité ascendante des jeux en ligne.

Un participant dans une application d'environnements virtuels distribués exige une expérience suffisamment immersive et satisfaisante : il s'attend à un environnement virtuel réactif à ses interactions, il présume également être au courant de ce qui se passe autour de lui et être informé assez rapidement des événements qui ont lieu pour pouvoir réagir dans les meilleurs délais. En même temps, les environnements virtuels distribués les plus populaires de nos jours sont ceux qui permettent à un nombre assez élevé de participants simultanés d'interagir au sein du même monde virtuel (les MMOGs[1]). Ces exigences variées sont difficiles à atteindre parce qu'elles sont contradictoires : si nous voulons accueillir un nombre élevé de participants et les informer tous de ce qui se passe dans le monde partagé, il faudra échanger un nombre conséquent de messages de mise à jour. Cet échange intense de messages créera des congestions au niveau du système, du réseau et de l'application de l'utilisateur lui-même, ce qui ralentira l'application et la rendra moins interactive.

[1] Massively Multiplayer Online Games

Pour ces raisons, le filtrage des messages échangés dans les environnements virtuels distribués est considéré indispensable pour économiser les ressources et concilier les exigences d'interactivité, de cohérence et de passage à l'échelle. Une méthode de filtrage doit être transparente aux yeux des participants en livrant à chaque participant les messages concernant la partie de l'environnement dont il est conscient à une fréquence assez rapide à ses yeux. En même temps, la méthode de filtrage doit conserver un taux de messages échangés assez bas pour ne pas créer des congestions au niveau du réseau ou des machines avec le passage continu à l'échelle.

Les méthodes de filtrage existantes présentent deux types de solutions : la prédiction de comportement et la gestion des intérêts des participants. La prédiction de comportement des participants distants permet à un participant de n'avoir besoin de recevoir des mises à jour que lorsqu'un participant distant change de comportement. D'autre part, la gestion des intérêts des participants étudie pour chaque participant ses intérêts dans l'environnement et ne lui envoie que les messages qui l'intéressent. Pour ceci, les méthodes de gestion d'intérêt se basent sur la division de l'espace en régions ou sur l'étude individuelle des intérêts de chaque participant pour sélectionner les messages intéressant chacun d'eux.

Comme la prédiction de comportement n'est possible que si les comportements des participants sont prédictibles, nous nous sommes intéressés dans nos travaux à la gestion d'intérêt. Nous avons étudié les différentes méthodes de gestion d'intérêt et nous avons trouvé que chacune de ces méthodes possède une vision différente des intérêts des participants. Par conséquent, certaines de ces méthodes présentent des solutions à un ou plusieurs parmi les aspects suivants : l'utilisation de plusieurs médias de communication, l'expression de la manifestation des participants, l'établissement de relations asymétriques entre les participants, l'adaptation aux environnements virtuels ouverts, l'adaptation aux architectures emboîtées, l'utilisation du multicast, un passage à l'échelle continu... Ces aspects permettent aux applications d'environnements virtuels distribués de s'adapter à tous les domaines d'application possibles.

Notre objectif est de présenter une méthode de gestion d'intérêt qui répond à tous ces aspects de la manière la plus simple possible. Nous souhaitons ainsi fournir une méthode qui effectue un filtrage plus exact et par suite plus performant, permettant une meilleure combinaison entre l'interactivité et le passage à l'échelle.

Nous commencerons notre exposé en présentant les caractéristiques et les problématiques

des environnements virtuels distribués. Nous exposerons ensuite les différentes méthodes de filtrage, leurs caractéristiques, leurs avantages et leurs limitations. Nous analyserons ensuite les avancées de ces méthodes vis-à-vis des aspects présentés ci-dessus. Nous déduirons de notre étude la possibilité d'optimiser les méthodes de filtrage existantes et de présenter une méthode de gestion d'intérêt qui permet de répondre aux divers aspects indispensables d'une manière simple et efficace.

A partir de cette analyse, nous présenterons notre méthode de gestion d'intérêt, la gestion d'effet. Nous verrons que notre méthode permet l'utilisation des différents média de communication et l'expression de la manifestation dans chaque média : elle distingue, par exemple dans le média visuel, entre la capacité de voir et la capacité à être vu, ce qui permet d'exprimer la manifestation de chaque participant et d'établir des relations asymétriques entre eux.

Nous décrirons ensuite le système dans lequel nous avons mis en œuvre la gestion d'effet. Ce système s'adapte aux environnements ouverts et aux architectures emboîtées à l'aide de la division de l'espace en régions et la gestion des intérêts des participants à travers les régions. Le multicast a également été utilisé dans ce système dans le but d'économiser la consommation les ressources du réseau. Enfin, l'adoption d'une architecture client/multi-serveurs permet un passage à l'échelle continu et régulier.

L'évaluation de notre méthode et sa comparaison à d'autre méthodes existantes nous permettront de voir les avancées de la gestion d'effet et de constater les pistes à développer.

Première partie

État de l'art

1

Discussion sur les environnements virtuels distribués

1.1 La réalité virtuelle

La réalité virtuelle [AFT06] permet à un utilisateur d'interagir au sein d'un environnement tridimensionnel simulant un monde réel ou imaginaire. Les simulations au sein des environnements virtuels sont des expériences immersives [CBC06] principalement visuelles, affichées sur un écran d'ordinateur, des écrans de grande taille ou un visiocasque... La majorité des simulations diffusent des informations sonores à travers des enceintes ou des écouteurs. Des systèmes plus avancés utilisent des informations tactiles ou kinesthésiques à travers des périphériques à retour de force comme les gants de données à retour tactile, les bras à retour d'effort, les exosquelettes...

1.1.1 Domaines d'application

La réalité virtuelle a de nombreuses applications, sans être exhaustif :

- les jeux vidéo sur ordinateurs, consoles ou bornes d'arcades ;
- la formation par simulation : vol, conduite de véhicules, aérospatiale, médecine ;
- les applications médicales : traitement des phobies, simulation de chirurgie ;
- la télérobotique, les téléopérations : l'utilisation d'un environnement virtuel comme interface utilisateur d'un système de téléopération permet à l'opérateur de retrouver toute

l'information spatiale en détails et de comprendre et analyser l'environnement distant plus facilement. De plus, la simulation 3D interactive permet à l'opérateur d'expérimenter avec les machines et les objets de l'environnement distant sans s'inquiéter pour les dommages qui peuvent être provoqués par un usage inapproprié.

- les visites et les présentations de musées et de sites virtuels, la reconstruction d'objets et de sites détruits ou endommagés ;
- le cinéma : la production de cinéma d'animation connaît depuis quelques années une progression considérable. Les films d'animation se retrouvent sous les feux de la rampe (Toy Story en 1996, Shrek 1 en 2001, Nemo en 2003, Ratatouille en 2007, etc.) ;
- la visualisation scientifique : cette technique utilise les environnement tridimensionnel pour la représentation et l'analyse des données ; par exemple pour la météorologie, les représentations tridimensionnels des phénomènes atmosphériques permettent une meilleure visualisation et analyse de ces phénomènes ;
- l'architecture, l'urbanisme ;
- (...)

1.1.2 Historique

La réalité virtuelle a beaucoup avancé depuis sa création dans les années 60. La liste suivante retrace les dates clés de son évolution :

- **1960** Morton Heilig invente le Sensorama Simulator qui est un jeu électronique multisensoriel, associant vision en trois dimensions, odeur, son stéréo, vent et siège qui vibre, et simulant une randonnée à moto dans les rues de New York.
- **1966** Ivan Sutherland, professeur à Harvard, crée le visiocasque ou casque HMD (Head-Mounted Display), système d'affichage d'images de synthèse utilisant un petit écran pour chaque oeil.
- **1968** Création de la première souris ; elle est réalisée dans un des laboratoires de l'Advanced Research Projects Agency (ARPA) à partir des travaux de Englbart et Licklider.
- **1970** Daniel Vickers équipe le visiocasque d'un capteur de position et d'orientation de la tête.
- **1981** Sortie du premier gant de données, le Dataglove, inventé par Thomas Zimmerman.

Chaque doigt est doté de capteurs optiques permettant de connaître à tout instant la position des doigts par rapport à la main.

- **1982** Sortie au cinéma de Tron, film de Steven Lisberger. C'est le premier à utiliser des images de synthèse.
- **1984** Sortie du premier Macintosh Apple qui fait bénéficier ses utilisateurs de la première interface graphique conviviale. C'est l'annonce d'une grande course des développeurs pour créer des interfaces de plus en plus conviviales.

  Parution de Neuromancer, le livre de William Gibson qui fait naître le concept de "Cyberspace".
- **1989** Démonstration d'environnement virtuel, lors du SIGGRAPH : une pièce meublée visitable avec une combinaison de Réalité Virtuelle (celle de VPL).

  La NASA sort un casque équipé d'écrans à cristaux liquides.
- **1991** L'artiste Daniel Sandin et l'ingénieur Thomas DeFanti mettent au point l'environnement d'immersion CAVE (Cave Automatic Virtual Environment), dans lequel des images s'affichent sur trois murs et sur le sol.
- **1991** Sortie du film "Toy Story", le premier film 100% en image de synthèse des studios Pixar.
- **1994** Apparition des premiers systèmes de téléchirurgie.
- **1997** Un projet de Thierry Blandet au LSP permet de visualiser des données VRML dans un casque de réalité virtuelle de fabrication.
- **1998** Beaucoup d'interfaces de la Réalité Virtuelle comme les gants, les visiocasques deviennent accessibles au grand public.

Cet historique nous a décrit l'évolution des applications mono-utilisateur de la réalité virtuelle. On s'intéressera par la suite au côté multi-utilisateurs collaboratif de la réalité virtuelle, qui connaît actuellement les évolutions les plus importantes de la réalité virtuelle.

1.2 Les environnements virtuels distribués

Les systèmes d'environnements virtuels distribués (EVDs) [JT06] permettent à plusieurs utilisateurs distants, connectés par un réseau local ou longue distance, d'interagir en temps réel

au sein d'un monde virtuel partagé. Les utilisateurs sont représentés par des entités particulières appelées avatars. Le but de ces environnements est de créer une collaboration ou une compétition immersive partagée entre les participants.

La terminologie des "Environnements Virtuels Distribués" est assez variée. Les EVDs sont aussi connus comme des "Environnements Virtuels Collaboratifs" (CVEs : Collaborative Virtual Environments) ou comme des "Environnements Virtuels sur Réseau" (NVEs : Networked Virtual Environments). Nous utiliserons dans notre rapport le terme "Environnements Virtuels Distribués".

1.2.1 Domaines d'application

Les environnements virtuels distribués sont utilisés dans plusieurs domaines :

1. Les jeux vidéo : Une étude réalisée par le cabinet PricewaterhouseCooper estime les revenus mondiaux des jeux vidéo en 2007 à 37,5 milliards de dollars (25,6 milliards d'euros). Les ventes de jeux vidéo passeraient pour la première fois de leur histoire devant celles de la musique qui engrangeraient 35,6 milliards de dollars. L'étude estime que les revenus de l'industrie du jeu vont progresser de 9 % dans le monde, passant de 31,6 milliards de dollars (23,5 milliards d'euros) en 2006 à 48,9 milliards (36,3 milliards d'euros) en 2011.

   Les jeux en ligne occupent une part importante de l'industrie des jeux vidéo. Ils sont divisés en deux catégories principales :

   (a) Les MOGs (Multiplayer Online Games - jeux en ligne multi-joueurs). Ces jeux sont souvent établis sur le principe de sessions : pendant une session, chaque joueur essaie de survivre ou de gagner le plus de points. Une session a un nombre limité de joueurs : pour permettre plus de joueurs simultanés, les joueurs sont regroupés dans des sessions indépendantes n'ayant aucune coordination ou synchronisation entre elles.

   Les MOGs sont apparus dans les années 80 avec les jeux "mode-texte" comme les MUDs (Multi-User Dungeons). Avec la sortie de Doom dans les années 90, les jeux de tir subjectif (FPS - First-Person Shooter games), où plusieurs joueurs se battent les uns contre les autres, sont devenus populaires (Quake, Half Life...). Les jeux de stratégie temps réel (RTS - Real-Time Strategy games) ont suivis, profitant de l'évo-

lution d'Internet (Warcraft : Orcs & Humans, Warcraft II : Tides of Darkness...). D'autres types de jeux, comme les jeux sur navigateur, sont assez populaires. Ces jeux se jouent par l'intermédiaire d'un navigateur, directement à partir d'Internet, sans qu'aucun téléchargement ne soit nécessaire.

(b) Les MMOGs (Massively Multiplayer Online Games - les jeux en ligne massivement multi-joueurs) [YC06]. Les MMOGs se distinguent des MOGs par un nombre plus important de joueurs simultanés. Les MMOGs les plus populaires recensent des milliers de joueurs en ligne en permanence et peuvent atteindre des millions de joueurs enregistrés. Les MMOGs n'organisent pas des sessions comme les MOGS, ils présentent des mondes virtuels persistants où chaque joueur incarne un personnage permanent. Ces mondes sont toujours disponibles, des événements sont tout le temps en train de se produire même lorsque le joueur n'est pas connecté.

On distingue différentes catégories de MMOGs :

- Les MMORPGs (Massively Multiplayer Online Role-Playing Games - les jeux de rôle en ligne massivement multi-joueurs) sont les plus connus et les plus populaires comme World of Warcraft, EverQuest, Lineage...
- Les MMOFPS (Massively Multiplayer Online First-Person Shooter - les jeux de tir subjectif massivement multi-joueurs). Quelques MMOFPS ont été réalisés à partir des années 2000. Ces jeux présentent généralement un combat en équipe sur un vaste terrain. Le fait que la notion de persistance du monde soit présente ajoute des éléments que l'on trouve généralement dans les jeux de rôles, comme par exemple les points d'expérience. Le premier MMOFPS est probablement 10SIX (sorti en 2000). D'autres jeux populaires sont sortis depuis comme World War II Online ou PlanetSide.
- Les MMORTS (Massively Multiplayer Online Real-Time Strategy - les jeux de stratégie temps réel massivement multi-joueurs). Un certain nombre de développeurs ont essayé d'unir les jeux de stratégie en temps réel avec les MMOGs. On peut ainsi nommer Mankind ou Shattered Galaxy.

2. Un autre domaine d'applications concerne les simulations militaires : Depuis les années 80,

le département Américain de la défense développe des systèmes d'entraînement militaire collectif. Ils ont développé SIMNET [CDG+93], DIS [Com94], NPSNET [MZP+95, Mor96] . Ces systèmes continuent à évoluer jusqu'à présent à travers HLA.

3. D'autres applications : Les EVDs sont utilisés pour des buts collaboratifs divers comme la téléconférence, l'éducation, l'entraînement civil, la conception collaborative...

1.2.2 Historique

Voici un rappel synthétique des principales dates ayant marqué l'évolution des EVDs :

- **1980** MUD (Multi-User Dungeon), les premiers MUDs apparaissent en 1978, ils gagnent en popularité aux États-Unis dans les années 1980. Les joueurs incarnent un personnage et voient des descriptions textuelles de salles, d'objets ou d'autres personnages dans un monde virtuel. Ils peuvent interagir entre eux et avec l'environnement en tapant des commandes qui ressemblent au langage courant. Les MUDs ont permis de connecter jusqu'à quelques centaines de personnes.
- **1985** Amaze suit les MUDs. Amaze est un labyrinthe 2D où les joueurs peuvent se déplacer et se tirer dessus.
- **1983** Le département Américain de la défense commence le développement de SIMNET, une plateforme de simulateurs distribués de combat. SIMNET devient opérationnel en 1990.
- **1989** Sortie de RB2 (Reality Built for two). RB2 permet à deux utilisateurs de partager le même monde virtuel en utilisant des visiocasques et des gants de données.
- **1992** DIS, une évolution de SIMNET, devient un standard IEEE. DIS dispose de plus de flexibilité et de liberté.
- **1993** MR Toolkit est disponible. C'est un outil qui simplifie le développement d'applications d'EVDs.

  Doom, signifiant littéralement destin funeste, est un jeu vidéo de tir subjectif qui est sorti en 1993. Doom est connu pour être l'un des titres majeurs qui ont lancé ce type de jeu vidéo. Il est reconnu comme étant le pionnier des graphismes en trois dimensions immersifs, du jeu multi-joueurs en réseau, et à avoir permis aux joueurs de créer leurs propres contenus.

Distribué comme partagiciel, Doom a été téléchargé par approximativement 10 millions de personnes en une année.

- **1997** Sortie de Ultima Online, un jeu vidéo de rôle médiéval fantastique massivement multi-joueurs. Les autres MMOGs vont suivre au cours des années suivantes.
- **2000** HLA, la plateforme développée par le département Américain de la défense, devient un standard IEEE. Les travaux de recherche militaire américaine se concentrent sur ce standard. Le plus de HLA est qu'elle permet l'interopérabilité de simulations hétérogènes.
- **2003** Linden Lab lance Second Life, un jeu en ligne qui permet à ses joueurs, appelés résidents, d'explorer le monde, de rencontrer d'autres résidents, de socialiser, de participer à des activités individuelles ou collectives comme le commerce, les services... Au total, 9,9 millions de joueurs sont inscrits mais la majorité d'entre eux sont inactifs.
- **2004** Blizzard Entertainment lance son quatrième jeu de la série Warcraft universe, qui a été introduite par Warcraft : Orcs & Humans en 1994, suivi par WarcraftII et WarcraftIII. Ce jeu, un MMORPG appelé World of Warcraft, a un grand succès : en juillet 2007, Blizzard Entertainment annonce avoir 9 millions de joueurs dans le monde.

Nous avons vu comment au fil des années les EVDs ont pris une place des plus importantes sur la scène informatique. Cette importance grandissante évolue parallèlement avec une augmentation constante du nombre d'utilisateurs des EVDs, cela n'étant pas sans poser de nouveaux défis. C'est ce que nous allons voir dans le chapitre suivant.

1.3 Problématique

1.3.1 Besoins des environnements virtuels distribués

L'évolution rapide des environnements virtuels distribués a levé un nombre de défis :

1.3.1.1 Adaptabilité

Les EVDs peuvent avoir des utilisateurs connectés à travers des outils et des réseaux de capacités différentes. L'application doit donc s'adapter aux caractéristiques de chacun de ses utilisateurs. Elle doit prendre en considération que les utilisateurs peuvent ne pas disposer de périphériques spécialisés et ne pas avoir les mêmes systèmes d'exploitation. De plus, il se peut

qu'il y ait des participants ayant une connexion à très bas débit.

1.3.1.2 Extensibilité durant l'exécution

La possibilité d'étendre le système sans devoir l'arrêter est un besoin pour les systèmes qui offrent des services 24 heures/24, 7 jours/7, comme les jeux en ligne multi-joueurs. Ces systèmes doivent évoluer en permanence pour suivre les besoins du marché et de leurs participants. En même temps, un arrêt du service peut causer des pertes commerciales importantes. Ainsi l'extensibilité durant l'exécution s'impose comme un besoin pour les applications d'EVDs commerciales.

1.3.1.3 Sécurité

Les services de sécurité doivent être fournis dans les applications d'EVDs commerciales. Ces applications doivent empêcher toute triche au niveau de l'authentification et du déroulement de la simulation. En particulier, l'application doit surveiller que les participants respectent les règles en cours (les règles physiques, les actions autorisés...) et exclure les contrevenants de la simulation.

1.3.1.4 Causalité

Lorsque les utilisateurs sont engagés dans le déroulement d'actions simultanées, leur déroulement doit être perçu dans le même ordre par tous les utilisateurs. En effet, il est important pour le réalisme d'éviter les ambiguïtés dans les interactions : les causes doivent intervenir avant les effets.

1.3.1.5 Persistance

La sauvegarde des objets qui ont été créés par des participants distants est nécessaire. Cette sauvegarde doit être persistante même si les participants se sont déconnectés. La persistance exige donc une allocation de ressources adéquates qui doit être planifiée et organisée par l'application.

1.3.1.6 Cohérence

Quand un utilisateur entre dans un environnement virtuel, il choisit ou on lui assigne un avatar et une position de départ. Des fantômes de cet avatar sont créés chez les autres participants. Quand l'utilisateur interagit dans l'environnement, il change l'état de son avatar (position, orientation, vitesse...), il peut aussi changer l'état de l'environnement (défoncer un mur, manger un beau gâteau au chocolat...). Alors les résultats de l'interaction d'un utilisateur doivent être communiqués aux utilisateurs qui partagent son environnement pour que tout le monde ait une vue cohérente de l'environnement partagé.

1.3.1.7 Interactivité

Le sentiment d'immersion [BFV06a] d'un participant est fourni par plusieurs composants : les graphismes 3D réalistes, les sons, les interactions spéciales, les outils de visualisation... Mais tout ceci ne peut être suffisant si l'utilisateur n'a pas un retour rapide de ses interactions. Effectivement, si la latence est élevée, l'utilisateur perd forcément son sentiment d'immersion.

L'interactivité [Man00] est la limite en deçà de laquelle le participant se sent convaincu de l'effet mutuel d'actions et de réactions. Une meilleure interactivité fournit une interaction plus agréable et plus réaliste. Le niveau d'interactivité est donc une fonction des outils d'interaction du participant et de la rapidité de la réponse de l'interaction. Et pour que la rapidité de la réponse de l'interaction soit acceptable, il faut que le transfert des interactions entre les participants se fasse dans des délais acceptables.

1.3.1.8 Passage à l'échelle

Le passage à l'échelle[2] dans les EVDs [MKZB03] est associé à :

- la complexité des environnements virtuels associée au rendu graphique et aux comportements des entités simulées, etc.
- le nombre de participants simultanés et leur distribution géographique.

Le passage à l'échelle est un besoin prioritaire des EVDs actuels, surtout des jeux massivement multi-joueurs qui essaient d'avoir de plus en plus de participants simultanés.

[2] La traduction française de scalability.

1.3.2 Limitations

Les obstacles qui restreignent la performance et le passage à l'échelle des systèmes d'EVDs sont relatifs à la capacité et à la rapidité du traitement et de la communication.

1.3.2.1 Limitations des ressources Réseaux

Voici les facteurs qui affectent la performance du réseau :

1. **La bande passante** : La bande passante décrit la quantité de données que peut transporter le réseau en une seconde ; elle est mesurée en bits par seconde. La limitation de la bande passante impose des limitations sur la quantité de messages de mise à jour échangés entre les sites. C'est pour cette raison que l'on n'a pas une liberté absolue sur le nombre et la fréquence de messages qu'on veut échanger.
2. **La latence** : La latence [BFV06b] est le temps pris par un message pour passer de l'émetteur au destinataire ajouté au temps de traitement de l'information émise et reçue ; elle est mesurée en millisecondes. L'interactivité des utilisateurs souffre des transitions brusques entre les états dues aux délais causés par la latence : l'interaction d'un utilisateur n'apparaît aux autres utilisateurs qu'après ce délai. Ceci implique une divergence de l'environnement durant la durée du délai, suivie d'un bond soudain pour resynchroniser afin que l'état courant soit affiché. Ces discontinuités apparaissent anormales aux yeux des utilisateurs, d'où la nécessité de réduire la latence au minimum. La latence est plus importante sur Internet que sur les réseaux locaux, la moyenne considérée acceptable sur Internet est de 50 ms.
3. **La gigue** : La gigue est la variation de latence au cours du temps. Si elle devient élevée, le comportement des entités simulées ne sera plus uniforme et les observateurs sentiront un manque de réalisme à cause de l'irrégularité de la simulation des mouvements des entités qui sont censés avoir un comportement régulier ou monotone.

La bande passante et la latence ne sont pas directement liées. La bande passante dépend de la nature de la connexion et la latence dépend de la longueur du trajet que suit un message pour arriver à son destinataire. Mais l'une peut affecter l'autre : quand la bande passante est réduite, il faudra plus de temps pour faire passer toutes les informations dans ces "tuyaux" à capacité limitée. Et quand la latence est élevée, les messages occuperont plus longtemps une partie de la

bande passante. Ce qui réduira la bande passante effective en ne laissant plus de place suffisante pour les autres données qui veulent transiter.

1.3.2.2 Limitations des ressources machines

Le traitement des messages échangés au sein d'une application d'EVDs implique une consommation des ressources locales. Les limitations de ces ressources se manifestent à travers les limitations de la capacité du rendu, de la mémoire et de la charge du processeur. Ces limitations sont présentes au niveau des architectures égal-à-égal et client/serveur. Par contre, dans les architectures client/serveur le problème de limitations de ressources machines est plus imposant parce que la charge du serveur est beaucoup plus importante. En effet, la limitation des ressources du serveur est dangereuse parce elle est capable de limiter le passage à l'échelle et l'interactivité bien plus que la limitation au niveau des clients. Il faudra donc porter une attention importante à ne pas le surcharger.

1.3.3 Problématique et solutions

L'augmentation du nombre de participants impose plus de charge sur les machines et sur le réseau. Lorsque le nombre de participants augmente, le nombre de changements intervenant dans l'environnement augmente. Maintenir la cohérence nécessite alors un échange plus intense de messages. Cet échange impose une charge supplémentaire sur les machines et sur les réseaux qui ont des capacités limitées, ce qui peut aboutir à un ralentissement de l'application et donc une perte d'interactivité. Cette montée en charge peut aller jusqu'à causer dans les cas extrêmes l'effondrement de l'application. Pour cette raison, la maintenance d'une cohérence absolue s'avère irréalisable et heureusement non nécessaire. En effet, un utilisateur ne peut pas être intéressé par le monde virtuel entier pour de nombreuses causes : les limitations de la perception et de la conscience des avatars, la distance qui sépare les avatars, la conception de l'environnement virtuel (chambres, murs, obstacles...). Par conséquent, la cohérence peut être maintenue d'une manière partielle qui fournit à l'utilisateur une vue cohérente de la partie de l'environnement qui l'intéresse. Cette cohérence partielle est transparente aux yeux des utilisateurs et permet en même temps une meilleure interactivité et un meilleur passage à l'échelle. L'installation des règles de cohérence partielle qui constitue le filtrage qui minimise le nombre de messages reçus

par l'utilisateur et donc allège la charge sur l'utilisateur et sur le réseau. Cette cohérence partielle permettra une meilleure interactivité malgré les contraintes introduites par le passage à l'échelle.

En conclusion, maintenir l'équilibre entre la cohérence, l'interactivité et le passage à l'échelle est une opération nécessaire dans le développement des EVDs. Cette opération peut être assurée par une méthode de filtrage des messages échangés. Pour que les EVDs puissent donc satisfaire les besoins de cohérence, de passage à l'échelle et d'interactivité, surtout sur Internet, ils doivent mettre en œuvre une méthode de filtrage qui leur permet de s'adapter aux limitations des diverses ressources. C'est dans cette optique que notre étude s'est portée sur les méthodes de filtrage, leur évolution, leurs différences et a proposé également une nouvelle méthode de filtrage optimisé.

1.4 Caractéristiques des EVDs

1.4.1 Introduction

Les systèmes d'environnements virtuels distribués possèdent des caractéristiques assez différentes concernant leurs architectures de contrôle, leurs architectures de données, les modes de transmission et les protocoles de transport qu'ils utilisent. L'implémentation d'une technique de filtrage dépend fortement de ces caractéristiques. Voyons donc ces caractéristiques et leurs influences sur les techniques de filtrage.

1.4.2 Architectures des systèmes d'EVDs

Les architectures de contrôle et de données sont des éléments assez importants des systèmes d'EVDs. La méthode de filtrage doit prendre en considération ces architectures et même des fois être conçue pour elles.

1.4.2.1 Architectures de contrôle

L'architecture de contrôle définit l'organisation et les rôles des différents acteurs du système. Le choix de cette architecture affecte l'interactivité, la montée à l'échelle, la mise en œuvre d'une méthode de filtrage et bien d'autres composants.

Architectures client/serveur

Dans ces architectures, les clients communiquent entre eux à travers le serveur. Un client envoie ses données au serveur qui les renvoie aux autres clients.

Une architecture centralisée permet le contrôle de la distribution des utilisateurs dans l'environnement et le contrôle du déroulement de la simulation. De plus, l'état du monde reste persistant chez le serveur, ce qui résout les problèmes de concurrence et de causalité. Cette architecture permet aussi un gain de la bande passante du côté du client puisqu'un client n'envoie ses mises à jour qu'une seule fois au serveur et ne reçoit que les messages qui l'intéresse. En effet, la mise en œuvre du filtrage est facile et naturelle dans une architecture centralisée parce que le filtrage se fait au niveau du serveur : c'est le serveur qui décide, selon les propriétés du filtrage choisi, quelles données il va transférer et à qui.

Un des inconvénients de cette architecture est qu'elle rajoute un point de relais supplémentaire au transfert de messages. Les messages ne sont plus envoyés directement de l'expéditeur au destinataire, mais de l'expéditeur au serveur et du serveur au destinataire. Ce point de relais augmente la latence rendant la communication plus lente et donc moins interactive. D'autre part, quand le nombre de participants augmente, le serveur risque de devenir un goulot d'étranglement, ce qui augmenterait encore les délais de transfert et diminuerait d'autant l'interactivité. Enfin, le fait que le serveur soit l'unique point d'échec de l'application rend le choix d'utilisation d'un seul serveur assez dangereux.

Architectures client/multi-serveurs

Ces architectures sont utilisées pour résoudre les problèmes de limitations de ressources, de goulot d'étranglement et d'unique point d'échec des architectures centralisées mono-serveur. Elles présentent l'avantage d'augmenter les ressources disponibles, ce qui, associé à un algorithme de répartition de charge, permet d'accroître les capacités du système. En contrepartie, l'utilisation de ces architectures peut rajouter plus de points de relais au transfert de messages et donc plus de latence.

La distribution des utilisateurs chez les serveurs peut être faite de différentes manières : distribution arbitraire, distribution géographique selon les positions des utilisateurs dans le monde, distribution selon les positions des avatars dans l'environnement virtuel, distribution fonction-

nelle selon les caractéristiques des avatars...

Architectures égal-à-égal

Dans les architectures égal-à-égal[3], les participants communiquent directement entre eux, ce qui permet une réception rapide des informations. Pour cette raison, ce type d'architectures est très adapté aux systèmes fortement interactifs. Les architectures égal-à-égal résolvent aussi les problèmes de goulot d'étranglement et de point d'échec, mais introduisent plus de complexité pour chaque participant à cause des responsabilités de gestion de simulation et elles génèrent aussi beaucoup d'échange de données parce que chaque nœud peut envoyer des données à tous les autres nœuds. De plus, pour établir une méthode de filtrage, les participants doivent négocier en permanence leurs communications ; Ces négociations sont coûteuses en communication et en traitement. Ce qui fait que les architectures égal-à-égal consomment beaucoup plus de bande passante.

Architectures mixtes

Plusieurs systèmes combinent en même temps une architecture centralisée et une architecture égal-à-égal. Ces systèmes utilisent la centralisation pour l'authentification, la distribution des ressources, la persistance et pour le reste, la simulation se déroule comme dans une architecture égal-à-égal.

Choix d'une architecture de contrôle

Comme on a vu, le choix de l'architecture de contrôle affecte différents aspects des systèmes d'EVDs. La majorité des applications commerciales utilisent les architectures client/multi-serveurs pour les raisons suivantes :

– L'authentification et le paiement sont plus facile à établir dans les EVDs commerciaux puisque les utilisateurs s'enregistrent chez le fournisseur de jeux qui assure l'authentification et dirige le joueur vers un serveur. Ce serveur assurera ensuite le contrôle continu du déroulement de la simulation.

[3]Traduction française de peer-to-peer.

- La charge du filtrage est déplacée des clients aux serveurs. Les clients n'ont qu'à envoyer un seul message au serveur. C'est alors lui qui est responsable du filtrage et de la diffusion. Cette baisse importante de charge du côté client est importante pour les EVDs commerciaux, désireuses de ne pas imposer un standard matériel trop élevés à leurs potentiels clients.
- Les serveurs possèdent plus de capacité de calcul car ils ne font pas d'affichage ni de simulation de comportement. D'autre part, les serveurs peuvent avoir plus de ressources de mémoire et de calcul parce que la qualité des machines serveur peut être garantie au contraire des machines des clients.
- La rapidité de la connexion des serveurs peut également être garantie alors que les clients peuvent ne pas être tous sur des réseaux rapides. La lenteur de la connexion d'un client dans une architecture égal-à-égal peut affecter l'interactivité de la simulation.
- Même dans une architecture multi-serveurs, il y a en principe moins de serveurs que de clients. Le trafic de messages entre les serveurs sera alors moins important qu'entre les clients ; si la distribution des utilisateurs chez les serveurs est faite intelligemment, l'échange de messages entre les serveurs sera donc relativement faible.

1.4.2.2 Architectures de données

La distribution de la base de données des environnements virtuels est une caractéristique importante du système. La base de données peut être :

- centralisée chez les serveurs ;
- totalement ou partiellement répartie ;
- totalement ou partiellement dupliquée sur les différents sites.

Une combinaison quelconque entre ces méthodes est aussi possible. Mais quelque soit cette distribution, elle doit être maintenue cohérente aux yeux des participants malgré les requêtes auxquelles elle est soumise. La duplication s'est montrée comme étant l'approche la plus rapide et la plus performante. Pour maintenir la cohérence entre les duplicata de la base de données, les messages de mise à jour doivent être échangés le plus vite possible. La cohérence exacte dans l'ordre et dans le temps est bien sûr impossible à garantir puisque tous les utilisateurs ne peuvent pas être informés exactement au même moment des mises à jour de l'environnement. Ainsi plus

la cohérence est proche d'être exacte, meilleure est l'interactivité.

1.4.3 Modes de transmission

Pour la transmission de messages, plusieurs solutions sont disponibles. Voyons leurs avantages et leurs inconvénients :

1.4.3.1 Diffusion

La diffusion est un mode de transmission qui permet à une source d'envoyer des paquets de données à un ensemble de machines d'un même réseau. Cela est possible en utilisant des adresses spécifiques qui correspondent à l'ensemble des machines d'un réseau ou d'un sous-réseau. En utilisant la diffusion, l'expéditeur ne s'occupe pas de déterminer qui va recevoir son message. En plus, le paquet n'est transmis qu'une seule fois sur le réseau (dans le cas d'un réseau à diffusion). En contre partie, la plupart des informations délivrées aux participants ne leurs seront pas pertinentes, ce qui cause une perte de temps et de ressources. D'autre part, pour éviter que des paquets transitent de manière irraisonnée à travers toute la planète, la diffusion n'est possible qu'à l'intérieur d'un réseau local. Ainsi les paquets diffusés ne peuvent pas, en général, franchir les routeurs sur Internet.

La diffusion a été utilisée dans les premiers systèmes d'EVDs comme SIMNET. Mais, vu que ce mode de transmission ne permet aucun filtrage, il n'est plus adopté par les systèmes récents.

1.4.3.2 Unicast

L'unicast (aussi appelé "communication point à point") permet à une machine, dite source, d'envoyer des paquets vers une seule machine destinataire. Chacun des deux ordinateurs est identifié par une adresse réseau et un numéro de port. Les paquets de données sont routés sur le réseau suivant l'adresse du destinataire. Normalement, seul le destinataire intercepte et décode le paquet qui lui est adressé. Ce mode de transmission est très utilisé dans les applications d'EVDs.

1.4.3.3 Multicast

Le multicast (aussi appelé "diffusion restreinte") permet de transmettre un message en une seule fois vers un groupe de destinataires. Pour ceci, le multicast crée des groupes de machines

sur le réseau, appelés groupes multicast. Un message adressé à un groupe multicast est transmis une seule fois à tous les membres de ce groupe.

Le multicast est un mode de transmission nouveau et assez adapté aux EVDs. Pour ces raisons, on va détailler le fonctionnement, les limitations et l'adaptabilité du multicast aux EVDs.

Fonctionnement

L'ensemble des machines d'un groupe multicast est entièrement dynamique (une machine peut rejoindre ou quitter le groupe à tout moment) et ouvert (une machine peut émettre un message à un groupe sans en faire partie). Le résultat est semblable à la radio : Les expéditeurs diffusent sur plusieurs canaux, et les récepteurs captent les canaux qui les intéressent. La seule différence avec le multicast est qu'une machine peut souscrire à plusieurs groupes multicast simultanément.

Un groupe multicast est désigné par une adresse IP spéciale. Les émetteurs utilisent cette adresse pour envoyer leurs messages et les abonnés l'utilisent pour informer le réseau qu'ils sont intéressés à joindre ou quitter ce groupe. Le réseau met en œuvre une intelligence pour dupliquer les paquets quand les liens vers les destinataires se séparent au lieu de les dupliquer à la source. La duplication aura lieu dans les routeurs, permettant une économie de ressources, une accélération matérielle et une réception plus rapide du paquet.

Limitations du multicast

Le multicast a un nombre de limitations :

- Le support du multicast est obligatoire dans IPv6 mais il ne l'est pas dans IPv4. Alors pour le moment, les routeurs Internet sont divisés en trois catégories : ceux qui ne supportent pas le multicast, ceux qui supportent l'envoi mais pas la réception de paquets multicast (puisque l'envoi d'un message multicast ne nécessite pas de joindre le groupe multicast correspondant) et ceux qui supportent entièrement le multicast.

  Le transfert des messages multicast est donc soumis à des relais sur des routeurs qui ne supportent pas le multicast. C'est pour cette raison que le MBone ([Eri94]) a été créé. Le MBone est un réseau multicast virtuel construit par-dessus Internet, il est basé sur des

îles multicast connectées par des tunnels. Quand un routeur multicast (une île) transfère un message à un autre routeur qui ne supporte pas le multicast, il encapsule ce message dans un message unicast. Le routeur pourra donc le recevoir et le transférer et dès que le message arrive à une autre île, il est décapsulé pour reprendre son aspect d'origine. Le passage entre deux îles est appelé tunnel multicast. Pour l'instant, le Mbone n'est pas fourni par les fournisseurs d'accès Internet, mais il est souvent disponible dans les universités et les instituts de recherche.

D'autres solutions pour l'utilisation du multicast ont été présentées comme le multicast applicatif (Application-Level Multicast [YLE04]). Le multicast applicatif déplace les fonctionalités du multicast des routeurs aux utilisateurs. Ainsi, la gestion de l'adhésion aux groupes multicast, le routage et la duplication des paquets sont mis en œuvre chez les utilisateurs. Un arbre de distribution est construit dans la couche applicative. Les nœuds de cet arbre sont les utilisateurs (sources et destinataires). Ainsi un message émis suit l'arbre de distribution et sera dupliqué au moment optimal. La transmission des paquets entre les membres utilise l'unicast, de cette manière l'infrastructure du réseau n'a pas besoin de supporter le multicast.

– En ce moment, il existerait 268 millions d'adresses multicast si toutes les adresses multicast de la classe D (224.0.0.0 jusqu'à 239.255.255.255) étaient disponibles. Mais les adresses allant de 224.0.0.0 jusqu'à 224.0.0.255 sont réservées pour l'utilisation locale, les paquets destinés à ces adresses ne sont donc jamais renvoyés par les routeurs multicast. En outre, les adresses allant de 239.0.0.0 jusqu'à 239.255.255.255 sont réservées à des utilisations administratives. Le nombre restant de groupes ne permet pas à chaque application d'EVDs d'associer un groupe multicast par objet. Avec l'introduction de l'IPv6, la restriction du nombre de groupes multicast deviendra moins importante.
– Joindre et quitter un groupe multicast est coûteux en ressources réseau et calcul chez les clients et chez les routeurs. Ce coût doit être pris en compte lors de l'établissement de la politique de l'application.
– Le multicast est basé sur UDP, par conséquent il n'est pas fiable. Des tentatives de fiabilisation du multicast ont eu lieu mais elles ne sont pas encore assez mûres pour être adoptées par Internet.

Le multicast et les EVDs

L'utilisation du multicast est adaptée à la communication au sein des EVDs. En effet, une mise à jour d'un avatar a naturellement plusieurs destinataires, ce qui fait que la création de groupes de destinataires est très naturelle. L'échange de messages via le multicast constitue en lui même un filtrage intrinsèque au niveau de la communication.

En plus du gain des ressources réseau, le multicast rapporte un gain de calcul chez les expéditeurs qui, une fois qu'ils ont l'adresse d'un groupe multicast, n'auront plus besoin de calculer la liste des expéditeurs intéressés par chaque message envoyé. En revanche, envoyer les messages à un groupe multicast sans effectuer un calcul individuel d'intérêt pour chaque destinataire produit un filtrage qui n'est pas toujours exact. Mais dans certaines applications, les gains de calcul et de bande passante compensent ce manque de précision.

1.4.4 Protocoles de transport

1.4.4.1 TCP

TCP (Transmission Control Protocol) [Ste96] est un protocole de transport fiable, en mode connecté. Dans le modèle TCP/IP, TCP est situé entre la couche réseau (généralement le protocole IP) et la couche application. Les applications communiquent grâce à des flux d'octets. TCP découpe le flux d'octets en segments, dont la taille dépend de la MTU (Maximum Transmission Unit : la taille la plus grande d'un paquet que le réseau peut transmettre) du réseau sous-jacent (couche liaison de données). Une session TCP fonctionne en trois phases :

1. l'établissement de la connexion ;
2. les transferts de données ;
3. la fermeture de la connexion.

TCP assure la réception de tous les messages. Pourtant dans les applications d'EVDs l'essentiel est la réception de la dernière mise à jour, les mises à jour précédentes peuvent très bien être éliminées plutôt que d'arriver en retard. De plus, la gestion de la connexion (contrôle de flux et contrôle d'erreur) consomme du temps et diminue la réactivité de l'échange des messages. Par conséquent, l'utilisation de TCP n'est pas conseillée dans les environnements virtuels distribués.

1.4.4.2 UDP

UDP (User Datagram Protocol) [Ste96] est un protocole de transport non fiable. Il fait également partie de la couche transport de la pile de protocole TCP/IP. Ce protocole permet une transmission simple de paquets entre deux entités, chacune étant définie par une adresse IP et un numéro de port (pour différencier différents utilisateurs sur la même machine).

UDP présente certaines différences essentielles par rapport à TCP (figure 1.1). UDP travaille en mode non-connecté, contrairement à TCP. Il ne garantit pas l'arrivée des messages, ni l'ordre, ni la non duplication. De plus, il ne prévoit ni contrôle de flux, ni contrôle de congestion. UDP assure quand même un minimum de qualité de service en incluant dans les paquets une estampille temporelle et un numéro de séquence. C'est pour ces raisons qu'un datagramme UDP comporte une entête plus petite que celle de TCP et qu'il n'a pas besoin d'échanger des messages pour établir une connexion et assurer tous les services et les garanties par TCP. UDP a donc un fonctionnement plus simple et plus rapide que TCP, d'où l'intérêt de son utilisation dans les applications d'EVDs qui nécessitent un transfert rapide de données.

	UDP	TCP
Mode connecté	Non	Oui
Timeout and retransmission	Non	Oui
Garantie de l'ordre	Non	Oui
Détection de duplication	Non	Oui
Contrôle de flux	Non	Oui
Message boundaries	Oui	Non
Séquencement	Non	Oui
Accusé positifs	Non	Oui
Checksum	Opt.	Oui

FIG. 1.1 – Comparaison entre UDP et TCP

1.4.5 Conclusion

Les caractéristiques des systèmes d'EVDs sont assez variées. Certaines variations de ces caractéristiques sont bien adaptées aux besoins de filtrage (comme les architectures client/multi-serveurs, le multicast, l'UDP...), d'autres peuvent être adaptées à ces besoins (l'unicast par exemple) alors que certaines ne sont pas du tout adaptables (comme la diffusion).

2

Le filtrage dans les environnements virtuels distribués

2.1 Introduction

La cohérence, l'interactivité et le passage à l'échelle sont trois facteurs essentiels dans l'évolution des EVDs. Mais les limitations des ressources posent des contraintes sur ces composants interdépendants. En effet, pour garantir la meilleure cohérence possible, il faut échanger beaucoup de messages alors que pour assurer l'interactivité et le passage à l'échelle, il faut échanger le moins de messages possible. Ces trois composants ont ainsi des exigences contradictoires vis-à-vis de la communication. Les méthodes de filtrage se présentent alors pour régler le problème en minimisant l'échange de messages. Les messages échangés doivent assurer une cohérence suffisante qui n'affecte pas le sentiment de réalisme de l'utilisateur et aide à avoir un bon passage à l'échelle. Ces méthodes sont donc nécessaires pour maintenir l'équilibre entre la cohérence, l'interactivité et le passage à l'échelle. Elles permettent au final aux EVDs d'augmenter significativement leurs performances et leurs nombres de participants.

Beaucoup de systèmes d'EVDs ont été développés jusqu'à présent. Notre état de l'art porte que sur les méthodes de filtrage conçues dans ces systèmes. On peut classer les méthodes de filtrage en deux catégories :

1. **Les méthodes de prédiction de comportements** : Ces méthodes ont pour but de

prédire le comportement des objets distants, les mises à jour ne seront alors envoyées que lorsque le comportement d'un objet dévie du comportement prédit.

2. **Les méthodes de gestion d'intérêt** : Ces méthodes cherchent à réduire la quantité de données échangées selon les intérêts de chaque participant. L'intérêt d'un utilisateur dépend de beaucoup de facteurs : son comportement, sa position, sa fonction, etc.

2.2 La prédiction de comportement

Pour réduire la consommation de bande passante et surtout l'impact de la latence, les méthodes de prédiction envoient les messages de mise à jour moins fréquemment. Ces méthodes compensent le manque de mises à jour en prédisant les états des entités jusqu'à l'arrivée des nouvelles mises à jour. La prédiction se base sur les dernières positions reçues et sur les caractéristiques des entités pour prédire des positions approximatives en attendant les positions réelles. L'utilisateur calcule l'état prédit de son propre avatar et n' envoie ses mises à jour que si la différence entre l'état réel et l'état prédit dépasse un certain seuil (figure 2.1). Le dépassement de ce seuil indique l'incapacité des autres utilisateurs à prédire fidèlement l'état de son avatar. Les méthodes de prédiction permettent donc d'animer les entités avec une fréquence plus élevée que celle de la réception des mises à jour.

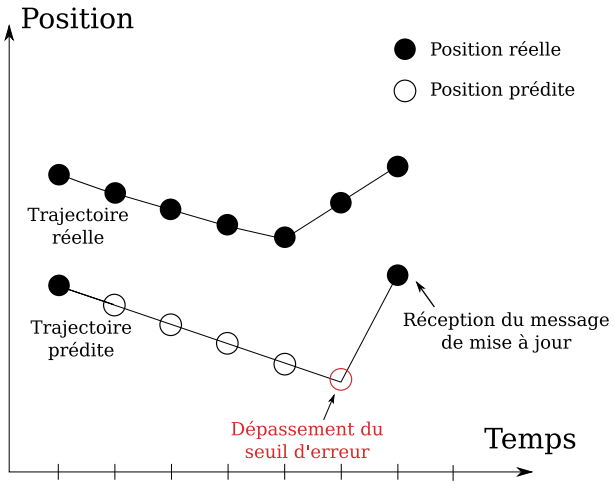

FIG. 2.1 – La prédiction du comportement d'une entité

Dans ce qui suit, on présente deux méthodes de prédiction le "dead-reckoning" et le modèle explicite de comportement.

2.2.1 Le “dead-reckoning”

Le “dead-reckoning” (terme de la marine qui peut se traduire par ”navigation à l’estime”) est la première méthode de prédiction de comportement qui a été conçu. Il fût mis en œuvre dans SIMNET [CDG+93], il est composé de deux techniques : une technique de prédiction et une technique de convergence.

La technique de prédiction utilise des dérivées polynomiales d’ordres différents. Les dérivées de premier ordre utilisent la vitesse pour déterminer la position prédite. Les dérivées de second ordre utilisent l’accélération et la vitesse, ce qui donne des résultats plus précis sauf si l’accélération des entités change fréquemment. Cette technique a évoluée grâce à plusieurs améliorations : l’utilisation des seuils dynamiques qui s’adaptent aux besoins de l’application [CLC99], des variantes de techniques d’extrapolation comme l’extrapolation basée sur les systèmes de classifieur [TTS07].

D’autre part, le passage de la position prédite à la position réelle peut être perturbant surtout si le seuil de limite de prédiction est important. Pour cette raison, le “dead-reckoning” utilise des méthodes de convergence permettant de ramener l’entité de sa position courante à sa position réelle de la façon la plus naturelle possible. Plusieurs méthodes de convergence ont été proposées : convergence immédiate, linéaire, accélérée [DG03]...

2.2.2 Le modèle explicite de comportement

Le modèle explicite de comportement est une amélioration des méthodes de prédiction qui l’ont précédé, il a été créé pour WAVES [Kaz93]. Dans ce modèle, les objets possèdent un modèle explicite de comportement (des entités autonomes qui encapsulent leurs états). Chaque objet est caractérisé par un ensemble d’attributs qui définissent son état et un ensemble de comportements exécutables qui peuvent mettre à jour cet état. Tant que le comportement de l’objet ne dévie pas significativement de ce que le modèle de comportement prédit, il n’y aura pas une nécessité d’envoi de mises à jour.

Le modèle explicite de comportement présente un avantage majeur par rapport aux autres méthodes de prédiction : Les autres méthodes sont concentrées sur les approximations du comportement du premier et du second ordre alors que le modèle explicite de comportement essaie

de reconnaître et d'identifier les actions et de prédire les états futurs selon cette identification. Les techniques de prédiction du premier ordre continuent à simuler l'action qui s'est déroulée d'une façon linéaire. Les techniques du second ordre prédisent les états suivants en se basant sur des comparaisons. Le modèle explicite de comportement se base sur la reconnaissance et l'identification des actions. Ainsi si le comportement de l'entité est plus ou moins régulier, la fréquence d'émission des mises à jour sera assez réduite. Par contre, quand le comportement simulé est complexe et imprédictible, des corrections fréquentes du comportement seront envoyées. Mais quoiqu'il en soit, le modèle de comportement reste meilleur que les deux autres techniques.

Inconvénients du modèle :

Le succès des méthodes de prédiction est directement lié à la possibilité de prédiction des comportements des entités simulées. Dans le cas de comportements non prédictibles, les méthodes de prédiction ne réussissent pas à réduire le montant de communication et d'échange de données nécessaires et à accommoder la latence du système.

2.3 La gestion d'intérêt

Les méthodes de gestion d'intérêt identifient les intérêts de chaque utilisateur et leur envoie uniquement les messages susceptibles de les intéresser. Les limitations perceptuelles et cognitives humaines présentent une base réaliste pour le filtrage de messages : aucun utilisateur ne peut percevoir et par suite ne peut-être intéressé par le monde virtuel en entier. Ceci est dû à plusieurs raisons : la conception du monde (chambres, murs, obstacles, etc.), les distances qui séparent les avatars, la conception des personnages des avatars et leurs limitations de conscience, etc. Tous ces facteurs définissent les intérêts des utilisateurs à l'intérieur du monde virtuel et permettent un filtrage dont le concept est de ne transmettre aux utilisateurs que les données qui les intéressent. Ce filtrage s'appelle la gestion d'intérêt.

Macedonia et al. [MZP+95] a classé le filtrage par gestion d'intérêt en trois catégories :

1. **Temporel** : Certaines entités n'ont pas besoin d'être au courant des changements d'états des autres entités en temps réel, elles n'exigent donc pas de recevoir des mises à jour fréquentes. Les entités ayant besoin de mises à jour dans des laps de temps identiques

appartiennent à une même classe temporelle. Un exemple d'entités de ce type est le simulateur d'un satellite espion qui a besoin d'une connaissance globale des véhicules terrestres, il accepte des mises à jour de faible fréquence et quand il a besoin d'une plus grande résolution, il focalise son attention sur une zone particulière et il change ainsi de catégorie d'intérêt pour recevoir les mises à jour qui l'intéressent en temps réel.

2. **Fonctionnel** : Une entité peut communiquer avec un sous-ensemble d'entités selon leurs fonctions. Il peut y avoir par exemple des entités capables d'entendre le trafic radio, d'autres intéressées par les entités aériennes et les événements qui se produisent au dessus du sol.

3. **Spatial** : L'espace est découpé en régions qui peuvent être traitées en parallèle et indépendamment l'une de l'autre. Les participants qui se trouvent dans la même région interagissent ensemble. Les échanges de messages se font donc entre des sous-ensembles d'entités présentes dans les mêmes régions (ou dans des régions voisines).

A l'exception de NPSNET, développé par Macedonia et al., le filtrage temporel a été rarement utilisé. Par contre, la majorité des méthodes de gestion d'intérêt utilisent des propriétés fonctionnelles et spatiales pour définir les intérêts des entités. De plus, ces deux types de propriétés sont étroitement liés. En général, les conditions fonctionnelles et spatiales doivent être toutes les deux satisfaites pour que les entités soient conscientes l'une de l'autre.

Les méthodes de gestion d'intérêt utilisent ces propriétés pour définir un filtrage basé sur la division de l'espace, sur la gestion individuelle d'intérêt ou sur un mélange des deux.

2.3.1 La division de l'espace

L'environnement virtuel est découpé en régions. Celles-ci peuvent être statiques ou dynamiques selon la conception de l'environnement et de la méthode. Les caractéristiques des entités et leur distribution dans les régions décident de l'échange de messages entre elles. Par exemple, les messages audio de deux participants distants ne sont pas échangés parce que ces deux entités ne peuvent naturellement pas s'entendre ; les mises à jour de position d'une entité dans une chambre voisine ne sont pas intéressantes non plus...

Beaucoup de méthodes qui adoptent la division de l'espace utilisent aussi le multicast en associant à chaque région un groupe multicast. Cette association est naturelle, efficace et facile

à gérer.

On verra les méthodes suivantes de division de l'espace : les locaux, le précalcul de visibilité et les shards.

2.3.1.1 Les locaux

Le concept des locaux a été introduit par SPLINE (Scalable Platform for Large Interactive Networked Environments) [BWA96]. Un local est une unité de l'environnement qui représente une région de forme et de taille arbitraires ayant son propre système de coordonnées. Les locaux peuvent représenter des chambres, des corridors, des lieux ouverts, tous concaténés pour constituer l'environnement. Le passage d'un local à un autre se fait à travers des **liens**. Ces liens permettent la transformation entre les systèmes de coordonnées correspondants. L'interaction et la conscience d'une entité au sein d'un local est limitée à son local et aux locaux voisins. A chaque local sont associés des groupes multicast qui correspondent aux types de données échangées (audio, visuel, mouvement). Un utilisateur s'abonne donc aux groupes multicast de son local et des locaux voisins.

SPLINE a adopté au début une architecture égal-à-égal. Ensuite, il a utilisé une architecture hybride dont le ou les serveurs prennent en charge la communication avec les utilisateurs ayant une mauvaise connexion et fournissent aux nouveaux utilisateurs l'état des locaux.

MASSIVE-3 [GPS00, PG00] a repris les locaux en y introduisant un aspect non-spatial à travers de nouveaux concepts comme les frontières, les aspects, les abstractions et les politiques. Les locaux de MASSIVE-3 utilisent une architecture client/serveur.

Les **frontières** sont une évolution des liens des locaux de SPLINE. Elles possèdent des propriétés supplémentaires comme la destination, le polygone à travers lequel l'utilisateur peut traverser, le type de la destination (adjacente ou non au local), la conscience (l'effet de la frontière sur la conscience).

Un local présente en plusieurs **aspects**. Chaque aspect est caractérisé par :

- une ou plusieurs classes fonctionnelles (qui peuvent correspondre aux différents média) ;
- les organisations des objets, des participants... ;

– une fidélité (correspondant à la résolution, la qualité...) ;

– et un coût (correspondant à la taille, la bande passante, le rendu graphique...).

Un local doit obligatoirement avoir un aspect appelé BASE qui contient les liens vers tous les aspects du local et vers les locaux voisins. La réplication de tous les aspects BASE permet de connaître la topologie du monde.

Il peut y avoir des locaux qui ont les mêmes classes fonctionnelles et organisationnelles mais des fidélités différentes. Les locaux ayant de basses fidélités s'appellent **abstractions**. Les abstractions ont été créées pour répondre aux besoins des utilisateurs qui sont moyennement intéressés par un local ou aux besoins des utilisateurs qui n'ont pas les ressources qui supporte une fidélité élevée. Un utilisateur choisit donc une abstraction selon ses intérêts et ses capacités.

Alors que SPLINE permet uniquement à un utilisateur d'être conscient de son local et de ses locaux voisins, MASSIVE-3 introduit des **politiques** qui permettent d'identifier les locaux qui sont les plus importants par rapport à un local. Plusieurs politiques ont été définies dans MASSIVE-3. Une application peut combiner plusieurs politiques à la fois. Voici quelques exemples de politiques standard :

– **Les voisins de niveau N**. Cette politique permet à un local de voir les locaux dont le niveau de voisinage est inférieur ou égal à N. Cette politique dépend donc de la distance topologique qui sépare les locaux. Dans la figure 2.2, la politique prend les voisins de niveaux 0 et 1 du local courant ;

– **Les M locaux les plus proches**. Augmenter le niveau dans la politique précédente peut rajouter un grand nombre de locaux. Pour cette raison, une politique prenant les M locaux les plus proches de l'utilisateur permet de réduire le nombre de locaux rajoutés. Cette politique peut aussi être appliquée indépendamment de la politique précédente. Dans la figure 2.3, la politique prend les 6 voisins les plus proches de l'utilisateur ;

– **Les P locaux les plus conscients**. Les deux politiques précédentes se basent sur la topologie des locaux et sur la distance. Cette politique garde les P locaux les plus conscients du local courant. Les critères de conscience sont déterminés selon les besoins de l'application.

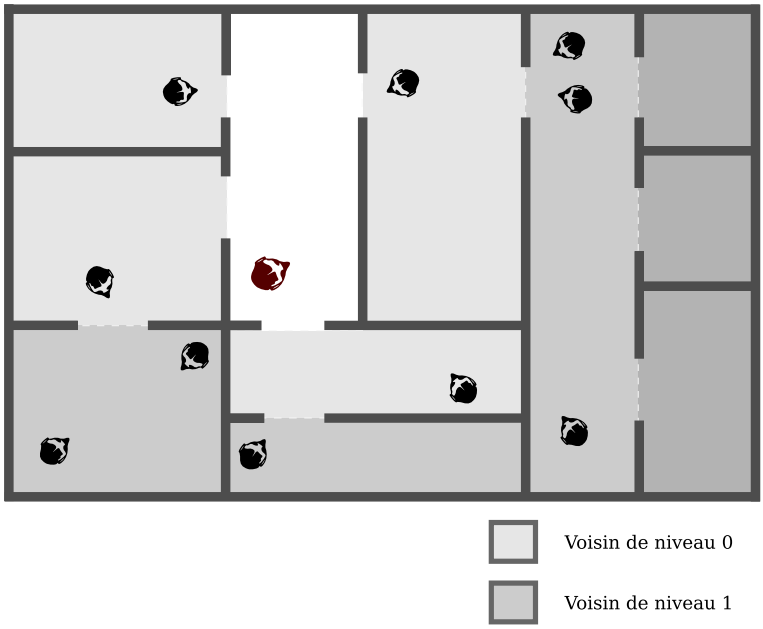

FIG. 2.2 – La politique des voisins de niveau inférieur ou égal à N

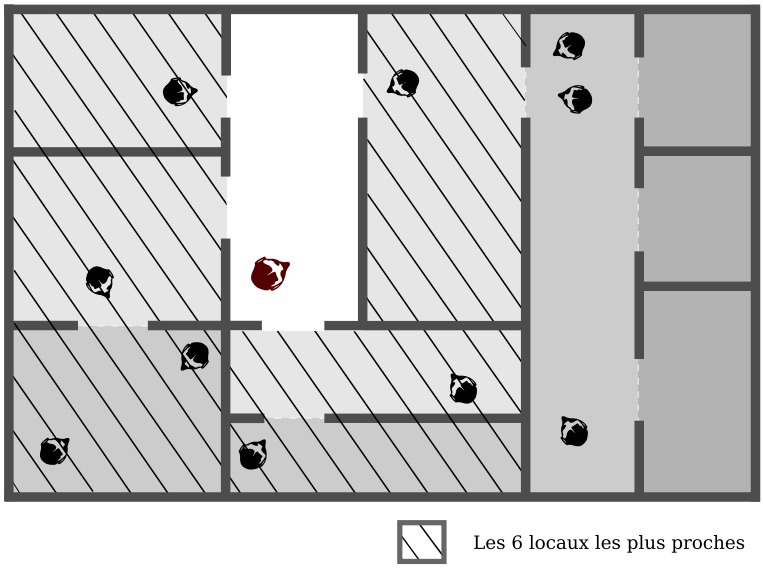

FIG. 2.3 – La politique des M locaux les plus proches

Avantages des locaux :

- Les locaux permettent la création de structures non-euclidiennes (des miroirs virtuels, des trous...)
- Au delà de leur division spatiale, les locaux divisent l'environnement en aspects selon des caractéristiques fonctionnelles et organisationnelles. Ceci permet un regroupement efficace des objets et des entités.
- Le choix des abstractions et des politiques permet à un utilisateur d'adapter ses intérêts et ses besoins à ses ressources.

Inconvénients des locaux :

- Les locaux sont appropriés aux architectures emboîtées (comme les bâtiments, les chambres...) mais ils ne sont pas adaptés aux architectures plus ouvertes qui permettent une vue plus étendue que celle des voisins directs.
- Les locaux sont incapables de différencier entre les différentes perméabilités des média. Ils associent à un nombre de groupes multicast correspondants aux différents média qu'ils supportent mais un utilisateur joint tous les groupes du local et de ses voisins directs sans faire aucune différence. Ceci ne correspond pas au fait que le son peut se propager différemment que la lumière. Autrement dit, il est possible d'entendre sans pour autant voir ce qui se passe derrière un mur.
- La sélection de la politique de filtrage est une opération difficile. Elle devrait dépendre de la charge qu'entraîne chaque politique et de la capacité du système. Mais aucune méthode n'a été conçue pour aider à faire le choix des paramètres de la politique adoptée.

2.3.1.2 Le précalcul de visibilité

Une autre méthode utilisée pour le filtrage par division de l'espace est le précalcul de visibilité. Cette méthode a été mise en œuvre dans RING [Fun95, Fun96]. Elle est utilisée essentiellement dans la simulation des environnements à denses occlusions (les bâtiments, les villes, etc.). Le concept de base est d'envoyer les mises à jour d'une entité aux entités capables de la voir malgré les occlusions de l'environnement. Pour cela, l'environnement est divisé en cellules dont les frontières sont des polygones. Un précalcul de visibilité intercellulaire détermine pour chaque

cellule l'ensemble des cellules potentiellement visibles. Ce calcul est effectué à l'aide d'un tracé des lignes de vue à travers les frontières transparentes (portes, fenêtres, etc.).

Durant la simulation, un message de mise à jour d'une entité appartenant à une cellule est envoyé aux résidents de cette cellule et des cellules considérées comme étant capables de la voir. Par contre, ceci implique la livraison de messages inutiles parce qu'une cellule considérée visible depuis une autre cellule n'est pas forcément complètement visible, et n'est pas non plus forcément visible de n'importe où dans l'autre cellule. La visibilité d'une entité depuis l'extérieur de sa cellule dépend évidemment de sa position et des positions des entités qui sont censées la voir à travers les occlusions. Ce qui fait que le précalcul de visibilité n'est pas exact mais il permet d'économiser le coût du calcul individuel de visibilité en temps réel.

RING a proposé la mise en œuvre de son filtrage dans plusieurs architectures :

1. Les architectures égal-à-égal : Deux modes de communication ont été proposés dans cette architecture :
   - Unicast : Chaque nœud garde une liste des entités qui participent à la simulation. Il utilise alors le précalcul de visibilité et cette liste pour envoyer par unicast ses mises à jour aux entités résidentes dans des cellules qui voient sa cellule. La limitation du passage à l'échelle de cette architecture parvient de l'obligation de maintenir à jour la liste des entités et leurs répartitions dans les cellules.
   - Multicast : Un groupe multicast est assigné à chaque cellule. Quand une entité entre dans une cellule, elle joint les groupes multicast de sa cellule et des cellules visibles. De cette façon, les entités ne doivent pas maintenir les listes des entités avec leur répartition dans l'espace. Elles ne font que joindre et quitter les groupes multicast correspondants à leurs positions, ce qui est coûteux si les entités changent fréquemment de cellules.
2. Les architectures Client/Serveur :
   - Connexion statique : Un client communique avec un serveur spécifique quelle que soit sa position et son déplacement dans l'environnement au cours de la simulation. Le serveur reçoit les mises à jour de l'entité et les renvoie aux entités et aux serveurs intéressés à l'aide d'une liste qu'il met à jour régulièrement.
   - Connexion dynamique : Chaque serveur est responsable d'une cellule. Les entités changent

de serveurs lorsqu'elles changent de cellules. Un serveur ne doit donc garder que les listes des cellules qui lui sont visibles ; ce qui réduit l'échange de données entre les serveurs et permet un meilleur passage à l'échelle.

– Connexion dynamique et communication multicast : à chaque cellule est associé un groupe multicast. Les entités d'une cellule et les serveurs gérant les cellules intéressées joignent son groupe multicast. Quand un serveur reçoit une mise à jour, il la renvoie au groupe multicast de la cellule de l'entité. Les entités de la cellule et les serveurs gérant les cellules qui la voient reçoivent la mise à jour. Ces serveurs renvoient la mise à jour à leurs entités à travers leurs groupes multicast. L'avantage de cette approche est que les serveurs joignent les groupes multicast des cellules visibles au début de la simulation, il n'y a donc aucun changement d'abonnement au cours de la simulation. Par contre, les entités continuent à changer de groupes multicast au cours de leurs déplacements.

Inconvénients de cette méthode :

– L'approche correspond aux environnements à occlusions denses mais elle ne peut pas s'adapter aux environnements ouverts.
– Cette méthode ne fait pas de différence entre les différents média. En particulier, son précalcul de visibilité peut ne pas être valable pour l'audio (le son peut être perméable à travers les murs).
– Le manque de précision dans le pré-calcul de visibilité cause une réception de messages inutiles.
– Le système peut devenir surchargé si une cellule est surpeuplée.

2.3.1.3 Les "shards"

Les jeux en ligne, surtout les jeux massivement multi-joueurs, divisent l'espace en régions gérées chacune par un serveur. Un joueur communique avec le serveur de sa région et quand il se déplace d'une région à une autre, il change de serveur. Dans ces jeux, chaque région a une limite de nombre de joueurs simultanés, et quand la limite est atteinte, l'entrée de nouveaux joueurs dans la région n'est possible que lorsqu'un joueur courant quitte la région. Ceci pose une limite assez contraignante pour le jeu. Pour résoudre ce problème Sony présenta le concept des

"shards" pour Everquest. Une **shard** [YC06] est une instance dupliquée du monde virtuel entier. En répliquant les "shards", un jeu pourra théoriquement servir un nombre illimité de joueurs. Beaucoup de MMOGs comme Ultima Online, Silkroad Online et World of Warcraft adoptent la division de l'espace en régions (éventuellement dynamiques) et les "shards" comme solution pour le passage à l'échelle.

Inconvénients des "shards" :

Vu que les shards sont des copies parallèles du même environnement simulé, elles entraînent des problèmes majeurs de réalisme :

- Il n'y a pas de cohérence entre les états du monde chez tous les participants. La cohérence est limitée au sein d'une même shard du monde.
- Les joueurs ont une possibilité d'interaction assez limitée entre eux. Deux joueurs dans deux zones différentes ont déjà une probabilité d'interaction assez faible d'interagir ensemble mais deux joueurs dans des shards différentes n'ont aucune chance de pouvoir le faire.

2.3.2 La gestion individuelle de l'intérêt

La deuxième catégorie des méthodes de gestion d'intérêt est la gestion individuelle de l'intérêt. Les méthodes de gestion individuelle d'intérêt permettent aux entités de définir leurs intérêts individuels implicitement ou explicitement selon leurs fonctionnalités et selon les propriétés de l'environnement. Ainsi, le filtrage des données se base sur les intérêts individuels de chaque entité. Cette gestion est donc plus précise que la division de l'espace mais elle a un coût plus élevé. Nous verrons parmi les méthodes de gestion individuelle de l'intérêt : la gestion des déclarations, la gestion de la distribution des données, le modèle spatial d'interaction, la gestion prédictive d'intérêt, les sphères étendues et les groupes représentatifs.

2.3.2.1 La gestion des déclarations

La gestion des déclarations (DM - Declaration Management) est un des services de filtrage de HLA. HLA ou High Level Architecture [CW96, MZ01, HRC96] est un système de simulation développé pour le département américain de la défense, il est devenu le standard IEEE 1516

en 1992. HLA divise les simulations (appelées fédérations) en un nombre de fédérés distribués. Les fédérés peuvent être des modèles de simulation, des collecteurs de données, des simulateurs, des agents autonomes ou des observateurs passifs. Les fédérés déclarent, avec le service DM, les types d'entités qu'ils gèrent (et donc pour lesquels ils vont envoyer des informations) et les types d'entités qui les intéressent (et donc pour lesquels ils vont recevoir des informations). Dans un premier temps, un fédéré s'abonne à une classe d'objet. Par la suite, le système notifie le fédéré chaque fois qu'un objet de cette classe est mis à jour par un fédéré qui s'est préalablement déclaré comme publiant cette classe d'objet.

Inconvénients du service :

- Les services de la gestion de déclarations ne sont pas suffisants pour les simulations à grande échelle.
- Ces services ne présentent pas de précision vis-à-vis de l'intérêt des entités. On peut le considérer comme étant un filtrage fonctionnel qui ne prend pas en compte la localisation spatiale de l'intérêt car il ne travaille que sur les classes d'objets et non pas sur les instances.

2.3.2.2 La gestion de la distribution des données

La gestion de la distribution des données ou DDM (Data Distribution Management) [BR00] est le deuxième service qui assure le filtrage dans HLA. Les services de la DDM permettent de fournir des informations additionnelles en plus de celles fournies à la DM (Declaration Management - cf. 2.3.2.1) pour exécuter un filtrage plus précis et plus efficace. La DDM gère en plus des types de données, les instances spécifiques de ces données. La DDM effectue son travail en trois phases :

1. L'expression de l'intérêt et de la disponibilité ;
2. Le calcul d'intersections ;
3. La livraison et la réception des données.

Expression de l'intérêt et de la disponibilité

les entités spécifient les données qu'elles veulent recevoir et envoyer. Elles peuvent exprimer leurs intérêts (**régions d'abonnement**) et leurs disponibilités (**régions de mise à jour**) statiquement (au début de la simulation) ou dynamiquement (plusieurs fois au cours de la simulation). La région d'abonnement d'un fédéré est l'union des régions d'abonnement de toutes ses entités. De même, la région de mise à jour d'un fédéré est l'union des régions de mise à jour de ses entités.

Calcul d'intersections

Le but de cette phase est de trouver les intersections entre les régions d'abonnement et les régions de mise à jour pour optimiser le transfert des données dans la phase suivante (livraison et réception de données). Comme les informations concernant les régions d'abonnement et de mise à jour sont distribuées, la DDM a besoin d'une infrastructure qui puisse gérer ces régions d'une manière efficace pour assurer le passage à l'échelle. Pour cette raison, plusieurs systèmes ont été créés :

- **La grille de routage :** c'est un système de coordonnées multidimensionnel où les entités expriment leurs intérêts et leurs disponibilités. Une dimension de la grille pourrait exprimer la longueur ou la largeur du monde ou une combinaison des deux, elle pourrait aussi correspondre à la fréquence radio ou à l'ensemble des adresses IP...

  Le calcul d'intersection se fait au niveau des cellules, si une cellule est partagée par une région de mise à jour et au moins une région d'abonnement, une correspondance est détectée.

  Une simulation peut avoir plusieurs grilles. Pour chaque grille on doit connaître :

  1. le nombre de dimensions ;
  2. la variable de routage correspondant à chaque dimension ;
  3. la mise en correspondance qui transforme les variables de routage en des coordonnées ;
  4. la taille initiale d'une cellule pour chaque dimension. En fait, les cellules peuvent être de taille fixe ou variable. L'utilisation de cellules de taille variable fournit une meilleure

précision du filtrage. En effet, si une région de mise à jour et une région d'abonnement chevauchent une cellule sans se chevaucher entre elles (la cellule 5 de la figure 2.4), elles vont donner lieu à une correspondance malgré leur non chevauchement. Ce qui entraîne une réception de données inutiles, d'où l'intérêt d'utiliser des cellules de taille variable.

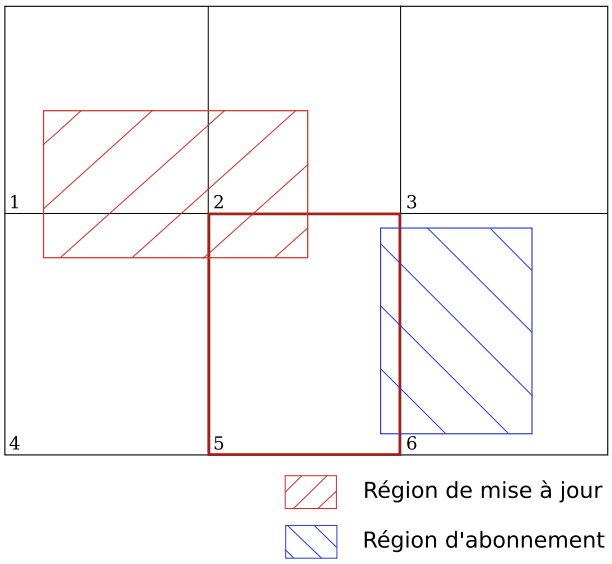

FIG. 2.4 – Calcul de l'intersection

Des grilles hiérarchiques ont aussi été utilisées [BR00]. Ces grilles consistent en plusieurs niveaux de grilles, chacune ayant une taille de cellule différente, la meilleure résolution étant bien sûr la plus basse. En général, les grilles hiérarchiques sont représentées par des quadtrees ou des octrees.

– **L'espace de routage :** il s'agit d'une grille sans cellule, c'est-à-dire un espace multidimensionnel sans partition.

  Un espace de routage possède les propriétés suivantes : le nombre de dimensions, les variables de routage et la mise en correspondance. En revanche, la taille initiale des cellules n'est pas présente dû à l'absence de partitions dans l'espace de routage.

  Comme dans la grille, une simulation peut avoir plusieurs espaces de routage ayant des caractéristiques différentes et utilisés dans des buts différents. Chaque fédéré détermine lesquels parmi les espaces de routage définis lui sont utiles.

– **Les régions :** ce sont des sous-ensembles d'une grille particulière ou d'un espace de

routage, elles ont un nombre de dimensions plus petit ou égal à celui de la grille ou de l'espace de routage. Ce nombre dépend des intérêts et des disponibilités des entités.

Livraison et réception des données

Après la phase d'expression de l'intérêt et de la disponibilité et celle du calcul d'intersection, la dernière phase est la livraison et la réception des données. Cette phase consiste à établir la connectivité entre les éditeurs (les fédérés qui publient des informations) et les abonnés (les fédérés qui souhaitent recevoir des informations) et ensuite à transférer les données à travers la connectivité établie, qu'elle soit en unicast ou multicast.

Au cas où le multicast est utilisé, les groupes multicast peuvent être associés :

- **Aux éditeurs :** On associe à chaque éditeur un groupe multicast.
- **Aux cellules :** Dans un environnement qui utilise les grilles, on peut associer à chaque cellule un groupe multicast. Aucune détection d'intersection entre les régions de mise à jour et les régions d'abonnement n'est effectuée : quand un éditeur a des régions de mise à jour qui chevauchent une cellule, il envoie ses données sur le groupe multicast de cette cellule. Et quand un abonné est intéressé par une cellule ou une partie d'une cellule, il joint son groupe multicast pour recevoir ses messages. Bien sûr, ceci engendre le risque de réception de messages non intéressants.
- **Aux intersections :** On associe des groupes multicast aux cellules où une intersection a été détectée.
- **Aux régions :** On associe des groupes multicast aux régions de mise à jour où une intersection a été détectée.
- **Au regroupement :** On peut utiliser un algorithme de regroupement qui regroupe les régions pour optimiser l'attribution des groupes multicast.

Des approches hybrides [BMDL05, BM05] qui combinent plusieurs procédures d'attribution des groupes multicast ont été également présentés.

Inconvénients du service

1. Les fédérés doivent se mettre d'accord au début de la simulation sur les dimensions des

espaces de routage ou des grilles. Il est impossible de spécifier une nouvelle dimension au cours de la simulation.

2. Seul le chevauchement des régions d'abonnement et de mise à jour est déterminé. Les fédérés abonnés recevront donc des mises à jour d'objets se situant en dehors de leur région d'abonnement mais à l'intérieur de la région de mise à jour chevauchante (figure 2.5).

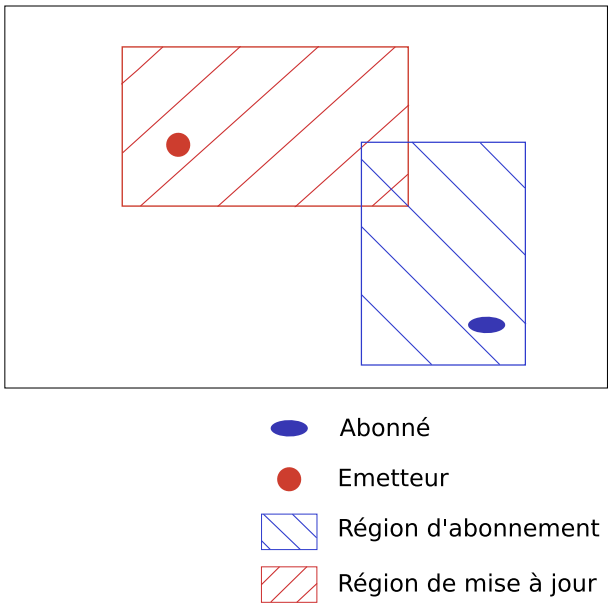

FIG. 2.5 – Réception de mises à jour non intéressantes

3. Il n'y a aucun contrôle sur la résolution des valeurs des attributs dans les régions. Une fois que le test de chevauchement a laissé passer une mise à jour d'un attribut, toutes les valeurs sont envoyées à l'abonné. Il n'a pas de contrôle sur le montant de changement d'un attribut avant qu'il ne soit traité chez lui.

4. S'il y a un grand nombre de dimensions, d'espaces de routage ou de grilles, le nombre de groupes multicast nécessaires sera élevé, ce qui entraînera pour les fédérés de nombreux et coûteux changements de groupes multicast.

Inconvénients des grilles

1. Les régions peuvent chevaucher la même cellule sans qu'elles ne se chevauchent entre elles, ce qui cause la livraison de données non pertinentes.

2. Si les régions de mise à jour associées à des objets volumineux sont très répandues, le nombre de transmissions de messages identiques sera très élevé parce que le nombre de groupes multicast est élevé. Cette duplication de transfert est due à l'ignorance, par les expéditeurs, des destinataires qui sont à l'écoute. La figure 2.6 nous montre une région de mise à jour assez répandue. Cette région de mise à jour chevauche toutes les cellules de l'environnement. Ainsi, l'entité correspondante **A** envoie ses mises à jour sur tous les groupes multicast de l'environnement. Et comme l'entité **B**, par exemple, possède une région d'abonnement qui chevauche quatre régions, **B** recevra les mêmes messages de **A** à travers les quatre groupes multicast auxquels elle est abonnée.

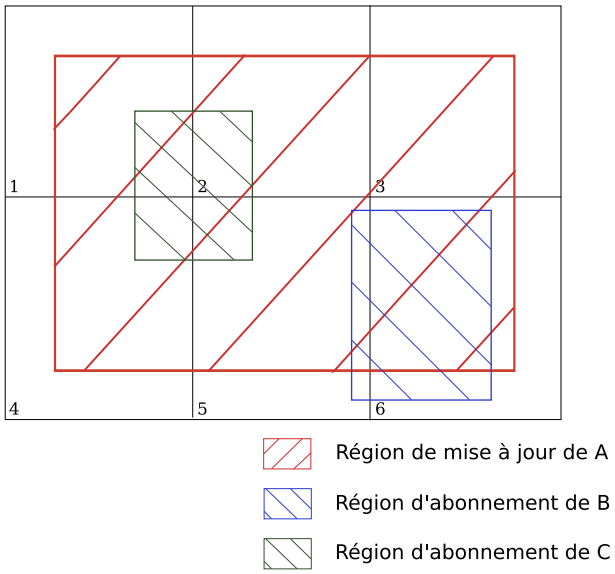

FIG. 2.6 – Duplication de transmission de messages

3. La taille des cellules dans la grille pose des problèmes. Une taille élevée produit de grands groupes multicast et donc beaucoup de données non pertinentes alors qu'une taille réduite produit des groupes plus petits mais des changements plus fréquents de ces groupes.

Le choix entre la gestion de déclarations (DM) et la gestion de distribution de données (DDM) dépend de la taille et des caractéristiques de l'application. La DM est appropriée pour les applications simples. Alors que la DDM a réussi à réduire la quantité de données reçues par les fédérés même dans des applications à grande échelle.

2.3.2.3 Le modèle spatial d'interaction

Le modèle spatial d'interaction est une méthode de gestion individuelle d'intérêt qui a été développé à l'origine au sein du projet COMIC [BF93]. Puis il a été mis en œuvre dans DIVE [Gre94] et a été le concept de base dans la mise en œuvre de MASSIVE (Model, Architecture and System for Spatial Interaction in Virtual Envrironments) [GB95].

Le modèle spatial d'interaction détermine la conscience d'une entité en étudiant ses relations bilatérales avec chacune des autres entités. Ce modèle est dédié à la téléconférence et aux réunions limitées. Dans ce modèle, les entités interagissent entre elles à travers des médias. Un **média** peut représenter un média de communication (audio, visuel ou texte) ou un autre type d'interface spécifique aux entités. Chaque entité peut interagir à travers une combinaison de médias, les entités négocient leurs médias compatibles quand elles se croisent dans l'espace.

Fonctionnement du modèle

Le modèle spatial d'interaction procède en deux étapes :

1. **Détermination de la possibilité d'interaction** : le modèle spatial définit pour chaque entité une aura dans chaque média. Une **aura** délimite une partie de l'espace au sein de laquelle une interaction avec une autre entité peut avoir lieu. Une aura peut avoir n'importe quelle forme et quelle taille. Elle peut aussi ne pas entourer l'objet qu'elle représente et ne pas être contiguë dans l'espace.

   Les entités portent leurs auras avec elles lorsqu'elles se déplacent. Quand deux auras dans le même média entrent en collision, l'interaction entre les deux entités devient possible dans ce média. Un processus surveille toutes les auras des entités et notifie les entités quand une interaction devient possible ou arrête de l'être, c'est ce qu'on appelle le gestionnaire d'auras.

2. **Contrôle de l'interaction** : après la notification de collision d'auras, les entités se mettent en contact l'une avec l'autre à travers un échange des identificateurs des objets, des adresses, des références... Ensuite, la création, la maintenance, la terminaison des connections et des communications inter-entités sont la responsabilité des entités.

Pour gérer les interactions, les entités négocient leurs niveaux de conscience mutuelle à travers leurs *foci* et leurs *nimbi*. Le **focus** d'une entité est un sous-espace qui décrit l'attention allouée par cette entité ; son **nimbus** est également un sous espace qui décrit sa manifestation ou son observabilité à travers sa présence, son identité, son activité ou une combinaison des trois. Le nimbus est donc la réciproque nécessaire du focus, exigée pour accomplir un équilibre dans les interactions.

> "La conscience qu'a une entité A d'une entité B dans un média M est une fonction du focus de A et du nimbus de B."

Autrement dit :

- plus une entité est dans mon focus, plus je suis consciente d'elle.
- plus une entité est dans mon nimbus, plus elle est consciente de moi.

L'utilisation des consciences mutuelles permet d'établir des relations asymétriques entre les entités. Cette asymétrie est présente dans la possibilité d'existence d'une conscience dans un sens mais pas dans l'autre et dans la possibilité d'avoir des niveaux mutuels de conscience inégaux.

La figure 2.7 illustre trois scénarios de niveaux de conscience. Le focus d'une entité est représenté par une forme hexagonale et son nimbus par une forme elliptique. Les trois cas illustrés sont :

- **Scénario 1** : L'entité A (resp. B) est totalement consciente de l'entité B (resp. A) et donc A (resp. B) reçoit les messages de B (resp. A).
- **Scénario 2** : A n'est pas consciente de B alors elle ne reçoit pas ses messages. Par contre, B est consciente de A.
- **Scénario 3** : A est semi consciente de B alors que B est totalement consciente de A.

Les niveaux de conscience résultants sont utilisés pour :

- activer l'interaction. Ils permettent aux entités d'être conscientes l'une de l'autre à partir d'un seuil bien spécifié du niveau de conscience.
- contrôler les médias comme le volume du canal audio, le niveau de détail du rendu graphique... En général, plus d'attention donc plus de ressources seront dévouées aux

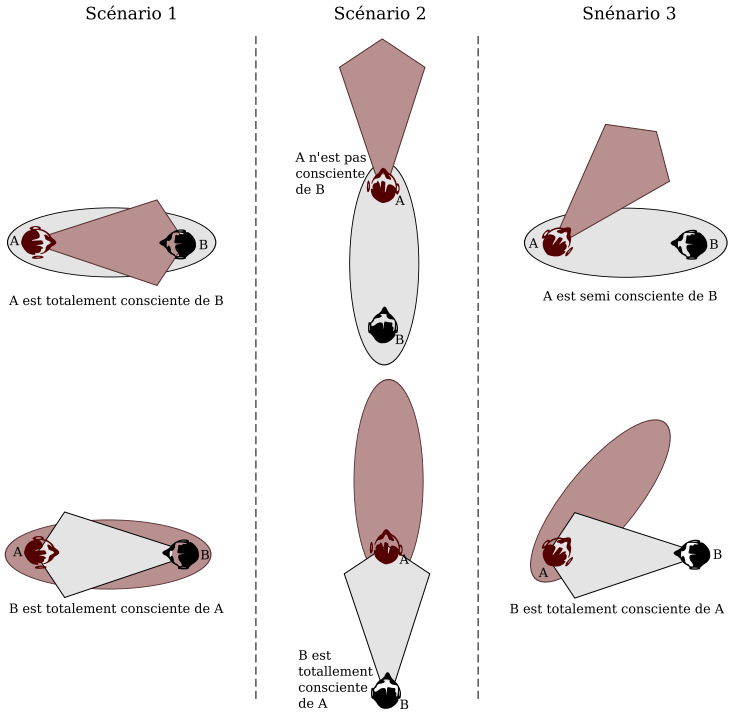

FIG. 2.7 – Les relations de conscience symétriques et asymétriques

entités ayant des niveaux de conscience élevées.

Manipulation des niveaux de conscience

Les entités peuvent manipuler leurs niveaux de conscience pour rendre les autres entités plus ou moins conscientes d'elles ou pour devenir plus ou moins conscientes des autres. Ceci permet d'une part aux entités d'être autonomes et d'autre part d'équilibrer les interactions. En fait, l'utilisateur peut gérer son aura, son focus et son nimbus et donc contrôler ses interactions :

- d'une manière très naturelle : en se déplaçant, en changeant d'orientation...
- à l'aide d'interfaces spécialisées ou d'objets **adaptateurs** variés comme un mégaphone, des jumelles ou une table de conférence. Ces objets permettent de modifier la concentration, la manifestation...

Le modèle spatial d'interaction a été mis en œuvre dans MASSIVE-1 qui utilise une architecture égal-à-égal pour établir les communications entre utilisateurs et un serveur qui joue le rôle du gestionnaire d'auras.

Avantages du modèle :

- Le modèle spatial permet d'établir des relations asymétriques entre les entités, ce qui est réaliste et naturel.
- Il permet aussi d'établir un filtrage différent selon les différents média (une entité peut entendre une autre sans la voir ou l'inverse).
- Les concepts du focus et du nimbus rendent tous les participants à une interaction capables d'influencer cette interaction quel que soit leurs rôles. Il y a un équilibre entre les auditeurs et les orateurs dans une conversation et entre les observateurs et les observés dans une scène.

Inconvénients du modèle :

- Le passage à l'échelle du gestionnaire d'auras est critique car il est le seul processus qui fait la détection de collisions pour toute l'application.
- Ce modèle supporte mal le passage à l'échelle dans les environnements densément peuplés. Le problème est dû à l'utilisation exclusive des interactions bilatérales entre les objets, d'où la difficulté de la gestion des foules. En effet, quand une entité est consciente d'un grand nombre d'entités, elle doit maintenir des communications séparées avec chacune d'elles et effectuer des calculs fréquents de leurs niveaux de conscience, ce qui est assez coûteux.
- L'asymétrie des relations entre les entités ne peut avoir lieu sans le calcul de niveaux de conscience mutuels (à travers les foci et les nimbi). Ce calcul est assez lourd surtout dans les environnements peuplés. D'autre part l'utilisation d'une version simplifiée du modèle spatial, qui n'utilise que les auras, ne permet pas d'établir des relations asymétriques entre les entités.
- Le modèle manque de considération pour les facteurs environnementaux dans le calcul de la conscience (ceci inclut les effets des régions fermées telles que des salles et des bâtiments).

2.3.2.4 La gestion prédictive d'intérêt

La gestion prédictive d'intérêt [ML03] reprend les auras du modèle spatial d'interaction. Elle essaye de résoudre le problème d'interaction ratée qui peut se produire au sein du modèle spatial et fournit en même temps une détection de collision d'auras moins coûteuse que celle du modèle

initial.

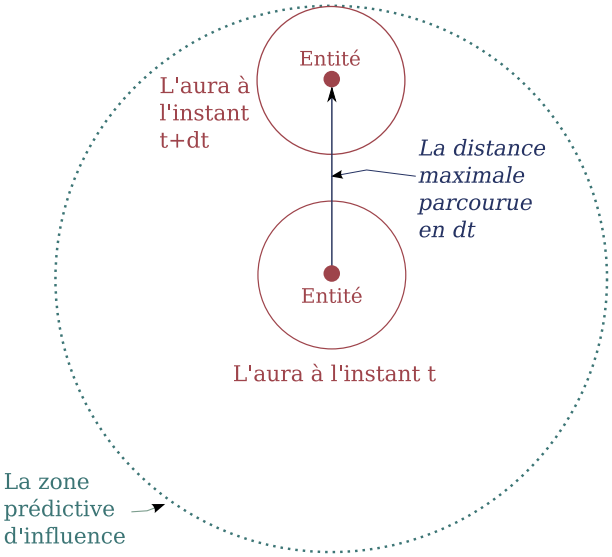

FIG. 2.8 – Zone d'influence prédictive

Si la fréquence d'échange de messages est basse, les entités peuvent ne pas être conscientes d'une collision d'auras au cas où cette collision n'aurait pas duré assez longtemps pour provoquer une notification de collision. Pour résoudre ce problème, au lieu d'augmenter la fréquence des messages et d'encombrer le système avec beaucoup de messages inutiles, la gestion prédictive d'intérêt associe à chaque entité une zone d'influence prédictive (Predictive Area of Influence : PAI). Cette zone est circulaire, centrée sur l'entité et identifie l'étendue de l'aura pendant une période de temps *dt* (Figure 2.8). Son rayon est égal à la somme de la distance rectiligne maximale que peut parcourir l'entité en un temps *dt* et du rayon de l'aura. Si les zones d'influence de deux entités chevauchent, il y aura une possibilité que ces entités deviennent conscientes l'une de l'autre dans le futur proche. Sans ce chevauchement, la collision d'auras ne peut pas avoir lieu dans un temps inférieur à *dt*. Dès lors la détection de collision d'auras est remplacée par la détection de collision de zones d'influence.

Cette approche est moins coûteuse grâce à la possibilité de diminuer la fréquence d'envoi de messages : au lieu d'envoyer fréquemment les messages de mise à jour de son aura, une entité peut envoyer, moins fréquemment, des messages de mise à jour de sa zone d'influence. Lorsqu'une détection de collision de sa zone d'influence a lieu, l'entité commence à envoyer des messages de mise à jour plus fréquents de son aura.

La gestion prédictive d'intérêt élimine ainsi le problème d'interaction ratée grâce à l'augmentation de la fréquence d'échange de mises à jour lors d'une collision de zones d'influence.

2.3.2.5 Les sphères étendues

L'approche des sphères étendues [MSL04, SLM04] utilise aussi les auras du modèle spatial d'interaction. Elle a pour but d'alléger la charge de la détection de collision entre les auras.

Pour alléger la charge, l'approche minimise le nombre de paires de comparaisons entre les auras. Elle rassemble les entités dont les auras chevauchent dans des ensembles appelés **relations de collision** (RC : Figure 2.9). L'aura de chaque entité d'une RC chevauche toutes les auras de la RC. La RC est représentée par une sphère étendue qui englobe toutes les auras de ses entités.

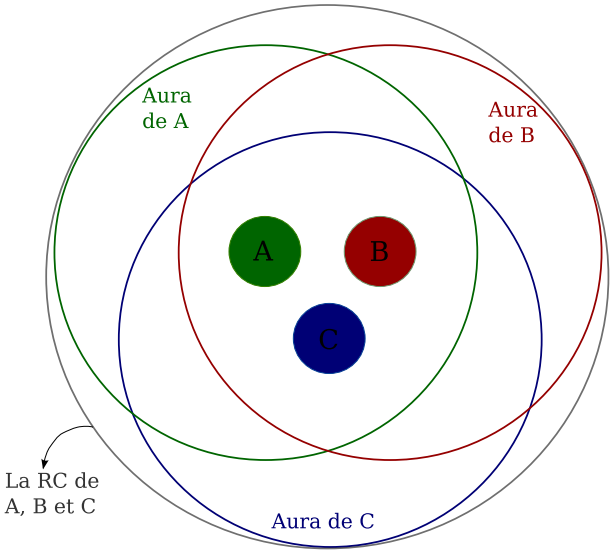

FIG. 2.9 – Sphère étendue

La formation des RCs permet d'économiser le nombre de comparaisons de détection de collision parce que si l'aura d'une entité ne chevauche pas une RC, elle ne chevauchera aucune auras appartenant à cette RC. Donc les tests individuels de collision entre l'aura de l'entité et les éléments de la RC sont inutiles, ils sont remplacés par un seul test de collision.

D'autre part, la création des relations de collision fournit une régionalisation des entités qui peut être utilisée pour l'attribution de groupes multicast.

Dans l'ensemble, cette approche permet une détection de collision à un coût moins élevé que le modèle spatial lui même.

Limitations de cette méthode :

L'attribution des groupes multicast selon les RCs n'est pas toujours efficace. En effet, il se peut qu'il y ait des entités appartenant à plusieurs RCs à la fois. Dans ce cas, elles envoient et reçoivent les mêmes messages plusieurs fois à travers de groupes multicast différents. Ce qui cause un envoi, un transfert et une réception répétitifs des données.

2.3.2.6 Les groupes représentatifs

L'approche des groupes représentatifs [HLL00] se base sur le principe de rassemblement des entités ayant les mêmes intérêts. Elle cherche à combiner les concepts d'intérêt et de proximité et à créer un transfert de données de fidélités différentes.

Chaque entité déclare ses intérêts et les associe à une zone d'intérêt circulaire, centrée sur elle. Au cours de la simulation, l'entité reçoit les messages des entités qui se trouvent à l'intérieur de sa zone d'intérêt. Si ces entités sont classées intéressantes par notre entité, elle recevra leurs informations en haute fidélité sinon elle les recevra en basse fidélité.

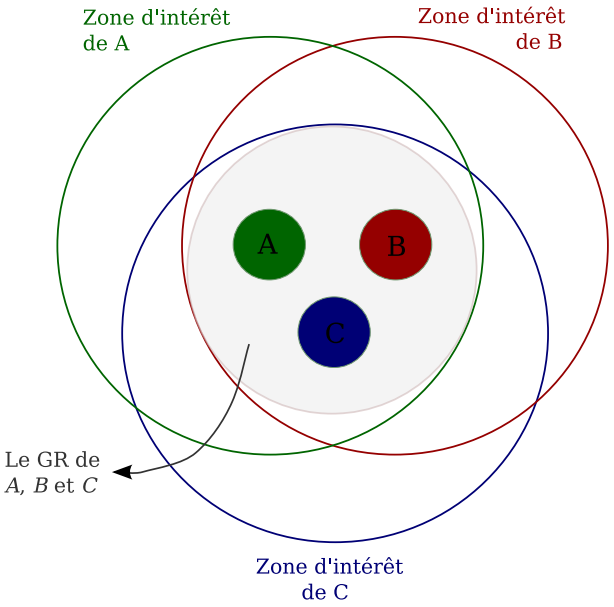

FIG. 2.10 – Groupe représentatif

D'autre part, quand deux entités ayant les mêmes intérêts se trouvent dans leurs zones d'intérêt respectives, elles forment un groupe, appelé **groupe représentatif** (GR) dont elles deviennent les membres. Un GR est toujours inclus dans les zones d'intérêt de ses membres (figure

2.10). Les entités d'un même groupe échangent leurs données en haute fidélité. Cet échange est utilisé pour former les données de basse fidélité qui seront envoyées aux entités peu intéressées par le groupe. Pour accomplir ceci, à la création de chaque GR, un représentant est élu. Il est chargé de l'envoi des données de basse fidélité aux entités dont les zones d'intérêt chevauchent le groupe. Cet envoi est fait à travers une adresse multicast spécialisée dans la transmission des données de basse fidélité du groupe.

De plus, une entité ne peut appartenir qu'à un seul GR pour éviter une transmission répétitive des données. Donc, l'entité appartient au GR qui correspond à ses intérêts prioritaires.

Cette approche utilise essentiellement une architecture égal-à-égal, les clients gèrent le filtrage tout au long de la simulation. Il existe malgré tout des serveurs dont le rôle se limite à la gestion de l'adhésion des entités aux groupes.

2.3.3 Le filtrage hybride

Cette catégorie de filtrage utilise une combinaison entre la division de l'espace et la gestion individuelle d'intérêt. Plusieurs méthodes de filtrage hybride ont été présentées, nous verrons parmi elles : la gestion par zone d'intérêt, la gestion d'intérêt en trois étapes, la fonction de surface projetée et les objets tiers.

2.3.3.1 Gestion par zone d'intérêt

La gestion par zone d'intérêt [MZP+95, Mor96] détermine l'intérêt d'une entité en fonction de ses caractéristiques et de la distance qui la sépare des autres entités. Cette gestion a été mise en œuvre dans NPSNET (Naval Postgraduate School's Network Vehicle Simulator).

La gestion par zone d'intérêt divise l'espace en cellules hexagonales égales et de taille fixe, chacune de ces cellules est associée à un groupe multicast. Chaque entité a une zone d'intérêt (AOI : Area of Interest) circulaire centrée sur elle. Le rayon de la zone d'intérêt dépend des caractéristiques et des capacités de l'entité dans son environnement. Une entité est un membre actif du groupe de la cellule où elle se trouve partiellement ou totalement : elle envoie et reçoit les mises à jour au sein de ce groupe. Elle est un membre passif des groupes des cellules qui chevauchent sa zone d'intérêt sans l'inclure : elle ne peut que recevoir des mises à jour au sein de ces groupes.

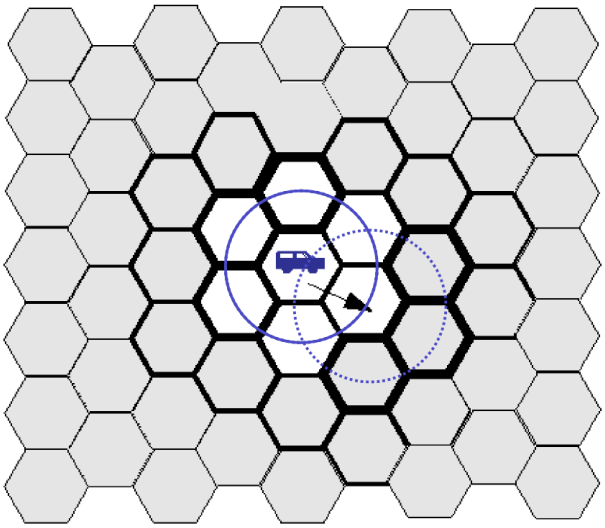

FIG. 2.11 – Division de l'espace dans NPSNET

Dans la figure 2.11, la zone d'intérêt du véhicule chevauche sept hexagones, ce véhicule est alors membre des sept groupes multicast associés à ces hexagones. Il écoute les sept groupes, mais n'envoie ses mises à jour qu'au groupe associé à la cellule dans laquelle il se trouve. En se déplaçant, le véhicule doit quitter et joindre trois groupes multicast à la périphérie de sa zone d'intérêt. Le délai de ce changement de groupe n'est pas critique puisque ces groupes sont associés aux cellules visibles les plus éloignées du véhicule.

Deux entités sont conscientes l'une de l'autre si elles peuvent communiquer ensemble. Par conséquent, l'entité A devient consciente de l'entité B si B est un membre actif d'un groupe auquel appartient A et vice versa. Si les deux entités sont des membres passifs des mêmes groupes, c'est que chacune d'elles est au delà de la vue et de l'influence de l'autre.

Limitations de cette méthode :

Cette méthode a été développée pour les simulations militaires terrestres. La taille des cellules, la division de l'espace et la distribution des entités conviennent aux besoins des véhicules militaires dans des conditions normales. Le véhicule le plus rapide changera de cellule au plus une fois par heure, ce qui est tout à fait acceptable au niveau de changement de groupes multicast. Par contre, dans les environnements plus hétérogènes, il y a forcément des limitations et des choix impossibles à faire :

- La taille des cellules : de même que pour les méthodes qui utilisent la division de l'espace, la détermination de la taille des cellules de répartition est une opération assez critique. En effet, si les cellules sont très petites par rapport à la rapidité de déplacement d'une entité, l'entité changera trop fréquemment de groupes multicast. Par contre, si les cellules sont très larges, elle risque de recevoir beaucoup trop d'informations d'entités qui ne l'intéressent pas.

  Si l'environnement simulé contient des entités ayant des vitesses très différentes, il n'y aura jamais une taille de cellule qui conviendra à toutes les entités. Donc, cette méthode est adaptée aux simulations dont les entités ont des vitesses homogènes mais elle l'est moins pour des entités de vitesses hétérogènes.
- La taille des cellules dépend aussi de la taille des zones d'intérêt. S'il y a une entité qui a une zone d'intérêt très répandue par rapport aux autres, elle sera membre d'un nombre considérable de groupes multicast. Lors de son déplacement, elle quittera et joindra un grand nombre de groupes multicast ce qui produira le même problème que pour les entités rapides. De plus, cette entité recevra un grand nombre de messages provenant de petites entités présentes dans sa zone mais qui ne l'intéressent pas nécessairement ; d'où la difficulté d'adapter l'application avec des entités ayants des intérêts hétérogènes.
- La division de l'espace en cellules statiques est performante si les entités sont dispersées dans l'environnement, mais si elles sont regroupées dans quelques cellules, l'approche ne sera plus efficace. Ceci revient au problème de regroupement qui survient lorsqu'une foule d'entités se retrouve dans une même région et toutes s'abonnent toutes à un même groupe multicast. Cette limitation est assez restrictive surtout dans les architectures emboîtées comme les bâtiments qui peuvent attirer un grand nombre d'entités dans une surface assez limitée.

2.3.3.2 Gestion d'intérêt en trois étapes

La gestion d'intérêt en trois étapes [AWZ98] est une amélioration de la gestion par zone d'intérêt adressant les problèmes de regroupement et de choix de la taille des régions. Elle divise l'environnement en des régions dynamiques de tailles différentes dans le but d'équilibrer la distribution des entités dans les régions et par suite dans les groupes multicast. Elle offre aux

entités le choix du niveau de détail qu'elles désirent et le choix des critères de filtrage.

Première étape

Cette étape effectue un filtrage préliminaire et imparfait qui allège le coût élevé du calcul de l'intersection entre les zones d'intérêt des entités et les autres entités de l'environnement.

Ce filtrage préliminaire est effectué à travers la division dynamique de l'environnement en des régions équilibrées suivant l'activité à l'intérieur des régions. Un octree dynamique est utilisé pour équilibrer la charge des régions à travers l'équilibre du nombre d'entités par groupe multicast. Si beaucoup d'entités sont inclues dans une même région, l'octree subdivise automatiquement cette région en huit régions filles. Et si plusieurs entités quittent un sous arbre de régions, les huit régions filles seront fusionnées dans la région parente. Dans la figure 2.12 (illustrant le procédé en deux dimensions), la région 4 est surpeuplée alors elle a été subdivisée. L'entité ne considérera que les régions 1,2 et 3 et une sous-région de la région 4. Ceci va permettre d'économiser son calcul d'intérêt.

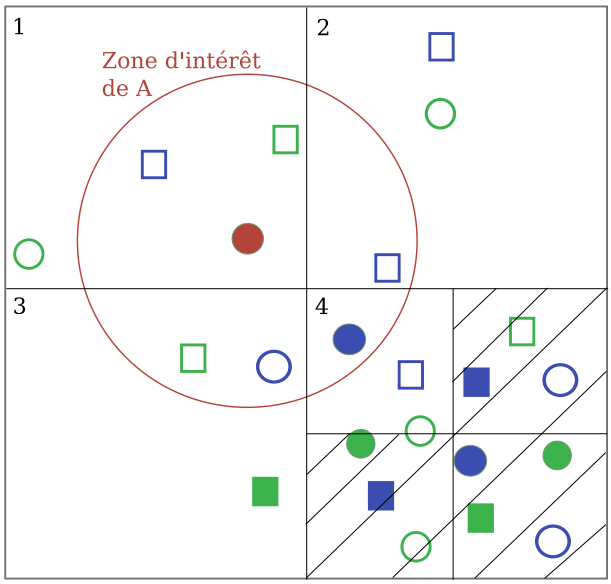

FIG. 2.12 – Première étape

L'équilibrage de charge des groupes multicast élimine le problème de regroupement et l'inconvenance d'une taille unique pour tout l'environnement. Par contre, la subdivision dynamique de l'octree crée un nouveau problème : s'il y a une très grande densité d'entités, les subdivisions

seront très nombreuses jusqu'au point où les entités risquent de changer de régions continuellement ; ce qui entraîne plus de changements de groupes multicast. Si on prend l'exemple d'un homme qui est sur une colline couverte de fourmis, à cause de la densité élevée des fourmis, l'octree va être subdivisé jusqu'à ce qu'il satisfasse un certain critère de densité d'entités. Ceci est convenable pour les fourmis, mais ne l'est pas pour l'homme. Si l'homme veut traverser la colline rapidement, il va passer par des dizaines et éventuellement des centaines de régions à chaque pas. Pour résoudre ce problème, la méthode présente le concept de la plus petite région. Une entité calcule la plus petite région qui lui est convenable suivant sa taille et sa vitesse. Quand la région où elle se trouve est divisée, l'entité vérifie si les nouvelles sous régions sont au-dessous de son exigence minimale de taille. Si c'est le cas, elle restera dans sa région courante au lieu de passer à l'une des huit feuilles.

Les entités se retrouvent alors partout dans l'octree et non seulement dans les feuilles. Elles sont distribuées non seulement selon leurs positions mais aussi selon leurs tailles et leurs vitesses. Ceci a l'avantage de réaliser un filtrage plus efficace qui permet de faire des agrégations. Si on reprend l'exemple de l'homme et des fourmis : Quand l'homme traverse la colline, s'il n'est pas intéressé par les objets de la taille d'une fourmi, il ne s'inscrit pas aux régions des fourmis. Cependant, il verra une vue agrégée des fourmis qui l'informe qu'elles sont présentes là où il passe. Mais s'il s'arrête pour examiner le sol, il pourra s'inscrire temporairement à des régions plus petites pour être capable de voir les fourmis en détail.

Donc, les informations de faible fidélité sont utilisées pour une approximation de la position et de l'orientation de l'entité. Par contre, les informations de haute fidélité concernent les entités jugées intéressantes dans la première étape et sont envoyées aux deux étapes suivantes.

En utilisant ce type de filtrage, on pourra aboutir dans les étapes suivantes aux données exactes nécessaires à une entité en ne partant que d'un sous-ensemble des données (au lieu de partir de l'ensemble total des données ou de se contenter d'une approximation).

Deuxième étape

Les entités utilisent les informations acquises lors de la première étape pour limiter la liste des candidats parmi lesquels elles choisissent les entités qui les intéressent. Les critères de filtrage

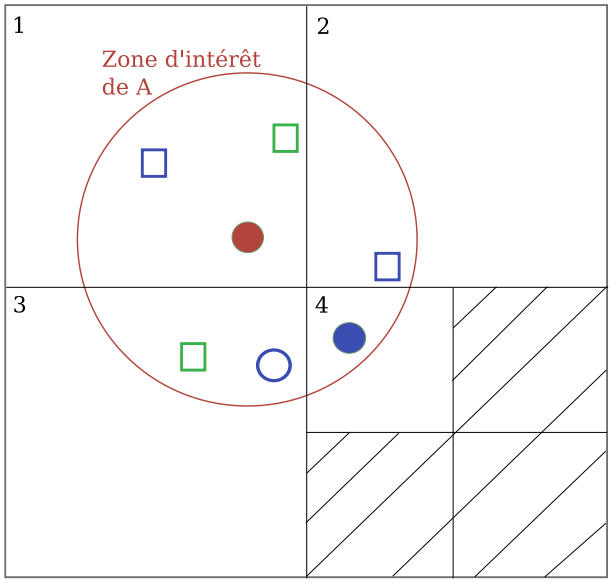

FIG. 2.13 – Deuxième étape

dans la deuxième étape ne sont pas spécifiques aux protocoles. C'est-à-dire qu'un utilisateur choisit dans cette étape les entités avec qui il veut communiquer avec sans se préoccuper des différents types de cette communication (comme les médias utilisés, les différentes fréquences d'envoi des mises à jour...). Dans notre exemple de la figure 2.13, l'entité choisit de ne garder parmi les entités de la première étape que celles qui sont à l'intérieur de sa zone d'intérêt.

La troisième étape

Cette étape rajoute une gestion d'intérêt spécifique aux protocoles. Elle permet à un utilisateur de choisir les groupes multicast de chaque entité qui l'intéresse. Ces groupes multicast peuvent correspondre aux différents média que l'entité utilise ou à des fréquences de mise à jour différentes. Dans notre exemple, l'entité décide de ne communiquer qu'avec les entités qui communiquent en audio, ces entités sont représentées par des cercles. Ainsi au terme de la troisième étape, notre entité est consciente de deux entités sur 22 (Figure 2.14).

Limitations de cette méthode :

– Cette technique exige un nombre élevé de groupes multicast parce qu'elle associe à chaque entité au moins un groupe multicast. Ceci limite énormément le passage à l'échelle de l'application.

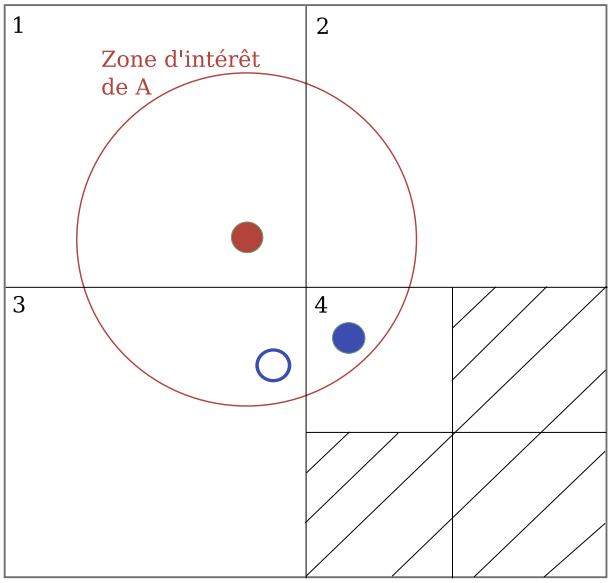

FIG. 2.14 – Troisième étape

– Il y a deux niveaux de groupes multicast dans le modèle : au niveau des régions et au niveau des entités. Le modèle n'explique pas si les entités font un choix entre ces deux niveaux ou si elles s'abonnent aux deux (dans ce cas il y aura une redondance de données).
– La gestion d'intérêt en trois étapes a été mise en œuvre partiellement. En fait, il n'y a que la première étape qui a été mise en œuvre, l'évaluation de cette méthode n'est donc pas complète.

2.3.3.3 La fonction de surface projetée

La fonction de surface projetée [FT98, Far99] détermine pour chaque entité les entités qui lui sont perceptibles en se basant sur la capacité de perception de l'entité observatrice et les positions et les tailles des entités observées. Cette fonction se sert d'une structure spatiale appelée "structure d'espace échelle" pour accomplir sa tâche.

Détermination de l'intérêt

La perception d'une entité est déterminée en projetant les sphères englobantes des entités sur un plan de projection spécifique à l'entité observatrice. Si la taille de la surface de projection d'une entité est supérieure à un certain seuil, elle est jugée perceptible. Ainsi, le coût du calcul

de la perception devient très élevé quand le nombre de participants augmente. Pour cette raison, une structure spatiale appelée "structure d'espace échelle" (SEE) a été introduite. La SEE est supportée par un octree qui lui permet de déterminer le voisinage pertinent d'une entité. Les entités sont ensuite positionnées dans la SEE pour être jugées perceptibles ou non.

Pour positionner une entité dans l'octree, deux critères sont pris en compte :

- La taille de l'entité. La taille de l'entité détermine la profondeur de sa position dans l'octree. Le niveau d'échelle convenable à l'entité correspond à la plus petite cellule qui pourrait contenir sa sphère englobante.
- La position de l'entité. L'entité est associée à la cellule de niveau correspondant qui contient son barycentre.

L'entité est ainsi référencée dans l'octree en fonction de ses caractéristiques d'espace et d'échelle.

On stocke dans la SEE seulement les cellules de l'octree qui contiennent des entités (cellules feuilles) et leurs ancêtres. La SEE est donc dans la pratique relativement vide.

Pour chaque entité observatrice, la SEE est parcourue à partir de la cellule racine et pour chaque cellule on évalue la fonction de filtrage. Si une cellule est rejetée, le parcours ne continuera pas dans la descendance de cette cellule. Sinon, elle est retenue comme cellule perçue.

Pour l'évaluation de la fonction de perception, les cellules de la SEE sont approchées par leurs sphères englobantes. La différence entre les surfaces projetées des cellules d'un même niveau est due à la différence de positions et par suite aux distances qui séparent les cellules de l'entité observatrice. Alors en fixant un seuil de rejet, on pourra déterminer la distance maximale acceptable pour laquelle une cellule à un niveau d'échelle est gardée. On obtient ainsi pour chaque niveau d'échelle, une distance maximale acceptable qui est traduite par une sphère de perception centrée sur l'observateur. L'observateur possède donc un ensemble de sphères de perception correspondant chacune à un niveau d'échelle de la SEE. Une cellule est alors rejetée si sa sphère englobante ne chevauche pas la sphère de perception correspondant à son niveau d'échelle (figure 2.15).

La décision de rejet ou d'acceptation des entités est le résultat d'un test d'intersection entre leurs sphères englobantes et la sphère de perception de l'observateur prise au niveau d'échelle correspondant (simple calcul de distance euclidienne).

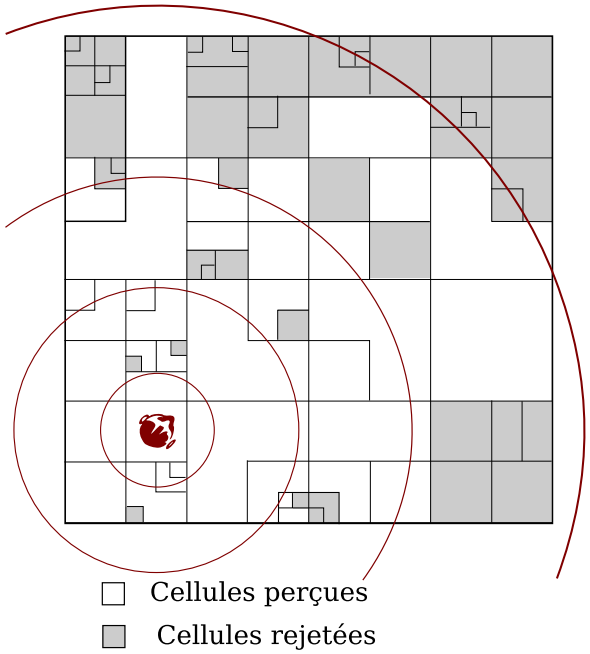

FIG. 2.15 – Représentation 2D d'une SEE

Distribution des messages

La fonction de surface projetée associe à chaque entité un groupe multicast. La gestion d'abonnement à ces groupes diffère selon l'architecture utilisée. En effet, la fonction de surface projetée a été mise en œuvre au sein d'une architecture égal-à-égal et d'une architecture client/multi-serveurs.

Dans l'architecture égal-à-égal, les entités s'abonnent les unes chez les autres. Un groupe multicast global permet aux entités de savoir à quels groupes elles doivent s'abonner.

Dans l'architecture client/multi-serveurs, les entités envoient leurs mises à jour au serveur qui met à jour la SEE et transmet la mise à jour à travers le groupe multicast de l'entité et à travers le groupe multicast global pour en informer les autres serveurs. Le serveur est responsable de la connectivité de ses clients au cours de la simulation.

Ainsi, la fonction de surface projetée permet une discrétisation multi échelle de l'espace.

Elle prend non seulement en compte la distance qui sépare les entités mais aussi l'importance individuelle des entités. La structure d'espace échelle permet à travers de son octree une gestion graduelle tridimensionnelle de l'espace.

Limitations de cette méthode :

- L'association d'un groupe multicast par entité ne permet pas un passage facile à l'échelle dans le cas d'un environnement très peuplé.
- Le groupe multicast global utilisé dans les deux architectures proposées constitue un goulot d'étranglement pour l'application.

2.3.3.4 Les objets tiers

Une autre méthode de filtrage hybride est les objets tiers. Les objets tiers [BG97, GB97] représentent une extension du modèle spatial d'interaction, ils ont été mis en œuvre dans MASSIVE-2 [Gre96]. Les objets tiers ont été introduits par Benford et Greenhalgh pour repousser les limites du modèle spatial relatives :

- au passage difficile à l'échelle dans les environnements densément peuplés. Ce problème étant dû à l'utilisation exclusive des interactions bilatérales entre les entités, d'où la difficulté de la gestion de foules.
- au manque de considération pour les facteurs environnementaux dans le calcul de la conscience (comme les régions fermées telles que des salles et des bâtiments).

Le concept des objets tiers est une notion étendue et améliorée des adaptateurs du modèle original 2.3.2.3. Les adaptateurs originaux (comme par exemple les tables de conférence, les mégaphones...) manipulent la conscience d'une manière limitée (ils ne font que remplacer le focus, le nimbus et l'aura des objets) et ne sont pas capables de travailler récursivement (comme dans les frontières emboîtées). Alors qu'un objet tiers est un objet indépendant qui affecte la conscience dans le monde virtuel en adaptant les relations de conscience entre d'autres entités ou en introduisant de nouvelles relations de conscience indirectes entre elles. Comme le focus et le nimbus, tous les aspects de l'opération des objets tiers sont spécifiques aux médias. Ces objets peuvent être mobiles ou fixes, peuvent être créés dynamiquement ou statiquement et peuvent appliquer leurs effets récursivement l'un sur l'autre. La possibilité de mobilité et de changement

de dimensions permet de supporter des foules dynamiques. De plus la possibilité de récursivité permet une hiérarchisation des foules (une foule pourra contenir une autre foule) et aide aussi à la construction de structures complexes emboîtées (comme les villes virtuelles).

Il y a deux aspects principaux des objets tiers : leurs effets (c'est-à-dire comment ils influencent les relations de conscience entre les autres objets) et leurs activations (c'est-à-dire quand et comment ces effets sont introduits dans l'opération). Considérons maintenant chacun de ces aspects.

Effets des objets tiers

Les objets tiers ont deux types d'effets sur la conscience :

1. **L'adaptation :** elle implique la manipulation des relations de conscience préexistantes entre les objets. Elle peut causer une atténuation ou une amplification des niveaux mutuels de conscience (l'objet tiers peut jouer le rôle d'une barrière ou d'un mégaphone par exemple).
2. **La conscience indirecte :** elle implique l'introduction de nouvelles relations de conscience indirectes entre les objets. Typiquement, la conscience indirecte consomme les informations d'un objet ou d'un groupe d'objets et les transforme dans le but de transmettre une vue alternative de l'objet ou du groupe (par exemple générer une vue agrégée d'une foule ou d'une pièce vue de loin).

La combinaison entre l'adaptation et la conscience indirecte est ce qui permet un passage à l'échelle aisé : dans des circonstances appropriées, un objet tiers peut remplacer plusieurs relations individuelles de conscience avec une seule conscience indirecte. De cette façon, il réduit la quantité de données transmises et traitées.

Activation des objets tiers

Nous considérons maintenant les circonstances dans lesquelles les différentes combinaisons de ces effets sont appliquées. L'activation des objets tiers est basée sur les rapports de conscience entre l'objet tiers et les autres objets :

1. **L'appartenance :** l'objet tiers est activé en fonction des niveaux de conscience qu'il a des autres objets. Ces niveaux de conscience montrent le degré d'appartenance des objets à l'objet tiers ; par exemple, un objet peut faire partie d'une pièce (l'objet tiers) en traversant sa frontière, ce qui implique une modification de la conscience.
2. **Le partage :** l'objet tiers est dans ce cas activé en fonction des niveaux de conscience qu'ont les autres objets de lui. L'objet tiers est dans un sens partagé par des objets et influence leurs consciences mutuelles. Par exemple, deux objets qui se concentrent sur un même objet peuvent éprouver une augmentation de leurs consciences respectives causée par l'objet d'intérêt commun.
3. **Le cas hybride :** c'est le cas où les effets de l'objet tiers dépendent transitivement de la conscience du premier objet de l'objet tiers et de la conscience de l'objet tiers du second objet. Dans ce cas là, l'objet tiers consomme l'information de ses membres et la rend disponible aux observateurs extérieurs comme une vue agrégée avec un niveau de conscience bas (par exemple l'observation des foules).

Connectivité

Les objets tiers utilisent le multicast pour un meilleur filtrage. A chaque objet tiers on associe un ou plusieurs groupes multicast (correspondant aux différents média par exemple). La hiérarchisation des groupes multicast, en cas de division de l'espace en régions, impose qu'un utilisateur n'envoie ses données que sur un seul groupe à la fois (la région la plus petite à laquelle il appartient). Les groupes supérieurs dans la hiérarchie pourront envoyer des agrégations des données émises aux bas niveaux.

Domaines d'application Les objets tiers sont capables d'effectuer un filtrage efficace dans plusieurs domaines d'application comme la division de l'espace, la vue agrégée, la gestion des foules, la gestion de la charge...

La division de l'espace en régions est l'un des usages essentiels des objets tiers. Avec cette division du monde virtuel, différents niveaux de conscience sont appliqués à travers les frontières. Une région peut être un objet tiers dont les objets deviennent membres dès qu'ils traversent l'une

de ses frontières. Les régions peuvent être emboîtées, ce qui permet la définition de structures complexes dans le monde virtuel comme des villes virtuelles, divisées en bâtiments, divisés en étages et pièces. Ceci est fait en appliquant des manipulations de conscience à travers leurs frontières. Le monde virtuel peut alors être divisé en un nombre quelconque de régions de tailles et de formes variées.

Les régions peuvent avoir des effets différents sur les divers médias selon leurs frontières (par exemple, une fenêtre est imperméable pour le média "audio" et perméable pour le média "visuel", par contre un rideau possède les effets inverses). Cette structuration implique un filtrage effectif des données car un participant ne recevra que les données relatives aux objets présents dans sa région et dans les régions qui lui sont accessibles.

Une autre application des objets tiers est de définir des vues abstraites des collections dynamiques d'objets. La différence primordiale avec les régions est que ces objets tiers peuvent ici être mobiles ; ce qui permet, par exemple, de gérer les foules.

Une foule est un objet tiers qui représente et dirige un grand nombre de participants dans un monde virtuel (comme une audience pour un événement virtuel). Les foules peuvent être mobiles ou statiques (attachées à des bancs par exemple).

Les foules sont en général asymétriques. De l'extérieur, une foule fournit une vue agrégée de ses membres à travers l'ensemble des médias appropriés. Typiquement, cet agrégé sera généré dynamiquement et fournira de l'information sur la composition et sur l'activité courante de la foule. D'autre part, de l'intérieur un membre peut percevoir le monde avec tous ses détails (ou même d'une manière amplifiée, s'il regarde un acteur par exemple). L'interaction entre les individus devient possible à l'intérieur de la foule.

Comme la foule est représentée par un objet unique pour les objets externes, le taux de données échangées avec ces objets est énormément réduit, ils ne pourront recevoir les informations concernant les objets individuels de la foule qu'après avoir traversé les frontières de la foule.

Également, la gestion de la charge est une application des objets tiers. Elle est utilisée au niveau du système plutôt qu'au niveau de l'application. Le passage à l'échelle permis par le remplacement des relations de conscience individuelles par des consciences indirectes agrégées peut être exploité par le système comme une façon de gérer la charge. Ainsi, quand le volume de calcul ou le trafic du réseau dépasse un seuil déterminé, le système introduira automatique-

ment des objets tiers dans le monde en regroupant certains participants. Plus généralement, des techniques ont été proposées pour prédéterminer les embouteillages sociaux dans les structures virtuelles urbaines. Ces techniques peuvent être utilisées pour introduire automatiquement des objets tiers dans des endroits du monde virtuel où la densité de la population est prévue d'être élevée.

Les objets tiers ont réussi à gérer les facteurs environnementaux et à renforcer le passage à l'échelle dans les environnements virtuels grâce à de l'introduction de groupe multicast qui n'étaient pas présents dans le modèle spatial d'interaction. Ils ont présenté une solution pour la gestion des foules (qui était jusqu'à lors un problème persistant) à l'aide de l'agrégation. Les objets tiers ont également proposé un moyen efficace pour la répartition de l'environnement virtuel en régions emboîtées.

Inconvénients des objets tiers :

- Une entité peut recevoir les informations en double dans des régions emboîtées parce qu'elle reçoit l'information complète au plus bas niveau et les informations agrégés aux niveaux supérieurs.
- L'agrégation nécessite beaucoup de calcul pour être effectuée.

2.4 Comparaison entre les méthodes de filtrage présentées

Dans cette section, nous comparons les méthodes de filtrage présentées selon quatre critères : la classification du filtrage, l'architecture de contrôle, l'utilisation du multicast et les domaines d'application.

2.4.1 La classification du filtrage de Macedonia et al.

Macedonia et al. [MZP+95] ont classifié le filtrage en : filtrage spatial, temporel et fonctionnel. Chacune des méthodes présentées effectue un filtrage qui peut être classé dans une ou plusieurs de ces trois catégories (figure 2.16).

Les méthodes de prédiction (le **"dead-reckoning"** et le **modèle explicite de comportement**) utilisent un filtrage temporel puisqu'une entité n'envoie une mise à jour que lorsque

	Filtrage temporel	Filtrage spatial	Filtrage Fonctionnel
Le dead-reckoning	√	×	×
Le modèle explicite de comportement	√	×	×
Les locaux de SPLINE	×	√	×
Les locaux de MASSIVE-3	√	√	√
Le précalcul de visibilité	×	√	×
Les shards	×	×	×
La gestion des déclarations (DM)	×	√	×
La gestion de la distribution de données (DDM)	×	√	√
Le modèle spatial d'interaction	×	√	√
La gestion prédictive d'intérêt	×	√	√
Les sphères étendues	×	√	√
Les groupes représentatifs	√	√	√
La gestion par zone d'intérêt	√	√	√
La gestion d'intérêt en trois étapes	√	√	√
La fonction de surface projetée	×	√	×
Les objets tiers	√	√	√

FIG. 2.16 – Classification du filtrage

son comportement réel dévie significativement de son comportement prédit. Donc ces méthodes considèrent que les entités n'ont pas besoin d'une réception fréquente des mises à jour, la fréquence de l'émission des messages dépend du comportement de l'entité : si l'entité a un comportement uniforme, il n'y aura pas de nécessité d'envoyer des mises à jour pendant de longues périodes.

Les **locaux de SPLINE** utilisent un filtrage spatial se basant sur la division de l'espace en locaux. Par contre, les **locaux de MASSIVE-3** utilisent en plus du filtrage spatial un filtrage fonctionnel grâce aux aspects correspondant aux différentes classes fonctionnelles. Les locaux de MASSIVE-3 utilisent aussi un filtrage temporel à travers les abstractions qui fournissent plusieurs niveaux de fidélité selon les besoins et les capacités des utilisateurs.

Le **précalcul de visibilité** décide de la réception des mises à jour en se basant sur un précalcul de visibilité effectué au début de la simulation et sur la distribution des entités dans l'environnement. Ce qui fait que cette méthode n'utilise qu'un filtrage spatial.

Les **shards** sont des duplications de l'environnement virtuel, elles ne permettent aucune cohérence de l'état de l'environnement chez les différentes shards. Le filtrage des shards n'a pas de propriétés spatiales, ni fonctionnelles, ni temporelles.

La **gestion des déclarations** de HLA utilise un filtrage fonctionnel étant donné que la déclaration d'intérêt ne dépend que des classes des objets.

Par contre, la **gestion de la distribution de données** effectue un filtrage fonctionnel et spatial. Les fédérés spécifient quelles sont les données qu'ils veulent envoyer ou recevoir et où elles se trouvent dans l'espace ou dans la grille de routage. Les dimensions de cette dernière peuvent exprimer les caractéristiques fonctionnelles en plus de leurs coordonnées spatiales.

Le **modèle spatial d'interaction** utilise un mélange entre le filtrage spatial et fonctionnel. Ce modèle utilise les distances séparant les entités pour déterminer si elles sont conscientes l'une de l'autre. D'autre part, chaque entité a une aura différente dans les différents médias, ce qui fait que deux entités peuvent interagir dans un média et pas dans un autre. Donc l'interaction entre deux entités ne dépend pas seulement de la distance qui les sépare, mais aussi des capacités individuelles de chaque entité.

La **gestion prédictive d'intérêt** est une extension du modèle spatial d'interaction, elle présente une amélioration de la détection de collision d'auras et résout le problème des inter-

actions ratées. Donc cette gestion utilise, comme le modèle spatial, un mélange entre le filtrage fonctionnel et le filtrage spatial.

Les **sphères étendues** se basent aussi sur le modèle spatial d'interaction, elles présentent une autre amélioration de la détection de collision entre les auras. Donc leur filtrage est un filtrage fonctionnel et spatial.

Les **groupes représentatifs** utilisent les notions d'intérêts individuels et de proximité pour établir leur filtrage. Ils permettent également un transfert de données de fidélités différentes. Leur filtrage peut être considéré comme étant une combinaison des trois catégories de filtrage de Macedonia et al.

La **gestion par zone d'intérêt** se base sur une combinaison du filtrage spatial, temporel et fonctionnel. Les entités sont distribuées dans des classes indépendantes des trois catégories. Les entités présentent dans une même classe peuvent interagir entre elles.

La **gestion d'intérêt en trois étapes** utilise un filtrage spatial, fonctionnel et temporel : la première étape effectue un filtrage spatial qui divise et subdivise l'espace en des régions selon la distribution des entités dans l'espace. La deuxième étape permet d'éliminer les entités non intéressantes obtenues à la fin de la première étape selon leurs positions et leurs fonctionnalités. Finalement, la troisième étape effectue un filtrage fonctionnel et temporel basé sur les médias et sur la fréquence ou l'agrégation.

La **fonction de surface projetée** utilise un filtrage spatial qui prend en considération la taille des entités et la distance qui les sépare.

Les **objets tiers** utilisent le filtrage spatial et fonctionnel. Les objets tiers sont une extension du modèle spatial d'interaction, ils s'appuient donc sur les mêmes caractéristiques de filtrage du modèle spatial. De plus, ils offrent la possibilité de structuration du monde virtuel en des régions et la possibilité de fournir une vue agrégée des foules, ce qui permet un filtrage temporel.

2.4.2 L'architecture de contrôle

Certaines méthodes de filtrage ont utilisé une seule architecture de contrôle (figure 2.17) alors que d'autres ont utilisé des architectures hybrides pour des raisons d'optimisation de fonctionnement : Les locaux de SPLINE ont utilisé au début une architecture égal-à-égal ; ensuite un serveur dédié aux utilisateurs à mauvaise connexion a été introduit. Le modèle spatial d'interac-

tion utilise également une architecture égal-à-égal pour la communication entre les utilisateurs et un serveur dont le rôle est de détecter la collision entre les auras des entités. Les groupes représentatifs ont aussi utilisé une architecture égal-à-égal avec des serveurs spécialisés mais ces serveurs sont dédiés à l'authentification.

	Égal-à-égal	Client/Serveur		Architecture hybride
		mono-serveur	multi-serveurs	
Le dead-reckoning	√			
Le modèle explicite de comportement			√	
Les locaux de SPLINE	√			Serveur dédié aux utilisateurs à mauvaise connexion
Les locaux de MASSIVE-3	√			
Le précalcul de visibilité	√		√	
Les shards			√	
La gestion des déclarations (DM)		√		
La gestion de la distribution de données (DDM)		√		
Le modèle spatial d'interaction	√			Serveur dédié à la détection de collision d'auras
La gestion prédictive d'intérêt		√		
Les sphères étendues		√		
Les groupes représentatifs	√			Serveurs dédiés à l'authentification
La gestion par zone d'intérêt	√			
La gestion d'intérêt en trois étapes		√		
La fonction de surface projetée	√		√	
Les objets tiers		√		

FIG. 2.17 – Comparaison entre les architectures de contrôle

La conception de certaines méthodes de filtrage est liée à l'architecture utilisée (comme la gestion de déclarations et la gestion de la distribution de données par exemple). D'autres méthodes sont adaptables à n'importe quel type d'architecture, pour cette raison quelques unes d'entre elles ont été même mises en œuvre dans plusieurs architectures de contrôle pour des

buts d'amélioration ou d'évaluation. Le précalcul de visibilité a été mis en œuvre dans une architecture égal-à-égal et dans une architecture client/multi-serveurs. Mais la comparaison entre leurs fonctionnements n'a pas été effectuée. Également, la fonction de surface projetée a proposé deux architectures de contrôle pour la mise en œuvre : une architecture égal-à-égal et une architecture multi-serveurs. Elle a été mise en œuvre uniquement dans une architecture égal-à-égal.

En général, les architectures égal-à-égal sont très adaptées aux systèmes fortement interactifs mais le contrôle de l'application et le filtrage sont plus difficiles à gérer. Les architectures client/serveur permettent ce contrôle mais le goulot d'étranglement associé à l'utilisation d'un seul serveur handicape le passage à l'échelle. Pour ces raisons, les applications d'EVDs à grande échelle (comme les MMOGs) utilisent des architectures client/multi-serveurs qui gardent les avantages du client/serveur sans être limitées par un goulot d'étranglement. La difficulté que présentent ces architectures réside dans la gestion de la simulation à travers les serveurs ; nous reviendrons sur cela dans nos travaux.

2.4.3 L'utilisation du multicast

Le multicast est un mode de communication très adapté au filtrage. Pour cette raison, beaucoup de méthodes de filtrage l'utilisent et l'associent à des régions, à des utilisateurs, ou à des groupes d'utilisateurs (figure 2.18).

SPLINE associe à chaque local un ou plusieurs groupes multicast correspondant aux différents média (audio, visuel, texte...). **MASSIVE-3** fournit pour chaque local plusieurs abstractions qui correspondent aux différents niveaux de fidélité (résolution, qualité...). MASSIVE-3 n'associe donc pas seulement les groupes multicast selon les média mais aussi selon les différentes fidélités.

HLA utilise le multicast. Pour les applications simples où la gestion de déclarations (DM) est chargée du filtrage, il associe à chaque donnée un groupe multicast. Pour les applications complexes, la gestion de la distribution de données (DDM) en a la charge. Selon l'approche qu'elle adopte, elle peut associer les groupes multicast :

- aux fédérés ;
- à leurs régions de mise à jour ;

	Multicast	Attribution des groupes multicast aux :		
		entités	groupes d'entités	régions
Le dead-reckoning	×			
Le modèle explicite de comportement	×			
Les locaux de SPLINE	√	×	×	un groupe/média
Les locaux de MASSIVE-3	√	×	×	un groupe/média un groupe/fidélité
Le précalcul de visibilité	√	×	×	un groupe/cellule
Les shards	×			
La gestion des déclarations (DM)	√	√	×	×
La gestion de la distribution de données (DDM)	√	√	×	√
Le modèle spatial d'interaction	×			
La gestion prédictive d'intérêt	×			
Les sphères étendues	√	×	√	×
Les groupes représentatifs	√	×	√	×
La gestion par zone d'intérêt	√	×	√	×
La gestion d'intérêt en trois étapes	√	×	√	√
La fonction de surface projetée	√	√	×	×
Les objets tiers	√	√	√	√

FIG. 2.18 – Utilisation du multicast

– aux intersections entre les régions d'abonnement et les régions de mise à jour ;

– ou bien aux cellules de la grille de routage.

La **méthode des sphères étendues** rassemble les entités dont les auras chevauchent entre elles dans des ensembles appelés relations de collisions (RCs). Chaque RC possède un groupe multicast à travers lequel ses membres échangent leurs données.

Les **groupes représentatifs** (GR) constituent des rassemblements des entités proches ayant les mêmes intérêts. Chaque GR a un groupe multicast que les entités externes joignent pour recevoir des données de basse fidélité du GR.

La **gestion d'intérêt** utilise le multicast et associe à chaque classe spatiale, fonctionnelle ou temporelle un groupe multicast.

La **gestion d'intérêt en trois étapes** utilise également le multicast. La première étape associe des groupes multicast aux cellules de l'octree qui divise l'espace, les deuxième et troisième étapes associent à chaque entité un ou plusieurs groupes multicast selon les différents médias et fidélités.

La **fonction de surface projetée** utilise le multicast basé sur les sources. Elle associe à chaque entité un groupe multicast, ce qui est assez coûteux lors du passage à l'échelle.

Les **objets tiers** utilisent le multicast et permettent une allocation dynamique des groupes multicast. Chaque monde possède un groupe multicast et à l'intérieur d'un monde, il peut y avoir des groupes multicast associés à des foules ou à des régions de ce monde. En même temps, il peut y avoir des groupes multicast associés à des artefacts particuliers.

Toutes ces méthodes ont des groupes multicast destinés à être joints par les utilisateurs des applications, à l'exception du **précalcul de visibilité** de RING qui a aussi utilisé le multicast pour établir la communication entre les serveurs. En effet, RING a été mis en œuvre dans une architecture égal-à-égal avec une communication via unicast ou multicast, et il a été mis en œuvre dans une architecture client/multi-serveurs avec une communication via unicast ou multicast. Par contre, la différence de fonctionnement entre ces différentes combinaisons n'a pas été évaluée en détails.

Toutes les autres méthodes communiquent en Unicast, exception faite du **"dead-reckoning"** de SIMNET qui utilise la diffusion.

2.4.4 Les domaines d'application

2.4.4.1 Critères de la classification

Comme les environnements virtuels sont assez variés, nous avons défini un nombre de critères que les méthodes de filtrage doivent satisfaire pour être utilisées dans le plus de domaines possible. Ces critères sont :

- **L'expression de manifestation**. Toutes les méthodes de gestion d'intérêt permettent aux entités d'exprimer leurs intérêts alors qu'il n'y a que quelques méthodes qui leur permettent d'exprimer leurs manifestations. En réalité, une personne est consciente d'une autre personne non seulement à cause de ses intérêts ou de la distance qui les sépare mais aussi à cause de la façon avec laquelle l'autre personne se manifeste. Par exemple, dans un champ il est beaucoup plus probable que l'on voie un éléphant qu'une souris s'ils sont à la même distance. Également, notre attention est beaucoup plus attirée par un orateur que par les personnes qui l'entourent. Donc la perceptibilité d'une entité dépend non seulement des entités qui la voient mais aussi de ses propres caractéristiques comme sa taille, sa fonction, son comportement... Malheureusement, peu de méthodes de filtrage incluent la manifestation des entités dans leurs fonctionnements, ce qui fait qu'elles fournissent un filtrage moins exact que les autres.
- **Les relations asymétriques**. En réalité, les relations asymétriques existent, c'est-à-dire on peut voir sans être vu, entendre sans être entendu... L'asymétrie est un aspect naturel et parfois essentiel des interactions. De plus, "symétriser" une relation asymétrique peut causer la réception inutile de messages non pertinents.
- **Les architectures emboîtées**. Le filtrage au sein des architectures emboîtées exige la possibilité de la division de l'espace en régions. Une gestion individuelle de chaque région est également nécessaire pour permettre ou interdire une transparence entre ces régions. Cette gestion individuelle permet un meilleur passage à l'échelle en éliminant les comparaisons entre les intérêts des entités présentes dans des régions isolées.
- **Les environnements ouverts**. Les méthodes basées exclusivement sur la division de l'espace n'ont pas des moyens de gestion des environnements ouverts. En effet, la division d'un environnement ouvert n'est pas naturelle et ne peut pas aboutir à un filtrage réaliste

si elle n'est pas supportée par d'autres notions de filtrage.

– **Le problème de regroupement**. Lorsque plusieurs entités se retrouvent regroupées dans un espace restreint, la gestion de l'interaction entre ces entités devient coûteuse. Pour cette raison, des méthodes de filtrage ont présenté des solutions à ce problème.

2.4.4.2 Classification

Voyons donc à quels critères répond chacune des méthodes présentées, ceci permettant de déterminer leurs domaines d'application (figure 2.19)

Les **méthodes de prédiction** sont appropriées aux applications ayant des entités au comportement prédictible. Elles ne présentent aucune solution pour l'expression de manifestation, les relations asymétriques, les architectures emboîtées, les espaces ouverts ou le regroupement.

Les **locaux** et le **précalcul de visibilité** ne définissent pas les intérêts des entités, ni leurs manifestations ; mais ceci n'empêche pas l'établissement de relations asymétriques : une entité peut voir une entité d'une chambre voisine qui ne la voit pas. D'autre part, ces méthodes ont été conçues pour les architectures emboîtées et ne peuvent pas s'adapter aux environnements ouverts. Les locaux et le précalcul de visibilité ne calculent pas les intérêts de leurs entités en temps réel, les entités regroupées dans un local ou une cellule communiquent donc toutes ensemble sans poser le problème de charge induite par le calcul d'intérêt. Néanmoins, le regroupement d'entités peut causer une augmentation de la charge de calcul de ces entités.

Les **"shards"** ne présentent aucune solution pour les contraintes d'application des EVDs. La solution qu'elles présentent, par exemple, pour le regroupement est de l'interdire : dès que la limite des utilisateurs souhaitant être présents dans le même environnement est atteinte, une nouvelle "shard" est créée.

Dans la **gestion de déclarations** de HLA, chaque entité exprime séparément ses intérêts et sa manifestation à travers la définition des données qu'elle veut recevoir et des données qu'elle veut envoyer. Ainsi si une entité est intéressée par les données d'une autre alors que l'inverse n'est pas vrai, ces deux entités peuvent établir une relation asymétrique. En revanche, ce service exprime l'intérêt et la manifestation d'une entité par classe, donc la gestion des architectures emboîtées, des environnements ouverts et du regroupement n'est pas possible parce que la notion d'espace n'est pas présente.

	Expression de manifestation	Relations asymétriques	Architectures emboîtées	Environments ouverts	Re-groupement
Le dead-reckoning	×	×	×	×	×
Le modèle explicite de comportement	×	×	×	×	×
Les locaux de SPLINE	×	√	√	×	√
Les locaux de MASSIVE-3	×	√	√	×	√
Le précalcul de visibilité	×	√	√	×	√
Les shards	×	×	×	×	√
La gestion de déclarations (DM)	√	√	×	×	×
La gestion de distribution de données (DDM)	√	√	dépend de la définition de l'intérêt	dépend de la définition de l'intérêt	×
Le modèle spatial d'interaction	√	√	×	√	×
La gestion prédictive d'intérêt	√	×	×	√	×
Les sphères étendues	√	×	×	√	√
Les groupes représentatifs	×	√	×	√	×
La gestion par zone d'intérêt	×	√	×	√	×
La gestion d'intérêt en trois étapes	×	√	×	√	√
La fonction de surface projetée	√	√	√	√	√
Les objets tiers	√	√	√	√	×

FIG. 2.19 – Les domaines d'application

La **gestion de la distribution de données** permet l'expression de manifestation et les relations asymétriques plus précisément que la gestion de déclarations. Elle peut aussi supporter les architectures emboîtées en délimitant les régions d'abonnement et de mise à jour d'une entité dans la région où elle est présente. La gestion des environnements ouverts est aussi possible en associant aux entités des intérêts restreints. Par contre, les gestions de déclarations et de distribution de données ne fournissent pas les règles et les moyens de définition d'intérêt et de manifestation, cette tâche est réservée au concepteur de l'application.

Les auras du **modèle spatial d'interaction** permettent une définition de l'intérêt d'une entité et de son importance dans la scène. Le modèle utilise aussi le focus pour exprimer l'attention allouée par une entité et le nimbus pour exprimer sa manifestation ; les foci et les nimbi permettent d'établir des relations bilatérales asymétriques. L'asymétrie de ces relations est traduite soit dans la présence de la conscience d'un seul côté sans l'autre soit dans des niveaux de conscience différents des deux côtés. Le modèle spatial d'interaction ne s'adapte pas aux architectures emboîtées mais il est efficace dans les environnements ouverts parce que la notion d'auras permet à des entités distantes d'être présentes dans le même environnement et de ne pas communiquer entre elles. Par contre, ce modèle ne présente aucune solution pour le problème de regroupement. En effet, si les objets ne sont pas dispersés dans l'espace, chaque objet sera conscient de beaucoup d'autres objets et le nombre de relations bilatérales deviendra élevé.

La **gestion prédictive d'intérêt** et les **sphères étendues** utilisent les auras du modèle spatial, ce qui leur permet de supporter l'expression de manifestation. Mais ces méthodes n'utilisent pas les foci et les nimbi pour des raisons de simplicité, donc elles ne peuvent pas avoir des relations asymétriques. D'autre part, ces deux méthodes se comportent similairement au modèle spatial vis-à-vis des architectures emboîtées et des environnements ouverts. Par contre, les sphères étendues présentent une solution efficace pour le problème de regroupement en rassemblant les entités dont les auras chevauchent dans une sphère étendue. De cette façon, les tests de collision entre les auras des entités de ce groupe et les auras des entités externes sont remplacés par un test de collision entre la sphère étendue et l'aura externe.

La **gestion d'intérêt** ne définit pas la manifestation des entités pourtant elle supporte les relations asymétriques entre les entités ayant des zones d'intérêt qui ne se contiennent pas mutuellement. Cette gestion est orientée vers la simulation militaire, elle n'est pas adaptée aux

architectures emboîtées, mais elle peut gérer des environnements ouverts grâce à la définition des zones d'intérêt des entités. De plus, le succès de cette gestion est lié à la dispersion des entités dans l'espace et à la vitesse réduite de leurs déplacements. Néanmoins, si les entités présentes dans l'environnement se regroupent dans la même région ou si elles sont très rapides et changent souvent de cellules et par suite de groupes multicast, la technique ne sera plus efficace.

La **gestion d'intérêt en trois étapes** possède les mêmes caractéristiques que la gestion d'intérêt parce qu'elle se base sur la définition des zones d'intérêt. Par contre, elle utilise une subdivision dynamique des régions densément peuplées selon l'intérêt de chaque hôte (le concept de "la plus petite région"). Le regroupement des entités ne pose plus une limite pour le passage à l'échelle.

La **fonction de surface projetée** prend en considération l'importance individuelle de chaque entité perçue, elle permet une expression de manifestation visuelle et même sonore. De plus, cette méthode permet d'établir des relations asymétriques entre les entités parce qu'elle calcule les intérêts de ses entités indépendamment les unes des autres. D'autre part, cette technique est bien adaptée à la gestion des environnements ouverts et la gestion des architectures emboîtées est aussi possible en excluant les cellules invisibles par une entité de la SEE en calculant sa perception. La fonction de surface projetée est très adaptée au regroupement parce qu'elle permet aux entités de partager les mêmes cellules, ce qui fait que le nombre de cellules de la SEE ne croîtra pas de la même manière que le nombre d'entités.

Les **objets tiers** sont une extension du modèle spatial d'interaction, ils utilisent ainsi les auras, les foci et les nimbi de ce modèle ; ce qui leur permet d'exprimer la manifestation des entités, d'avoir des relations asymétriques et de gérer les environnements ouverts. De plus, les objets tiers sont surtout utilisés pour les régions emboîtées et la gestion de foules. En effet, le problème de regroupement est résolu en éliminant les interactions bilatérales en cas de présence de foules en créant des agrégations.

Deuxième partie

Propositions

1

La gestion d'effet

Dans ce chapitre, nous présentons notre méthode de filtrage "la gestion d'effet" [ETJ06, ETRJ06]. La gestion d'effet est une extension du modèle spatial d'intérêt qui a pour but de :

- permettre un établissement moins coûteux des relations asymétriques entre les entités ;
- s'adapter aux environnements ouverts et aux architectures emboîtées ;
- s'adapter à une architecture client/multi-serveurs ;
- fournir un passage à l'échelle continu ;
- utiliser des modes de communication correspondant aux besoins de l'application (dont le multicast) ;

1.1 Conception de la gestion d'effet

Le fonctionnement du modèle spatial d'interaction consiste en deux étapes :

1. La détection de collision entre les couples d'auras. La détection d'une collision d'auras implique la possibilité d'interaction entre les deux entités.

2. Les entités négocient leurs niveaux de conscience à travers leurs foci et leurs nimbi pour décider si une conscience d'un côté ou de l'autre existe. Ainsi, si les niveaux de conscience de ces entités sont disproportionnés, le modèle spatial établit une relation asymétrique entre elles. Cette asymétrie peut être exprimée par l'existence de conscience d'un côté et pas de l'autre, ou par l'attribution de ressources différentes selon les niveaux de conscience.

Le concept de consciences asymétriques est très réaliste : on peut voir sans être vu, entendre sans être entendu... Par exemple, une entité importante est connue par toutes les entités alors que l'inverse n'est pas vrai. L'asymétrie est donc un aspect naturel et souvent essentiel des relations de conscience. De plus, symétriser une relation asymétrique peut causer la réception inutile de messages non pertinents.

Les négociations des niveaux de conscience du modèle spatial déterminent si la conscience est établie d'un côté, des deux côtés ou d'aucun côté. Ainsi, le changement d'état de l'une des deux entités relance les négociations des niveaux de conscience. Souvent, ces négociations n'aboutissent pas à un changement des relations de conscience. Par suite, cet établissement de relations asymétriques est assez coûteux pour le réseau et les machines.

Pour cette raison, la gestion d'effet propose d'établir des relations asymétriques dès la première étape sans l'utilisation des foci et des nimbi. La deuxième étape sera dédiée seulement à la détermination des niveaux de conscience **non nulles**. Ces niveaux de conscience serviront à l'attribution dynamique des ressources.

Tous les systèmes n'ont pas besoin d'allouer dynamiquement les ressources selon les niveaux de conscience. Sachant que pour des raisons de simplicité et de coût, toutes les méthodes qui se sont basées sur le modèle spatial (la gestion prédictive de comportement, les sphères étendues, les groupes représentatifs...) n'ont utilisé que la première étape. Et donc, il ne leur est pas possible d'établir des relations asymétriques.

Pour déterminer l'existence de la conscience séparément des deux côtés, la gestion d'effet remplace les auras par des zones de conscience et d'effet. Elle associe donc à chaque entité une zone de conscience qui reflète sa capacité de conscience et une zone d'effet qui reflète sa capacité d'être aperçue, au lieu de combiner les deux capacités dans un seul composant comme fait le modèle spatial en définissant les auras. La gestion d'effet définit ces zones dans chaque média ; quand la zone de conscience d'une entité A chevauche la zone d'effet d'une entité B dans un média donné, A devient consciente de B dans ce média.

1.1.1 Les zones d'effet

La zone d'effet d'une entité est une zone sphérique centrée sur l'entité. Une entité a autant de zones d'effet qu'il y a de média. Le rayon d'une zone d'effet dépend de l'importance de l'entité

dans le média correspondant. Cette importance est relative à sa taille dans le média visuel (un éléphant est plus perceptible qu'une souris), sa situation (un avion est facilement vu dans un ciel clair), son comportement (se déplacer rapidement, crier...), ou à sa fonction (un orateur dans un stadium est vu et entendu par toute l'audience malgré qu'il ait la même structure que tous les spectateurs).

L'avion de la figure 1.1 possède une zone d'effet visuelle qui couvre tout le sol. Donc toutes les entités le voient et restent en même temps inconscientes des petites entités distantes ayant des zones d'effet restreintes.

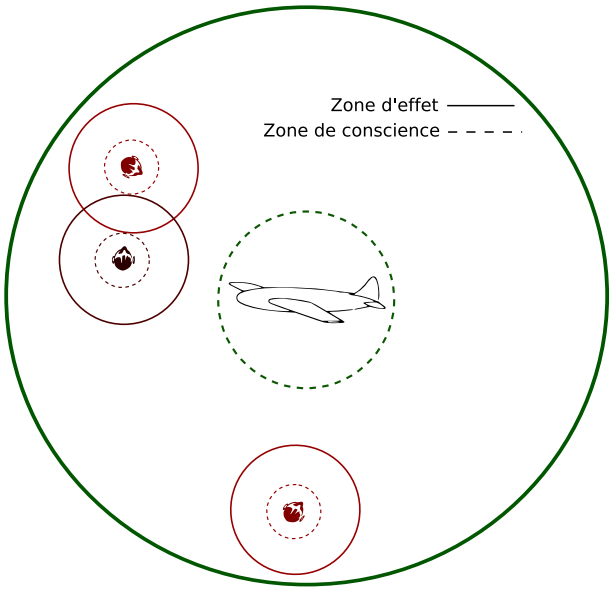

FIG. 1.1 – Zones de conscience et d'effet visuels

1.1.2 Les zones de conscience

Les entités n'ont pas toutes les mêmes capacités de conscience à cause de leurs structures, leurs fonctions ou leurs comportements. Une entité peut simuler une personne myope ou muette, ou représenter une personne équipée de jumelles. On peut aussi avoir une entité bien positionnée ou rapide. Les entités ont donc des degrés de conscience différents. Pour différencier entre ces degrés, on associe à chaque entité une zone de conscience spécifique à chaque média. Cette zone de conscience est également une sphère centrée sur l'entité, ayant un rayon relatif à son degré de conscience. Quand la zone de conscience d'une entité A chevauche la zone d'effet d'une entité B dans un média donné, A devient consciente de B dans ce média. Avoir des zones de conscience

sphériques permet d'éviter de refaire les calculs à chaque fois qu'une entité change d'orientation. Dans la figure 1.1, on peut remarquer que les zones de conscience visuelle de toutes les entités chevauchent la zone d'effet visuel de l'avion, ainsi toutes les entités seront capables de voir l'avion. En même temps, la zone de conscience de l'avion ne chevauche aucune des zones d'effet des entités, donc le pilote de l'avion ne recevra pas les messages de mises à jour des ces entités qui ne l'intéressent pas.

1.2 Détermination de la taille des zones de conscience et d'effet

Les méthodes de gestion d'intérêt existantes qui utilisent les zones pour exprimer les intérêts des entités, comme la gestion par zone d'intérêt ou le modèle spatial, définissent la taille de ces zones d'une manière totalement empirique. Dans ce qui suit, on présente notre étude qui a pour but de déterminer la taille des zones de conscience et d'effet visuels en se basant sur la perception graphique [EHTJ07]. On s'est intéressé au média visuel parce que ce média joue le rôle le plus important pour la détermination de la conscience des entités dans les EVDs.

1.2.1 Relation entre conscience visuelle, perception et perceptibilité

La perceptibilité d'une entité dépend de son importance dans l'environnement. Cette importance dépend de sa situation, sa fonction et sa taille. Mais l'importance d'une entité ordinaire dépend essentiellement de sa manifestation physique, c'est-à-dire sa taille. D'autre part, les entités n'ont pas la même capacité de perception. Ainsi la conscience visuelle qu'a une entité d'une autre entité dépend des capacités de perception de l'entité observatrice et de perceptibilité de l'entité observée.

1.2.2 La perception graphique

Plusieurs algorithmes de rendu graphique sont utilisés pour faire des tests de visibilité. Dans ce qui suit, on étudie ces algorithmes pour savoir s'ils peuvent spécifier des paramètres de définition de zones.

Un moteur 3D décide qu'un objet n'est pas visible au cas où l'une de ces trois propriétés est vraie :

– **P1** : l'objet est à l'extérieur du volume de vision de la caméra.

– **P2** : l'objet est caché par d'autres objets.

– **P3** : l'objet couvre moins d'un pixel sur l'écran.

P1 est un critère de filtrage très instable parce que la liste des entités visibles change totalement et dans un bref délai quand l'observateur change rapidement d'orientation. Un changement rapide d'orientation et donc de volume de vision est assez coûteux, ce qui empêche la définition de l'intérêt en se basant sur le volume de vision.

Plusieurs techniques basées sur **P2** ont été utilisées en informatique graphique. Ces techniques pourraient être utilisées à travers par exemple les requêtes sur les occlusions dans les GPUs récents [BMH98]. Mais ces techniques sont difficiles à mettre en œuvre et nécessitent des calculs complexes qui ne peuvent pas être faits dynamiquement (surtout dans les architectures client/serveur où le serveur est responsable du filtrage chez tous les participants).

P3 offre une méthode plus simple : connaissant la résolution de l'écran d'un utilisateur **U** et les paramètres du moteur 3D, on peut facilement déterminer la projection de la sphère englobante d'un objet **A** sur l'écran. Si la taille de projection de la sphère est inférieure à un certain seuil (qui peut être fixé à un pixel), **U** ne pourra pas voir **A**. **P3** sera donc la base de notre filtrage basé sur la perception graphique.

1.2.3 La distance maximale de perception

L'approche standard d'une caméra virtuelle utilisée dans OpenGL et Direct3d se base sur la projection de la scène rendue sur un plan de projection. La caméra est définie pas son étendue angulaire sur la scène. Cette étendue est appelée champ de vision (fov : field of view) et définit un volume de vue (une pyramide) qui est ensuite subdivisée en de plus petits volumes (un volume par pixel) (figure 1.2). Ces subdivisions permettent de déterminer l'angle de vue couvert par un pixel pour un utilisateur spécifié en se basant sur la résolution horizontale de la fenêtre ($NbPixelsX$) et le champ de vision de sa caméra (fov). L'écran couvre 100% du champ de vision, ainsi un pixel couvre un angle θ moyen[4] :

$$\theta = \frac{fov}{NbPixelsX}$$

[4]Le modèle de caméra utilisé induit une distortion pour les pixels éloignés du centre de l'écran. On considérera cette distortion négligeable dans notre étude.

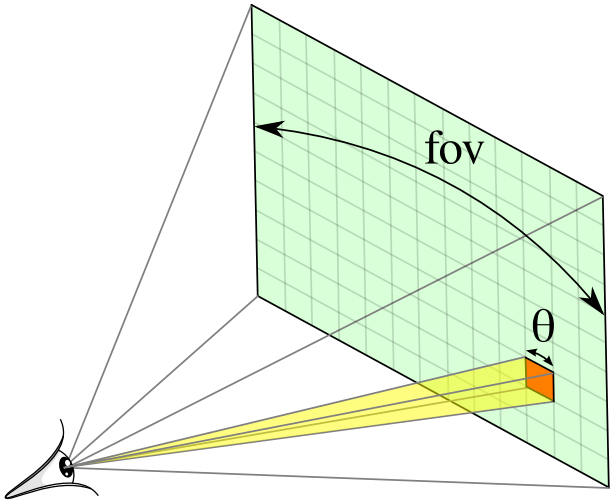

FIG. 1.2 – L'angle de vue d'une caméra virtuelle

D'autre part, on peut calculer trigonométriquement l'étendue angulaire "α" de la projection d'un objet 3D sur le plan de projection (figure 1.3). Pour calculer cette étendue, on voit dans la figure 1.4 que la relation entre l'angle " α ", le côté opposé "R" (le rayon de la sphère englobante) et le côté adjacent "D" (la distance entre l'objet et la caméra) est établie comme suit :

$$\tan(\alpha/2) = \frac{R}{D} \quad \textit{(équation 1)}$$

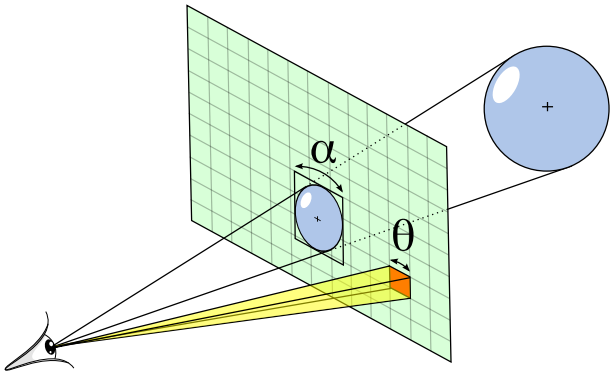

FIG. 1.3 – L'étendue angulaire d'un objet et d'un pixel

Selon nos critères de filtrage, la propriété **P3** permet l'affichage d'un objet lorsque la taille de sa projection est supérieure à un ou plusieurs pixels (figure 1.3). Le nombre minimal de pixels perçus par un observateur dépend de la capacité visuelle de son avatar. Un avatar ayant une vue avantagée pourra voir un objet si sa projection couvre un pixel mais un avatar ayant une vue moins avantagée aura besoin d'une projection de plusieurs pixels pour pouvoir voir l'objet

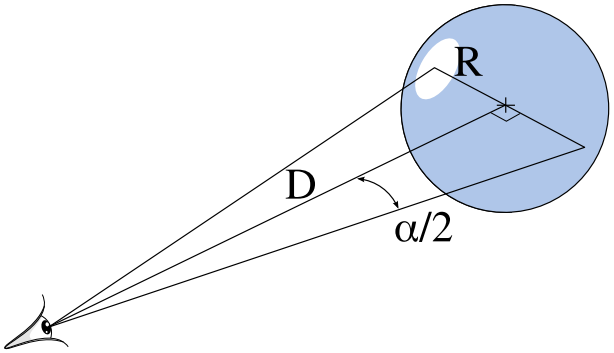

FIG. 1.4 – L'étendue angulaire d'un objet 3D

projeté.

Avoir une projection supérieure à un pixel, par exemple, est équivalent à avoir une étendue angulaire "α" supérieure ou égale à l'angle couvert par un pixel "θ" (figure 1.3). Pour un observateur non avantagé, cet angle minimal sera multiplié par le nombre minimal de pixels perçus par cet observateur ("n"). Donc sachant que "α" et "θ" sont toujours inférieurs à π, les équations suivantes sont possibles :

$$\begin{aligned}
& \alpha && \geq && n \cdot \theta \\
\Leftrightarrow\ & \tan(\alpha/2) && \geq && \tan(n \cdot \theta/2) \\
\Leftrightarrow\ & R/D && \geq && \tan(n \cdot \theta/2) \quad \textit{(d'après équation 1)} \\
\Leftrightarrow\ & D && \leq && \frac{R}{\tan(n \cdot \theta/2)}
\end{aligned}$$

Cette dernière inéquation montre qu'un observateur peut voir un objet si la distance qui les sépare est inférieure ou égale à "$R/\tan(n \cdot \theta/2)$". Ceci nous donne donc la distance maximale de perception qui permet à un utilisateur de voir une entité.

$$\begin{aligned}
Dmax &= \frac{R}{\tan(n \cdot \theta/2)} = R \cdot V \\
\textit{tel que}\quad V &= \frac{1}{\tan(n \cdot \theta/2)}
\end{aligned}$$

Cette distance maximale de perception dépend donc de la capacité visuelle de l'entité qui représente l'utilisateur "V" et de la taille de l'entité observée "R".

Le nombre d'entités visibles dépend des distances maximales de perception de l'entité observatrice. Ainsi, si à un moment donné de la simulation, un utilisateur ou le système global ne peut plus supporter le nombre de messages échangés et veut augmenter son filtrage, il pourra réduire artificiellement les valeurs des distances de perception. Pour ceci, on utilise un coefficient de filtrage "*CF*" :

$$Dmax \; = \; R \cdot V \cdot CF$$

CF aura toujours une valeur inférieure ou égale à 1. Ce qui permettra à l'utilisateur ou au système d'adapter le niveau de filtrage à ses besoins.

1.2.4 Le filtrage basé sur la perception

Notre étude avait principalement pour objectif de définir une technique qui détermine les tailles des zones visuelles d'une manière exacte. Par contre, ces règles de perception pourraient être utilisées pour définir une méthode de filtrage dans les EVDs. Dans ce cas, le serveur (dans une architecture client/serveur) ou les clients (dans une architecture égal-à-égal) calculeront la distance maximale de perception qu'a l'entité observatrice de l'entité observée et compareront ensuite cette distance maximale à la distance qui les séparent réellement pour déduire si une perception aura lieu ou pas.

L'utilisation du filtrage basé sur la perception comme méthode de filtrage parait intéressant et reste une piste à approfondir et à développer. Surtout que ce filtrage présente des avantages et des inconvénients divers. Il fournit par exemple un filtrage visuel très exact mais en même temps ce filtrage ne prend pas en compte les autres facteurs qui influencent la conscience visuelle comme les fonctions des entités, leurs situations et leurs comportements. De plus, ce filtrage ne sert qu'à un seul média : le média visuel et il est difficilement applicable dans les architectures client/multi-serveurs.

1.2.5 Perception graphique et gestion d'effet

Pour que la gestion d'effet satisfasse les règles du filtrage basé sur la perception, il faut que quand une entité en voit une autre avec le filtrage basé sur la perception, elle la voit avec la

gestion d'effet. La détermination des tailles des zones de conscience et d'effet sera donc basée sur la règle suivante :

R : "Quand A voit B avec le filtrage basé sur la perception, A doit voir B avec la gestion d'effet"

A voit B avec le filtrage basé sur la perception si la distance entre A et B est plus petite que la distance maximale de perception de A vers B, ce qui est équivalent à :

$$\mathrm{D(A,B)} \leq D_{max}\mathrm{(A,B)}$$

Et A voit B avec la gestion d'effet si la zone de conscience de A chevauche la zone d'effet de B :

$$\mathrm{D(A,B)} \leq \mathrm{C(A)+E(B)}$$

où C(A) est le rayon de la zone de conscience de A et E(B) est le rayon de la zone d'effet de B

Une condition qui rend notre règle **R** vraie est :

$$\mathbf{C} : D_{max}\mathrm{(A,B)} \leq \mathrm{C(A)+E(B)}$$

parce que :

$$\mathrm{D(A,B)} \leq D_{max}\mathrm{(A,B)} \text{ et } D_{max}\mathrm{(A,B)} \leq \mathrm{C(A)+E(B)} \Rightarrow \mathrm{D(A,B)} \leq \mathrm{C(A)+E(B)}$$

Donc la condition **C** permet de satisfaire notre règle **R**. Ce qui fait d'elle la base de la traduction de la perception graphique en gestion d'effet.

La figure 1.5 nous montre une entité à la limite de perception d'une autre entité. Cette limite correspond à la distance maximale de perception de l'autre entité. La gestion d'effet équivalente possède alors des zones de conscience et d'effet qui sont aussi à la limite de chevauchement.

Mais le problème devient plus compliqué dès que l'on considère plus de deux types d'entités. Dans ce cas là, les zones de conscience et d'effet d'une entité dépendront de l'ensemble des autres entités au lieu de dépendre d'une seule. Prenons l'exemple d'une application qui comprend trois types d'entités A, B et C. Le filtrage basé sur la perception crée une base de données de toutes les combinaisons des distances maximales de perception de l'environnement. Cette base de données a pour rôle de juger si deux entités de n'importe quels types peuvent se voir. Elle comprend

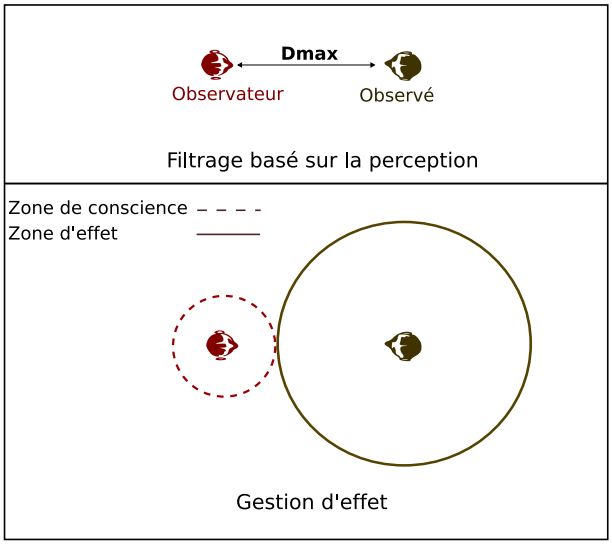

FIG. 1.5 – Traduction de la perception graphique dans la gestion d'effet

donc D_{max}(A,A), D_{max}(A,B), D_{max}(A,C), D_{max}(B,A), D_{max}(B,B), D_{max}(B,C), D_{max}(C,A), D_{max}(C,B) et D_{max}(C,C).

Les conditions de la traduction de la perception seront donc :

$C(A) + E(A) \geq D_{max}(A, A)$ (pour que deux entités de type A se voient quand il le faut)

$C(A) + E(B) \geq D_{max}(A, B)$ (pour que A voit B)

$C(A) + E(C) \geq D_{max}(A, C)$

$C(B) + E(A) \geq D_{max}(B, A)$

$C(B) + E(B) \geq D_{max}(B, B)$

$C(B) + E(C) \geq D_{max}(B, C)$

$C(C) + E(A) \geq D_{max}(C, A)$

$C(C) + E(B) \geq D_{max}(C, B)$

$C(C) + E(C) \geq D_{max}(C, C)$

On peut déduire donc la règle générale, S étant l'ensemble des types d'entités de l'application :

$$\forall a \in S, \forall b \in S, C(a) + E(b) \geq D_{max}(a, b)$$

Ces contraintes définissent un système d'inéquations linéaires générant un ensemble de solutions. Notre but est de trouver la solution optimale qui minimise le nombre d'interactions inutiles entre les entités. Sachant que le nombre d'interactions dépend du nombre de collisions

entre les zones de conscience et les zones d'effet et donc dépend de l'aire de ces zones, notre but sera par suite de minimiser cette aire :

$$AZ = \pi \cdot \sum_{n \in S} (C(n)^2 + E(n)^2)$$

Donc le système d'inéquations sera résolu en rajoutant la condition de minimisation de AZ. Ce qui nous donnera la solution optimale demandée.

Ce système peut être résolu à l'aide de plusieurs solveurs. Nous avons utilisé la méthode de Newton pour la résolution de systèmes d'équations et d'inéquations [Psh70]. Des exemples de détermination de taille de zones seront donnés dans la partie expérimentale.

1.3 Conclusion

La gestion d'effet permet l'expression de la manifestation, elle permet également l'établissement de relations asymétriques en une seule étape à travers la définition de zones d'effet et de conscience. La gestion d'effet permet aussi de déterminer l'existence de la conscience séparément dans différents médias. Nous avons mené une étude pour la détermination des tailles des zones de conscience et d'effet visuel. La détermination des tailles de zones dans le média sonore et autres est prévue dans la continuation des travaux.

2

Le système

2.1 La mise en œuvre

La gestion d'effet est mise en œuvre dans ASSET, un système facilitant le développement de systèmes de réalité virtuelle pour la télérobotique [ETJH07].

2.1.1 ASSET

ASSET (Architecture pour des Systèmes de Simulation et d'Entraînement en Téléopération) [RJT02, Rod03] a été développé par Nancy Rodriguez durant sa thèse dans l'équipe VORTEX à l'IRIT. ASSET est un outil spécialisé dans la construction rapide des systèmes de réalité virtuelle pour la télérobotique. Il a été développé en Java et Java3D.

ASSET définit une architecture générale pour les systèmes de téléopération. C'est un système modulaire orienté-objet ; son architecture est composée de trois modules : le module d'interaction avec l'utilisateur (Gestionnaire Utilisateur), le module de contrôle du système réel (Gestionnaire Système Réel) et le module central (Administrateur) qui coordonne les entités participantes, utilisateurs et robots. Dans le Gestionnaire Utilisateur, on trouve les dispositifs d'interaction et de visualisation, le simulateur et la gestion des communications et des événements. Le Gestionnaire Système Réel est constitué d'une façon similaire mais, à la place des dispositifs d'interaction et de visualisation, on a des capteurs et des effecteurs. Il gère aussi la gestion de la cohérence entre l'état réel et l'état simulé pour mettre à jour les simulations des utilisateurs. L'administrateur a deux composants qui s'occupent de coordonner les interactions entre les entités et de

communiquer avec le système réel.

2.1.2 Réorientation d'ASSET

Le but initial d'ASSET concernait le développement de systèmes de réalité virtuelle mono-utilisateur distribués orientés vers la télérobotique. Nos besoins de recherche exigent un système de réalité virtuelle distribuée totalement indépendant de la télérobotique. Pour cette raison, on n'a réutilisé que deux modules d'ASSET : le gestionnaire utilisateur et l'administrateur. On a étendu le fonctionnement de l'administrateur pour qu'il puisse gérer une application d'environnements virtuels distribués. Ainsi ASSET est devenu un système d'EVDs client/serveur.

2.1.3 L'architecture de contrôle

On a choisi d'utiliser une architecture client/serveur parce que ce type d'architecture permet un meilleur contrôle (authentification, déroulement de la simulation, persistance...). Ce qui est très important pour les applications cibles : les simulations à grande échelle ou les MMOGs. De plus, la mise en œuvre du filtrage est plus facile et naturelle dans une architecture centralisée.

2.1.4 La communication

L'envoi, le traitement et la réception des messages constituent la plus grande partie du travail d'un système d'EVDs. Ils consomment évidemment le plus de ressources et de temps de traitement. D'où la nécessité d'alléger la couche réseau du système et de l'adapter aux besoins d'interactivité, de passage à l'échelle et d'immersion. Ces besoins exigent :

- la rapidité de transfert des messages ;
- la fiabilité de certains messages.

Pour ces raisons, nous allons étudier le protocole de communication utilisé et détailler les solutions adoptées pour répondre aux besoins de l'application.

2.1.4.1 Protocole de communication

ASSET utilise UDP comme protocole de communication pour deux raisons : la rapidité du transfert de messages fournie par UDP et sa compatibilité avec le multicast. En fait, puisque

l'utilisation du multicast dans certaines de nos communications était prévue, l'adoption de l'UDP est devenue indispensable dans ASSET.

D'autre part, l'inconvénient principal d'UDP est sa non-fiabilité mais cette non-fiabilité peut être tolérée dans la majorité de nos communications. En fait, dans un système d'EVDs la majorité des messages échangés sont des messages de mise à jour et la perte de ces messages peut être tolérée au cas où ces messages ne dépendent pas les uns des autres. Cette indépendance est exprimée dans la notion de protocole sans état ("stateless"). Ce type de protocole permet de ne pas maintenir une relation entre les différentes requêtes échangées entre les différents composants. Chaque requête ou message n'a aucune relation avec ses précédents.

Par contre, la perte des messages d'authentification, du lancement de la simulation, de collisions de zones ne peut pas être tolérée. Pour résoudre ce problème on adopte deux types de communication : fiable pour les messages critiques et non fiable (et donc rapide et simple) pour les messages non critiques. Pour cette raison, nous avons fiabilisé UDP selon nos besoins ; les caractéristiques de cette fiabilisation sont détaillées dans la section suivante.

2.1.4.2 Fiabilisation d'UDP

Notre fiabilisation répond à nos besoins de s'assurer qu'un message critique est arrivé. Garantir l'ordre des messages n'est pas important parce que nos messages fiables ne sont pas fréquents. De plus, tous nos messages ont une taille inférieure à celle d'un datagramme par suite il n'y a pas de division de messages en plusieurs datagrammes et donc il n'y a pas de besoin de contrôler l'ordre des messages.

Notre fiabilisation utilise le principe d'accusé de réception positif, c'est-à-dire qu'à chaque fois qu'un message fiable arrive à son destinataire, le destinataire renvoie un accusé de réception à l'expéditeur. Si après un certain délai, l'expéditeur n'a pas reçu l'accusé de son message, il renvoie le message de nouveau. Le renvoi de messages est répété autant de fois que le concepteur l'a décidé.

En général, trois scénarios d'envoi de message fiable sont possibles (figure 2.1) :

1. L'expéditeur envoie le message, le destinataire le reçoit ; alors il lui envoie un accusé de réception que l'expéditeur reçoit.

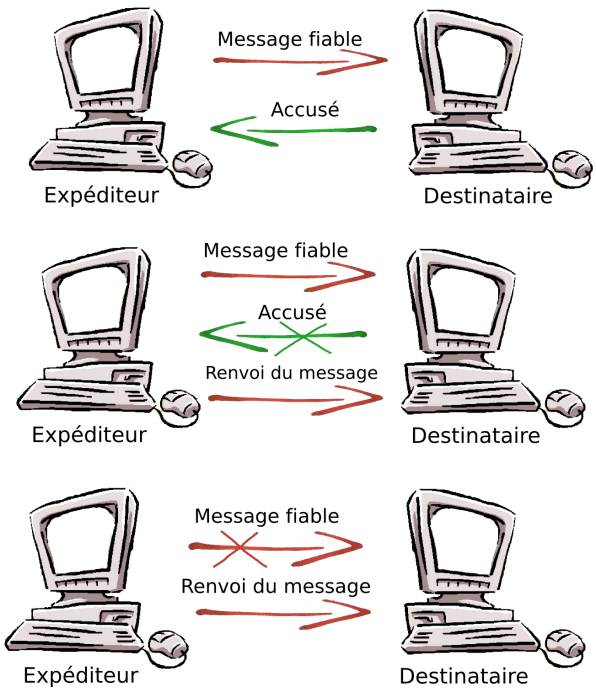

FIG. 2.1 – Scénarios

2. L'expéditeur envoie le message, le destinataire le reçoit. Il lui envoie un accusé de réception, mais l'accusé n'est pas reçu alors l'expéditeur renvoie le message. Le destinataire ignore le message mais envoie l'accusé de réception.

3. L'expéditeur envoie le message, le message n'est pas reçu alors l'expéditeur renvoie le message et attend de nouveau l'accusé de réception.

Ces trois scénarios sont traités dans l'algorithme de fiabilisation suivant :

L'expéditeur envoie le message fiable avec un entête contenant un compteur de message (un entier sur 32 bits). Après l'envoi, l'expéditeur rajoute le message à une liste de messages fiables envoyés et non accusés. À ce moment, un timer chargé d'attendre l'accusé de réception est lancé. Si l'accusé est reçu, le timer est arrêté et le message est éliminé de la liste. Par contre, si le temps s'écoule et que l'accusé de réception n'est pas reçu alors le timer est arrêté, le message est réexpédié et un nouveau timer est lancé. Si au bout de cinq envois successifs, aucun accusé n'a été reçu, on considère que la communication avec le destinataire n'est pas possible et on ne lui envoie plus de messages.

De l'autre côté, quand le destinataire reçoit un message fiable, il renvoie un accusé de réception non fiable. Cet accusé a comme identificateur l'identificateur du message. Le destinataire

sauvegarde les accusés qu'il envoie dans sa liste d'accusés. Cette liste lui permet de vérifier la présence préalable d'un message reçu. Ainsi, dans le cas d'une réception double (scénario 2), le destinataire pourra détecter cette duplication et ne traitera pas le message une deuxième fois.

2.1.4.3 La fiabilisation et la couche de communication

Dans un scénario classique, un système écoute en permanence l'arrivée de messages mais il n'envoie des messages que lorsqu'il en a besoin. La réception des messages est faite dans un processus léger (thread) qui écoute en permanence à l'aide de la fonction bloquante *receive()*. Le processus léger ne peut pas envoyer des messages parce qu'il est monopolisé par la fonction bloquante *receive()*. Ainsi l'envoi est séparé de la réception, les deux opérations se font au travers de numéros de port différents (l'utilisation du même numéro de port par deux sockets différentes n'étant pas possible).

Malheureusement, cette conception n'est pas adaptée à la fiabilisation : Quand un destinataire reçoit un message fiable, il envoie l'accusé de réception à l'adresse expéditrice[5]. Mais comme dans notre cas l'adresse expéditrice est réservée à l'envoi, l'accusé est donc envoyé vers un port non écouté. Et puisque la fiabilisation est effectuée dans la couche de communication, le processus léger récepteur ne sait pas que chaque correspondant possède deux adresses, une réservée à la réception et l'autre à l'envoi. Pour cette raison, le destinataire ne peut envoyer son accusé qu'à l'adresse expéditrice réservée à l'envoi.

Plusieurs solutions sont envisageables :

1. Le processus qui envoie un message fiable attend l'accusé de réception de ce message. Or cette attente qui peut être faite à l'aide de la fonction *receive()* bloque l'application jusqu'à la réception de l'accusé. Ainsi ce blocage ralentit l'application et peut provoquer des problèmes d'interblocage entre deux processus qui essaient de s'envoyer des messages fiables en même temps.

2. La deuxième solution est l'utilisation des communications asynchrones et non bloquantes du paquetage java NIO. NIO permet une écoute non bloquante des messages, ce qui rend l'envoi et la réception sur la même socket (et donc la même adresse) possible. Ainsi il n'y

[5] Adresse expéditrice de la socket = son adresse IP + son numéro de port

aura plus de problème d'adressage, un destinataire pourra envoyer l'accusé de réception à l'adresse expéditrice de son message.

2.1.4.4 NIO

Le nouveau paquetage **java.nio** (New Input Output) présente une nouvelle approche des entrées sorties par rapport au paquetage traditionnel **java.io**. La différence essentielle est que **IO** utilise les flux (*streams*) pour le transfert de données octet par octet alors que **NIO** utilise les tampons (*buffers*) pour un transfert bloc par bloc. Avec **NIO**, chaque opération produit et consomme un bloc de données par étape. Le traitement des données par bloc est beaucoup plus rapide que le traitement des octets streamés mais l'utilisation des entrées sorties orientées blocs est plus compliquée que celle des entrées sorties orientées flux.

Les canaux et les tampons sont les objets principaux de **NIO**. Les canaux sont analogues aux flux : toutes les données doivent passer par un objet canal (*channel*). Un tampon est un conteneur de données : les données qui doivent être envoyés sont mises dans un tampon. Et le canal envoie ces données à partir du tampon. De l'autre côté, un autre tampon est créé pour que les données soient reçues dedans.

De plus, **NIO** permet l'établissement de plusieurs canaux sur la même adresse en utilisant un multiplexeur de canaux. Ces multiplexeurs (*selectors*) sont utilisés pour des opérations d'entrée/sortie asynchrones, dans le cas notamment d'applications client/serveur. Le multiplexeur choisit le canal qui produit un événement de lecture ou d'écriture à l'aide des clés correspondant aux canaux sur lesquels s'est produit l'opération d'entrée/sortie (figure 2.2).

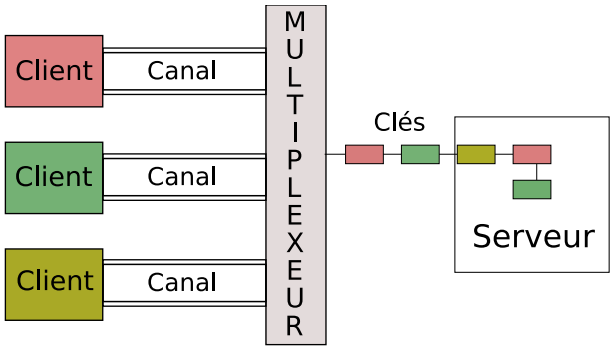

FIG. 2.2 – NIO

Ainsi, **NIO** permet la création d'un processus léger qui effectue l'envoi et la réception à

travers le même numéro de port et donc permet la fiabilisation de la communication.

2.1.5 Configuration du système

Plusieurs composants du système doivent être configurés avant le lancement de la simulation. Ces composants sont les entités, les dispositifs d'interaction, l'environnement, le filtrage et la communication.

Les entités Les entités participantes possèdent des tailles et des comportements différents. Elles ont des positions initiales distribuées dans l'espace. Les caractéristiques des types d'entités sont sauvegardées dans un fichier de configuration spécifique aux entités. Le serveur et les clients chargent ce fichier à leur lancement.

Les dispositifs d'interaction Une entité peut avoir un comportement autonome ou peut être gérée par un dispositif d'interaction. Ceci est précisé dans le fichier de configuration du client. Si l'entité est gérée par un dispositif d'interaction. Un processus spécialisé capte les ordres provenant du dispositif pour les appliquer sur l'état de l'entité. Sinon, dans le cas où l'avatar est autonome, son comportement prédéfini gère son déplacement dans la scène.

L'environnement L'environnement est limité par des frontières, il peut être divisé en régions selon la nature de l'application. Les propriétés de l'environnement sont spécifiées dans le fichier de configuration du serveur qui les communique aux clients qui en ont besoin.

Le filtrage Notre système peut fonctionner sans filtrage ou avec la gestion d'effet. On a également mis en œuvre le modèle spatial d'interaction et la gestion par zone d'intérêt pour les comparer à la gestion d'effet.

La communication Le serveur configure sa couche réseau en se basant sur un fichier propriétés contenant son adresse et son numéro de port qui sont fixés à l'avance. Cette adresse est fixée à l'avance pour que les clients distants puissent la connaître et se connecter au serveur. Par contre, les clients peuvent être lancés de n'importe quelle adresse.

2.2 Fonctionnement de la simulation

2.2.1 Lancement du serveur

Quand un serveur est lancé, il lit les fichiers de configuration et de propriétés pour configurer les modèles d'entités, les caractéristiques de l'environnement et sa couche de communication. Il se met ensuite en attente de la demande de connexion de clients.

2.2.2 Connexion d'un client

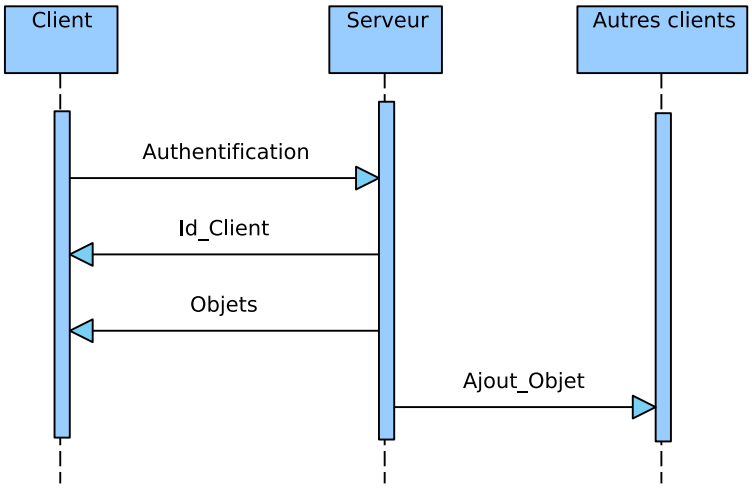

FIG. 2.3 – Connexion d'un client

Quand un client est lancé, il lit également les fichiers de configuration et de propriétés pour configurer les modèles d'entités et pour connaître son type d'entité, son dispositif d'interaction et l'adresse du serveur (figure 2.3). Il envoie ensuite une demande d'enregistrement au serveur qui lui renvoie son identificateur. Le serveur calcule alors les intersections entre la zone de conscience de la nouvelle entité et les zones d'effet des entités existantes pour déterminer lesquelles elle est capable de percevoir. Il lui envoie ensuite la liste de ces entités avec leurs états. Le serveur calcule aussi les intersections entre la zone d'effet de la nouvelle entité et les zones de conscience des autres pour notifier les clients qui sont capables de la percevoir.

2.2.3 Déroulement de la simulation

Durant la simulation, quand une entité A interagit dans le monde et change d'état, elle envoie un message de mise à jour au serveur (figure 2.4). A ce moment, le serveur met à jour l'état de A et fait des calculs d'intersection entre les zones de conscience et les zones d'effet pour savoir quels types de messages il doit envoyer et à qui. Le but de ces calculs est de déterminer :

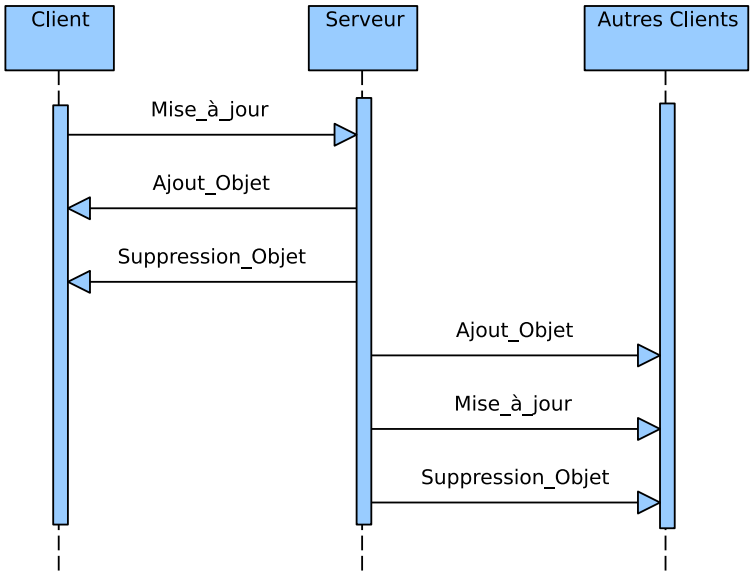

FIG. 2.4 – Mise à jour de l'état d'une entité

1. l'intérêt de A envers les autres entités. Le serveur calcule l'intersection entre la zone de conscience de A et chacune des zones d'effet des autres entités avant et après la mise à jour. Ce calcul sert à détecter les changements d'intérêt de A et envoyer les messages correspondants. Le tableau 2.5 montre les quatre cas possibles :

	Conscience préalable	Conscience présente	Message envoyé
Cas 1	Oui	Oui	Rien
Cas 2	Oui	Non	Suppression
Cas 3	Non	Oui	Ajout
Cas 4	Non	Non	Rien

FIG. 2.5 – Conscience du côté de A

   - Si la conscience de A n'a pas changé (cas 1 et 4), le serveur ne lui envoie aucun message.
   - Si A était consciente d'une entité et à cause de son déplacement elle ne la voit plus (cas 2), le serveur lui envoie un message pour supprimer cette entité.

– Si A n'était pas consciente d'une autre entité et maintenant elle est capable de la percevoir (cas 3), le serveur lui envoie un message pour rajouter cette entité.

2. Le serveur détermine également l'intérêt de chacune des autres entités vis-à-vis de A en calculant l'intersection entre leurs zones de conscience et la zone d'effet de A (tableau 2.6) :

	Conscience préalable	Conscience présente	Message envoyé
Cas 1	Oui	Oui	Mise à jour
Cas 2	Oui	Non	Suppression
Cas 3	Non	Oui	Ajout
Cas 4	Non	Non	Rien

FIG. 2.6 – Conscience du côté des autres entités

– Si l'entité était consciente de A et elle continue de l'être (cas 1), le serveur lui envoie une mise à jour.
– Si l'entité était consciente de A mais elle ne l'est plus après son déplacement (cas 2), le serveur lui envoie un message de suppression.
– Si l'entité n'était pas consciente de A mais elle l'est devenue (cas 3), le serveur lui envoie un message de rajout.
– Si l'entité n'était pas consciente de A et ne l'est toujours pas (cas 4), le serveur ne lui envoie aucun message.

Il faut noter que ces trois types de messages (ajout, mise à jour et suppression) sont envoyés en mode non fiable pour maintenir la rapidité de l'application. Ainsi, la perte de l'un de ces messages ne doit pas causer un problème pour la suite de la simulation. Pour ceci, le traitement des messages reçus prend en compte la possibilité d'une perte préalable de messages. Par exemple, si un utilisateur reçoit un ajout d'une entité qui existe déjà, il ne prend en compte que la nouvelle position de l'entité. S'il reçoit une mise à jour d'une entité qu'il ne connaît pas, il la rajoute en se basant sur les coordonnées de la mise à jour. Et s'il reçoit un message de suppression d'une entité qu'il ne connaît pas, il ignore le message.

2.2.4 Déconnexion d'un utilisateur

Lorsqu'un utilisateur quitte la simulation, il notifie le serveur qui notifie à son tour les utilisateurs conscients de l'entité.

2.3 Architecture client/multi-serveurs

L'architecture client/serveur permet un passage à l'échelle jusqu'à un nombre limité d'utilisateurs. Cette limite correspond à la capacité de traitement du serveur. Au delà de cette limite, le serveur aura des difficultés pour gérer toutes les requêtes des clients, ce qui ralentit le système et le rend moins fiable. En effet, il est impossible d'avoir un contrôle de flux avec UDP et donc les requêtes envoyées par les clients peuvent être détruites si le serveur ne les traite pas assez rapidement.

Pour éviter le goulot d'étranglement de l'architecture client/serveur et pouvoir monter en échelle, on a utilisé une architecture client/multi-serveurs. La distribution de la charge chez les différents serveurs se base sur une division de l'espace en plusieurs régions. Chaque serveur est responsable des utilisateurs d'une région dans tous les médias utilisés. Les utilisateurs envoient alors leurs de mises à jour au serveur de leur région. Réciproquement, le serveur leur envoie les données dont ils ont besoin. Un serveur communique également avec les serveurs des autres régions pour recevoir les données des entités externes qui intéressent ses clients.

Un méta serveur est responsable de la division de l'espace, de l'authentification et de la coordination entre les serveurs.

Pour gérer les intérêts des entités au sein de l'environnement, cette architecture possède deux couches de filtrage :

- Une couche de filtrage intra-régional : Le serveur est responsable du filtrage à l'intérieur de sa région. Il envoie à ses entités les données qui les intéressent comme dans l'architecture mono-serveur.
- Une couche de filtrage inter-régional : Les serveurs communiquent entre eux pour récupérer les données externes qui intéressent leurs clients.

2.3.1 La couche de filtrage inter-régional

Cette couche gère les relations d'intérêt entre les régions et donc entre les serveurs. Une relation d'intérêt entre les régions existe si une intersection entre la zone de conscience appartenant à une entité d'une région chevauche la zone d'effet d'une entité appartenant à une autre région. Pour que cette intersection ait lieu, il faut au moins que l'une des deux zones chevauche

la frontière séparant les deux régions (figure 2.7).

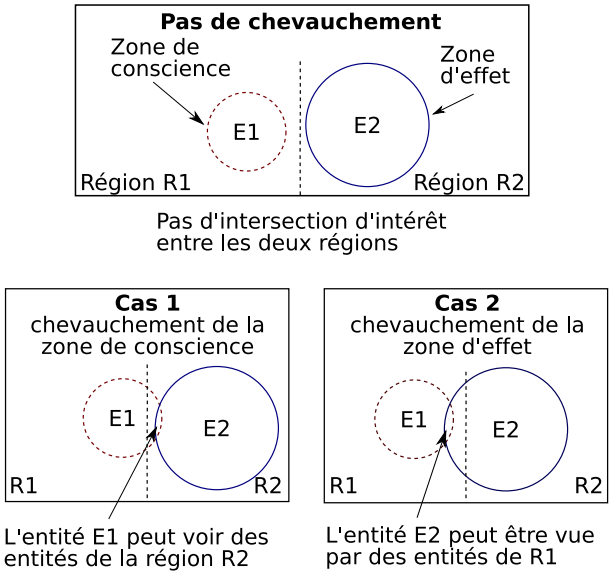

FIG. 2.7 – Relations d'intérêt inter-serveur

L'algorithme du filtrage inter-régional est donc basé sur la propriété suivante :

Une entité *E1* appartenant à la région *R1* est intéressée par une entité *E2* appartenant à la région *R2* si au moins une des conditions suivantes est vraie :

- La zone de conscience de *E1* chevauche *R2*.
- La zone d'effet de *E2* chevauche *R1*.

Étudions ces deux cas :

- **Cas 1** : Quand le serveur *S1* se rend compte que la zone de conscience de son entité *E1* chevauche la région du serveur *S2*, *S1* notifie *S2*. *S2* lui envoie à ce moment une mise à jour agrégée de toutes ses entités. *S1* vérifie si *E1* est intéressée par des entités de *S2*. Si c'est le cas, il informe *S2* qui commence à lui envoyer des mises à jour des entités intéressantes. *S1* redirige ensuite ces messages à *E1*.
- **Cas 2** : Quand le serveur *S2* se rend compte que la zone d'effet de son entité *E2* chevauche la région du serveur *S1*, *S2* notifie *S1*. *S1* vérifie si l'une de ses entités est intéressée par *E2*. Si c'est le cas *S1* exprime son intérêt à *S2* qui commence ainsi à lui envoyer des mises à jour des entités intéressantes. Si aucune des entités distantes n'est intéressée par son entité, *S2* continue à envoyer des mises à jour non fréquentes (*heartbeats*) au cas où une des entités devient intéressée au cours de la simulation.

Si les deux zones chevauchent la frontière, le premier chevauchement active la conscience et le deuxième n'aura aucun effet (figure 2.8).

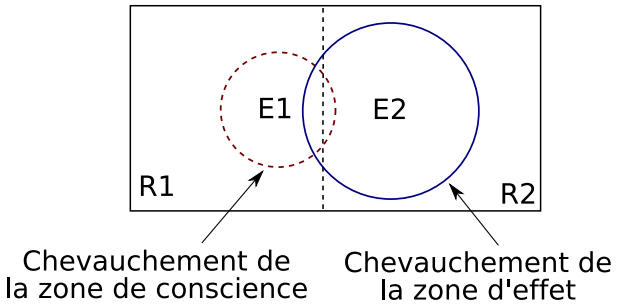

FIG. 2.8 – Intersection des deux zones avec la frontière

Si un serveur perd son intérêt dans une entité d'un autre serveur, il notifie son serveur qui arrête alors de lui envoyer les mises à jour de l'entité.

2.3.2 Fonctionnement de la simulation

2.3.2.1 Connexion d'un serveur

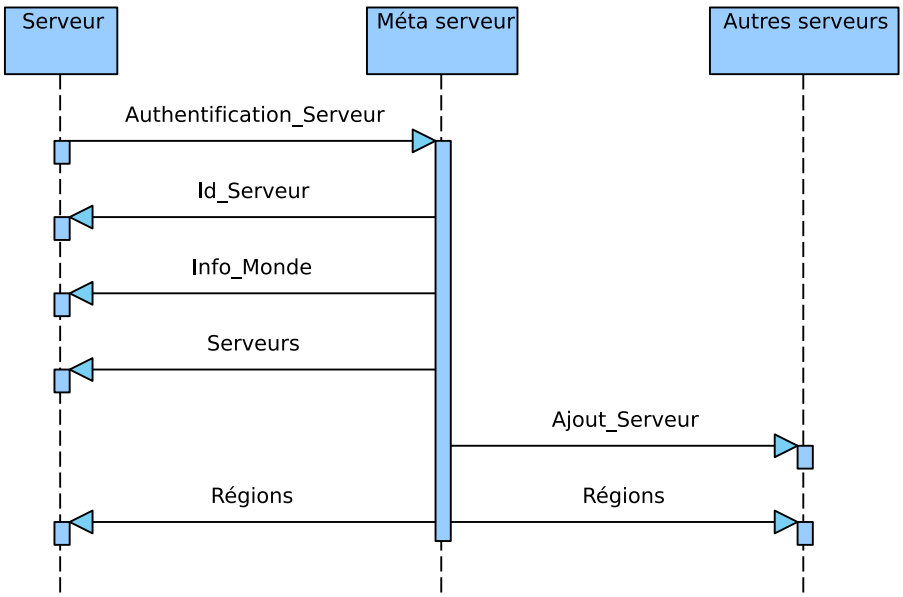

FIG. 2.9 – Connexion d'un serveur

Quand un serveur est lancé, il envoie un message d'enregistrement au méta serveur qui lui renvoie son identificateur, les informations sur l'environnement, le filtrage et les coordonnées des serveurs déjà connectés (figure 2.9). Il envoie aussi aux autres serveurs les coordonnées du

nouveau serveur. Le méta serveur redivise ensuite l'espace et envoie à tous les serveurs cette nouvelle division et la répartition des régions chez les serveurs.

2.3.2.2 Connexion d'un client

A son lancement, le client contacte le méta serveur qui lui renvoie son identificateur, les informations sur l'environnement et les coordonnées du serveur responsable de sa région. Le client contacte alors son serveur qui calcule son intérêt suivant sa zone de conscience (figure 2.10). Ce calcul est utile pour :

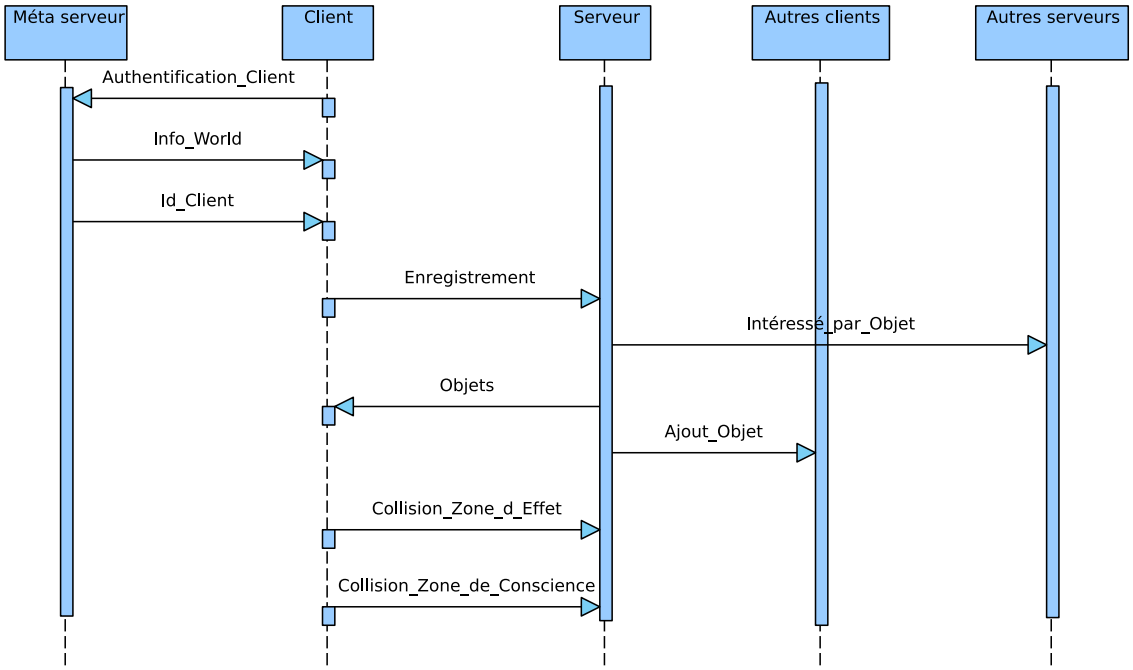

FIG. 2.10 – Connexion d'un client

- savoir si l'entité s'intéresse à des entités externes. Dans ce cas là, le serveur contacte les serveurs en charge des entités intéressantes.
- lui envoyer l'ensemble des entités qu'elle est capable de percevoir.

Le serveur détermine également lesquels de ses clients sont intéressés par la nouvelle entité et il les notifie.

D'autre part, le client vérifie si sa zone de conscience ou sa zone d'effet chevauche des régions externes et il notifie son serveur dans ces cas là.

2.3.2.3 Collision de la zone d'effet

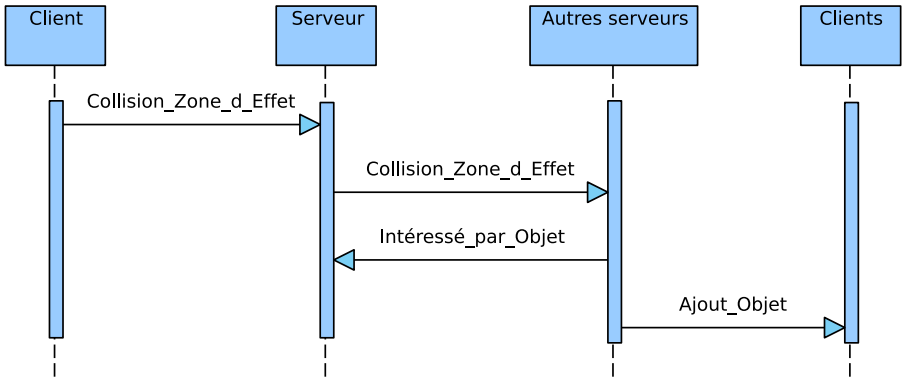

FIG. 2.11 – Collision de la zone d'effet

Si le client se rend compte que sa zone d'effet chevauche une région voisine, il envoie une notification à son serveur (figure 2.11). Le serveur notifie à son tour le serveur de la région correspondante et lui envoie les coordonnées de l'entité chevauchante. Si ce dernier constate qu'il a des clients intéressés par l'entité, il exprime son intérêt au serveur en charge pour qu'il lui envoie des mises à jour de cette entité et il envoie un message d'ajout aux clients intéressés.

2.3.2.4 Collision de la zone de conscience

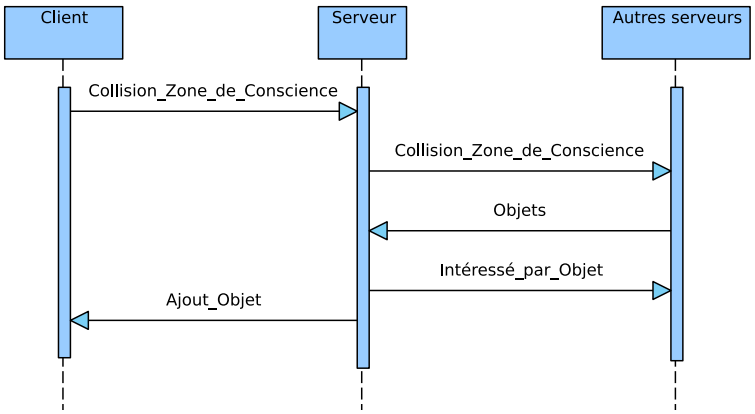

FIG. 2.12 – Collision de la zone de conscience

De même, si le client se rend compte que sa zone de conscience chevauche une région voisine, il envoie une notification au serveur (figure 2.12). Le serveur notifie le serveur de la région correspondante, qui lui renvoie les états de ses entités. Si l'une ou plusieurs des entités voisines

intéresse le client, son serveur envoie une expression d'intérêt au serveur de la région voisine et envoie ensuite au client les informations concernant les entités intéressantes.

2.3.2.5 Mise à jour

La gestion d'intérêt à travers les régions complique la mise à jour de l'état d'une entité chez les entités intéressées. Quand l'entité change d'état, elle envoie une mise à jour à son serveur. Le serveur calcule alors l'intérêt préalable et présent de l'entité et décide quels messages il va envoyer et à qui comme il le faisait dans un système mono-serveur (cf. 2.2.3). De plus le serveur envoie la mise à jour aux serveurs des clients externes intéressés par l'entité (figure 2.13).

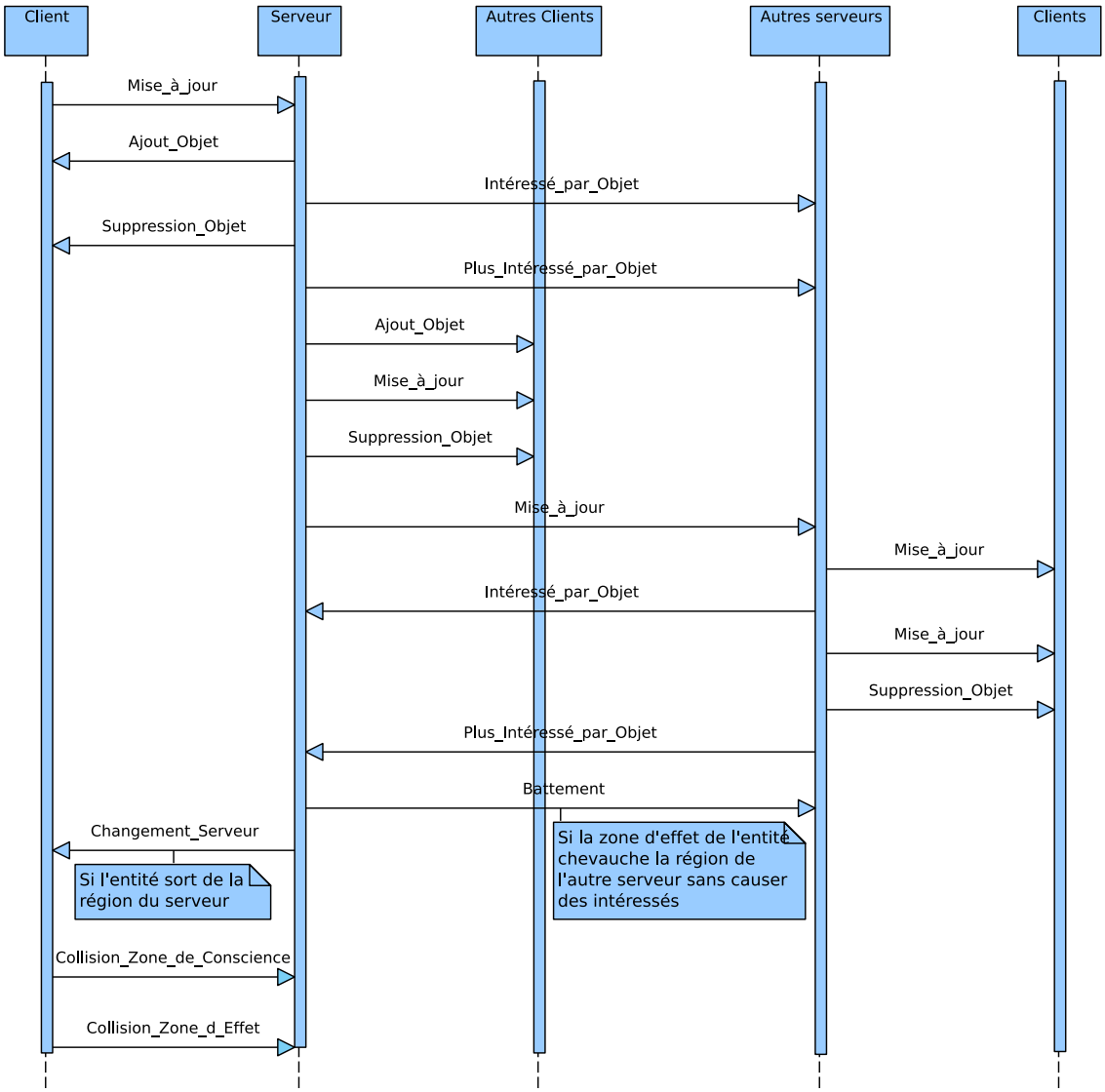

FIG. 2.13 – Collision de la zone de conscience

Si la zone d'effet de l'entité chevauche une région voisine qui n'a aucun client intéressé, le serveur envoie à l'autre région des messages de notification non fréquents (cf. 2.3.2.7).

D'autre part, si le serveur se rend compte que l'entité a quitté sa région pour rejoindre une autre région, il notifie l'entité et inclut dans la notification les coordonnées de son nouveau serveur. Les détails de l'opération de changement de serveur sont exposés dans la section suivante.

A chaque pas de simulation, le client vérifie si sa zone de conscience ou sa zone d'effet chevauche une région qu'elle ne chevauchait pas avant. Dans ce cas là, il notifie son serveur (figures 2.11, 2.12).

2.3.2.6 Changement de serveur

Quand une entité traverse la frontière de sa région, son serveur lui envoie un message avec les coordonnées de son nouveau serveur (figure 2.14). Le client contacte son nouveau serveur pour le rejoindre. Le nouveau serveur calcule alors les intérêts de sa nouvelle entité et contacte les serveurs des entités externes qui l'intéressent. Il informe ensuite l'ancien serveur qu'il est maintenant en charge de l'entité.

Si l'ancien serveur a des entités intéressées par son ancienne entité, il notifie le nouveau serveur. Il contacte aussi les serveurs des entités qui intéressaient son entité pour qu'ils arrêtent de lui envoyer des messages de mise à jour. Il dit aussi aux serveurs externes intéressés par son entité qu'il n'est plus responsable de cette entité et il leur donne les coordonnées de son nouveau serveur pour qu'elles réclament des mises à jour de ce nouveau serveur.

De plus, l'entité notifie son nouveau serveur des collisions de sa zone d'effet et de sa zone de conscience avec les régions externes pour qu'il fasse ce qui convient.

2.3.2.7 Messages de pulsation

Si un serveur est en charge d'une entité dont la zone de conscience ou d'effet chevauche avec une autre région et cette entité n'intéresse aucun client de l'autre région, son serveur envoie des mises à jour non fréquentes de l'état de cette entité (appelés messages de pulsation ou "heartbeats"). Ces mises à jour sont utiles au cas où le changement de l'état de l'entité implique un changement de l'intérêt des clients distants.

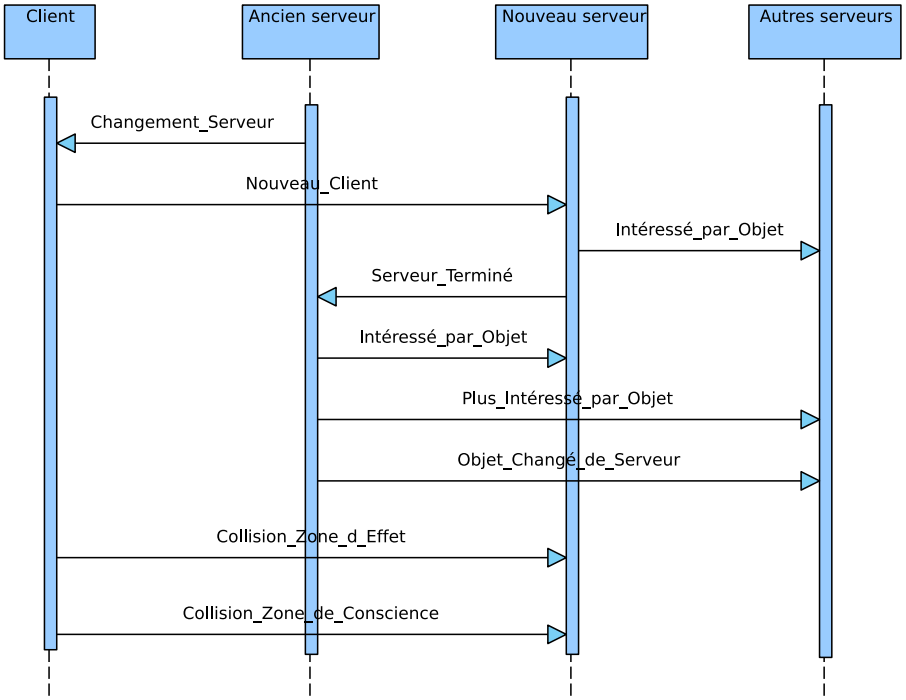

FIG. 2.14 – Changement de serveur

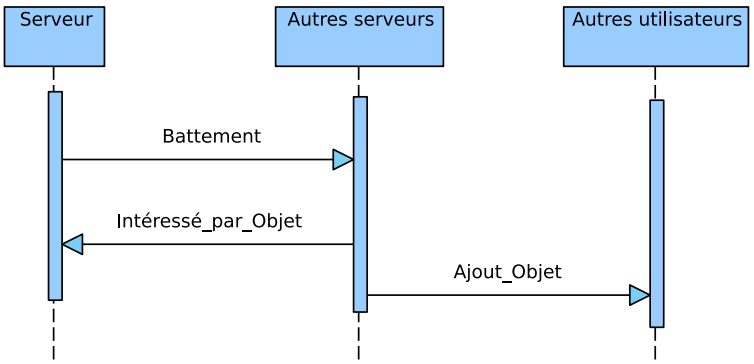

FIG. 2.15 – Messages de pulsation

2.4 L'utilisation du multicast

Comme on a déjà vu, le multicast est un mode de transmission très adapté aux EVDs. En effet, un message de mise à jour est en général envoyé à plusieurs destinataires. L'envoi de cette mise à jour par multicast implique un transfert unique du message au lieu de répéter ce transfert autant de fois qu'il y a de destinataires. Ce transfert commun permet un gain des ressources réseau et un gain de calcul chez les expéditeurs qui, une fois qu'ils ont l'adresse d'un groupe multicast, n'auront plus besoin de calculer la liste des expéditeurs intéressés par chaque message.

2.4.1 L'attribution des groupes multicast

L'attribution des groupes multicast au niveau des utilisateurs ou des serveurs dépend de la méthode de filtrage adoptée par l'application. Les mises à jour peuvent être envoyées à travers des groupes multicast attribués soit aux entités, à des groupes d'entités ou aux régions.

Si les groupes multicast sont attribués aux entités, chaque utilisateur s'abonne aux groupes multicast des entités qui l'intéressent et ne recevra que les informations qui lui sont pertinentes. Par contre, cette attribution demande un grand nombre de groupes multicast surtout pour les simulations à grande échelle. De plus, les utilisateurs changent aussi fréquemment de groupes multicast qu'ils changent d'intérêt. Ce changement peut-être très fréquent et donc très coûteux.

Les groupes multicast peuvent être attribués à des groupes d'entités formés en fonction de propriétés spatiales, fonctionnelles ou temporelles. De cette manière, le nombre de groupes multicast utilisés est restreint et les entités ne recevront que les messages qui les intéressent. Mais si les entités peuvent appartenir à plusieurs groupes en même temps, des réceptions en double pourront avoir lieu. En effet, si deux entités appartiennent à deux groupes communs, quand une entité envoie ses mises à jour, l'autre entité les recevra deux fois à travers les deux groupes. D'autre part, imposer aux entités d'appartenir à un seul groupe oblige les entités à choisir une propriété prioritaire. Les entités ne pourront donc pas envoyer ou recevoir des messages correspondants à plusieurs propriétés.

Dans le cas de la division de l'espace en régions, l'attribution des groupes multicast aux régions est très naturelle. Le nombre de groupes multicast est restreint et le changement de groupes multicast dépend de la rapidité avec laquelle les entités changent de régions. Normalement, la

taille des régions est choisie en fonction de la rapidité de déplacement des entités : les entités sont censées ne pas changer fréquemment de régions.

Par contre, l'inconvénient le plus important de cette attribution est la réception de messages non pertinents par les utilisateurs. En fait, quand une entité devient intéressée par une autre, elle commence à recevoir ses messages à travers le groupe multicast de sa région. Ainsi, toutes les entités de la région recevront ces messages même si elles n'y sont pas intéressées. Ainsi, un utilisateur ne recevra pas que les messages qui l'intéresse mais aussi les messages qui intéressent chaque utilisateur de sa région.

En conclusion, l'attribution des groupes multicast aux entités ne permet pas un passage à l'échelle important à cause du nombre de groupes multicast nécessaire. Ce qui nous a poussé à ne pas utiliser cette méthode. D'autre part, la formation de groupes d'entités est intéressante mais elle implique soit une réception double de messages, soit un choix restrictif des priorités. Pour cette raison, nous ne l'avons pas adopté non plus. L'attribution des groupes multicast aux régions correspond à notre méthode qui utilise la division de l'espace en régions. Les groupes multicast seront attribués aux régions et donc aux serveurs de ces régions. Et puisque dans notre architecture, il y a deux types de communications (les communications client/serveur et les communications serveur/serveur), on peut utiliser le multicast comme mode de transmission sur les deux types de communication en associant à chaque serveur un groupe multicast pour chaque type de communications. Cependant, l'utilisation de ces groupes multicast dans chacune de ces communications a ses avantages et ses inconvénients, comme on le verra dans ce qui suit.

2.4.2 Le multicast dans les communications client/serveur

Si le multicast est utilisé pour les communications client/serveur, les clients joignent le groupe multicast du serveur de leur région. Le serveur enverra alors à ses clients les mises à jour qui les intéressent à travers ce groupe multicast.

Avantages du multicast dans les communications client/serveur

- Les clients d'un même serveur (qui sont des voisins dans la même région) sont souvent intéressés par les mêmes entités. En utilisant le multicast, le serveur envoie une seule fois le message à tous ses clients au lieu de l'envoyer autant de fois qu'il a de clients intéressés. Ceci économise la bande passante et le coût de traitement de l'envoi.
- Un client ne change d'abonnement multicast que lorsqu'il change de région. A ce moment, il quitte le groupe multicast de son ancien serveur pour rejoindre le groupe multicast du nouveau. Ainsi, le changement de groupes multicast n'arrive pas très souvent, sauf dans le cas où la taille des régions est petite par rapport à la rapidité de déplacement des entités. Mais en général, la taille des régions choisie permet un déplacement peu fréquent entre les régions.

Inconvénients du multicast dans les communications client/serveur

- Dès qu'un client est intéressé par une entité, son serveur commence à lui envoyer les messages concernant cette entité. Et comme le serveur communique les informations à ses clients à travers son groupe multicast, tous ses clients recevront les messages concernant cette entité. Ainsi un utilisateur ne recevra pas que les messages qui l'intéressent mais il recevra aussi les messages qui intéressent les autres utilisateurs de sa région. Ce qui va consommer ses ressources sans rien apporter de plus à la simulation.
- L'utilisation du multicast en dehors d'un réseau local n'est pas toujours disponible sur Internet, mais ceci pourrait être résolu par l'utilisation du multicast applicatif (cf. 1.4.3.3).

Ainsi, le choix de l'utilisation du multicast dépend du choix entre d'une part le gain des ressources réseaux et des ressources chez le serveur et d'autre part le gain de ressources chez les clients. Nous reviendrons sur les facteurs qui permettent la prise de cette décision dans l'analyse de la partie expérimentale.

2.4.3 Le multicast dans les communications serveur/serveur

Quand le multicast est adopté comme mode de transmission dans les communications serveur/serveur, un groupe multicast spécialisé est associé à chaque serveur. Un serveur s'abonne chez un autre quand l'un de ses clients est intéressé par une entité de ce deuxième serveur.

Avantages du multicast dans les communications serveur/serveur

Les entités très importantes intéressent les utilisateurs de toutes les régions et par suite de tous les serveurs. Leurs messages de mise à jour sont envoyés par leurs serveurs à tous les autres serveurs de l'application. L'utilisation du multicast permet un transfert unique de ces messages au lieu d'un transfert multiple.

Inconvénients du multicast dans les communications serveur/serveur

Deux serveurs peuvent être intéressés par deux entités différentes logées sur un troisième serveur. Mais comme ils s'abonnent tous les deux au même groupe multicast du troisième serveur, chacun d'eux recevra les mises à jour d'une entité qui n'intéresse aucun de ses clients.

2.5 Conclusion

Dans l'ensemble, les zones de conscience et d'effet permettent l'établissement d'un filtrage spatial. Ces zones peuvent exprimer également les propriétés fonctionnelles des entités. La gestion d'effet utilise aussi un filtrage temporel à travers l'envoi non fréquent des messages de mises à jour aux serveurs qui n'y sont pas immédiatement intéressés.

La gestion d'effet permet une expression séparée d'intérêt et de manifestation à travers la définition distincte des zones de conscience et d'effet. Cette propriété permet d'établir des relations asymétriques en une seule étape au lieu de deux étapes dans le modèle spatial d'interaction. De plus, cette expression d'intérêt et de manifestation se fait séparément dans chaque média utilisé. L'activation et la désactivation de ces médias est à la charge du développeur de l'application.

La gestion d'effet peut s'adapter aux architectures emboîtées en associant les régions de l'espace aux régions de l'architecture et puis en activant et désactivant la transparence entre ces régions dans les différents média selon la simulation. De plus, la gestion d'effet s'adapte parfaitement aux environnements ouverts puisqu'elle permet de gérer les intérêts des entités à travers les régions de l'environnement.

Pour le moment, notre système ne permet pas une division dynamique des régions fortement peuplées, mais la mise en œuvre de cette division est prévue dans la suite de ces travaux.

Notre système utilise une architecture client/serveur pour les applications à petite échelle et une architecture client/multi-serveurs pour les applications à grande échelle. L'architecture multi-serveurs est totalement transparente pour le déroulement de la simulation. Cette transparence est due à l'échange des informations concernant les entités intéressantes entre serveurs.

Enfin, notre système donne le choix de l'utilisation du multicast dans deux types de communications : les communications client/serveur et les communications serveur/serveur. Selon ses besoins, l'application peut utiliser le multicast ou l'unicast dans ces communications. Pour le multicast, les deux types de communication associent les groupes multicast aux serveurs et donc aux régions. Ce qui permet l'utilisation d'un nombre restreint de groupes multicast et un changement minimal de groupes. L'étude des avantages et des inconvénients de l'utilisation du multicast dans ces communications sera détaillée dans l'analyse de la partie expérimentale.

3

Évaluation

Notre évaluation de la gestion d'effet couvre plusieurs aspects de son fonctionnement. Elle étudie premièrement la qualité et l'impact du filtrage effectué. Cette étude est complétée par une comparaison entre la gestion d'effet et d'autres méthodes de gestion d'intérêt qui utilisent des zones comme la gestion par zone d'intérêt et le modèle spatial d'interaction.

L'impact de l'architecture client/multi-serveurs sur le passage à l'échelle et la latence est ensuite analysé. Finalement, les avantages et les inconvénients de l'utilisation du multicast dans les communications client/serveur et serveur/serveur sont étudiés pour déduire les cas où le multicast est utile.

3.1 Environnement expérimental

Nous avons utilisé pour nos expériences 120 machines de la grapphe de l'INRIA à Sophia-Antipolis de Grid'5000 (cf. 3.1.1). Ces machines, équipées d'Opterons 246 (à 2 GHz) ou 275 (à 2.2 GHz) et de 2 à 4 Giga octets de mémoire, ont Rocks Linux 3.3.0 comme système d'exploitation. Elles sont connectées par un réseau Gigabit Ethernet doublé par un réseau Myrinet 2000 (2 Gbits/s), avec une latence inter noeuds observée de l'ordre de 0.1 ms.

3.1.1 Grid'5000

Le projet national GRID'5000 [CDD+05] vise à construire une plate-forme expérimentale de recherche en informatique, constituée d'une grille de calcul de grande taille avec l'objectif

d'atteindre le nombre symbolique de 5000 processeurs. Il s'agit donc de connecter via un réseau haut-débit (10 Gbits/s) une dizaine de grappes de grande taille distribuées sur neuf sites en France.

Cette initiative est soutenue par les principaux organismes de recherche en informatique, l'INRIA, le CNRS et par plusieurs universités. Le site de Toulouse a été l'un des premiers retenus parmi les 9 sites actuels (Bordeaux, Grenoble, Lille, Lyon, Nancy, Orsay, Rennes, Sophia-Antipolis et Toulouse) pour sa capacité à accueillir et à administrer une grappe de grande dimension et à coordonner des activités de recherche en liaison avec Grid'5000.

L'IRIT est très fortement impliqué dans le projet GRID-MIP qui regroupe les principaux acteurs locaux de la recherche dans le domaine des grilles de calcul : la Fédération FéRIA (IRIT, LAAS et ONERA), le CERFACS, ainsi que le groupement CalMip pour le calcul Haute Performance (constitué de 17 laboratoires), le CICT (qui héberge la grappe), le laboratoire de biotechnologie et d'amélioration des plantes et la Société R'Tech. Cette structure - qui regroupe les porteurs de projets - évolue rapidement avec des propositions émanant de chercheurs de divers organismes : le CIRIMAT, le LGC, le Laboratoire Biotechnologie et Bioprocédés et des collaborations avec des PME-PMI telles SeaNodes et QoS design ainsi que des grands comptes tels le CEA, le CNES, EADS, EDF, IFP.

3.1.2 L'environnement virtuel

Les simulations ont lieu dans un environnement ouvert de dimensions 2000x2000. Cet environnement est divisé en cellules lors de l'utilisation des architectures client/multi-serveurs. Chaque serveur est responsable d'une région comprenant un nombre de cellules contiguës. La répartition des cellules sur les serveurs est modifiée à chaque connexion d'un nouveau serveur.

3.1.3 Les caractéristiques des entités

Les propriétés des entités Pour avoir des simulations réalistes, les entités simulées sont de tailles différentes et possèdent des capacités de conscience variées.

Le comportement des entités Durant la simulation, les entités se comportent d'une manière autonome. Leur comportement est aléatoire suivant l'accélération, la valeur aléatoire de

l'accélération est comprise dans un intervalle. Cet intervalle détermine donc la vitesse moyenne de l'entité. Par contre, une entité a toujours plus d'accélérations positives que négatives ou l'inverse. Ce privilège de signe fait diriger l'entité vers une direction sans la ramener d'une manière rectiligne vers cette direction. Une entité change de direction lorsque :

- elle touche la frontière du monde. À ce moment elle rebondit dans le sens inverse de la collision et son privilège d'accélération est inversé pour qu'elle puisse repartir vers l'autre côté du monde.
- la direction privilégiée est changée par le générateur de comportement. Ceci est fait au plus tard toutes les 400 millisecondes d'un comportement stable.

3.1.4 Les simulations

Une expérience consiste en une simulation de deux minutes avec un pas de simulation de 200 millisecondes. Pendant une expérience, divers types d'informations sont collectés comme la latence, le nombre de messages échangés et la charge des clients et des serveurs. Les expériences sont faites en variant le nombre d'entités et/ou de serveurs. Chaque expérience a été répétée quatre fois pour calculer les moyennes des valeurs résultantes.

3.2 Évaluation du filtrage de la gestion d'effet

On a évalué en premier la qualité du filtrage fourni par la gestion d'effet en étudiant son impact sur le passage à l'échelle, les taux de communication et la latence.

3.2.1 Paramètres du filtrage

Pour effectuer cette évaluation, on a utilisé une architecture client/serveur. Les entités interagissent ensemble en utilisant le média visuel. Ces entités ont des tailles et des capacités de perception différentes. Ces entités profitent donc de la capacité de la gestion d'effet d'établir des relations asymétriques entre elles. Ce type de relations est indispensable parce que, par exemple, une entité grande ayant une vue limitée est perçue par beaucoup d'entités qu'elle n'est pas capable de voir.

Les entités simulées sont distribuées en quatre types : des entités ayant une taille et une vue

normales, des entités grandes ayant une vue normale, des entités grandes avec une vue avantagée et des entités de taille normale ayant une vue avantagée.

Pour pouvoir évaluer la gestion d'effet dans des conditions différentes, nous avons mené deux types d'expériences : des expériences où les entités ont des capacités de perception et de perceptibilité très différentes et des expériences où les entités sont plus homogènes et leurs différences ne sont pas très importantes.

3.2.1.1 Entités homogènes

Pour les entités homogènes, nous avons considéré qu'une entité ayant une vue avantagée possède un angle de vue de 120 degrés *(fov = 120)* et un écran de 1024 pixels de largeur *(NbPixelsX = 1024)*. Ceci nous donne l'angle couvert par un pixel (cf. 1.2.3) :

$$\begin{aligned} \theta &= fov/NbPixelsX \\ &= 120/1024 \\ &= 0.117 \end{aligned}$$

Comme cette entité a une vue avantagée, elle pourra voir les détails sur l'écran jusqu'à un pixel *(n = 1)*. Ainsi la capacité visuelle d'une entité avantagée est de :

$$\begin{aligned} V_{Avantagée} &= 1/\tan(n_{Avantagée} \cdot \theta/2) \\ &= 1/\tan(1 \cdot 0.117/2) \\ &= 1/\tan(0.058) \\ &= 977.84 \end{aligned}$$

Pour obtenir la distance maximale de perception qu'a une entité avantagée d'une entité normale par exemple, nous multiplions le rayon de l'entité normale (1.02 mètres) par la capacité visuelle de l'entité avantagée (cf. 1.2.3).

$$\begin{aligned} Dmax(Avantagée, Normale) &= R_{Normale} \cdot V_{Avantagée} \\ &= 1.02 \cdot 977.84 \\ &= 1000\ mètres \end{aligned}$$

Par contre, la distance maximale de perception qu'a l'entité avantagée d'une entité grande dépendra du rayon de cette entité grande :

$$\begin{aligned} Dmax(Avantagée, Grande) &= R_{Grande} \cdot V_{Avantagée} \\ &= 2.04 \cdot 977.84 \\ &= 2000\ mètres \end{aligned}$$

Les entités ayant une vue normale (non avantagée) pourront voir moins clairement sur l'écran. Nous avons considéré que leur seuil minimal de perception sera de deux pixels. Leur capacité visuelle sera donc de :

$$\begin{aligned} V_{Normale} &= 1/\tan(n_{Normale} \cdot \theta/2) \\ &= 1/\tan(2 \cdot 0.117/2) \\ &= 1/\tan(0.117) \\ &= 489.7 \end{aligned}$$

Ainsi, la distance maximale de perception qu'a une entité normale d'une entité grande est de :

$$\begin{aligned} Dmax(Normale, Grande) &= R_{Grande} \cdot V_{Normale} \\ &= 2.04 \cdot 489.7 \\ &= 1000\ mètres \end{aligned}$$

Ainsi, nous obtenons les distances maximales de perception qu'ont les couples de types d'entités (en notant que l'environnement est un carré de 2000 x 2000 mètres).

Le tableau 3.1 liste ces distances et montre par exemple que la distance maximale de perception qu'a une entité normale d'une entité grande est de 1000 mètres ou de 1 kilomètre .

D_{max}	Normale	Grande	Grande & Avantagée	Avantagée
Normale	500	1000	1000	500
Grande	500	1000	1000	500
Grande & Avantagée	1000	2000	2000	1000
Avantagée	1000	2000	2000	1000

FIG. 3.1 – Les distances maximales de perception des entités homogènes

Nous nous sommes basés sur les distances maximales de perception pour déterminer les tailles des zones de conscience et d'effet visuels de nos entités. Pour ceci, nous avons suivi la règle suivante (cf. 1.2.5) :

$$\forall a \in S, \forall b \in S, C(a) + E(b) \geq D_{max}(a, b)$$

Ce qui nous donne le système d'inéquations qui suit :

$C(Normale) + E(Normale) \geq 500$ (pour que deux entités normales se voient quand il le faut)

$C(Normale) + E(Grande) \geq 1000$ (pour qu'une entité normale voit une entité grande quand il le faut)

$C(Normale) + E(Grande\&Avantagée) \geq 1000$

$C(Normale) + E(Avantagée) \geq 500$

$C(Grande) + E(Normale) \geq 500$

$C(Grande) + E(Grande) \geq 1000$

$C(Grande) + E(Grande\&Avantagée) \geq 1000$

$C(Grande) + E(Avantagée) \geq 500$

$C(Grande\&Avantagée) + E(Normale) \geq 1000$

$C(Grande\&Avantagée) + E(Grande) \geq 2000$

$C(Grande\&Avantagée) + E(Grande\&Avantagée) \geq 2000$

$C(Grande\&Avantagée) + E(Avantagée) \geq 1000$

$C(Avantagée) + E(Normale) \geq 1000$

$C(Avantagée) + E(Grande) \geq 2000$

$C(Avantagée) + E(Grande\&Avantagée) \geq 2000$

$C(Avantagée) + E(Avantagée) \geq 1000$

Ce système est résolu en essayant de minimiser la somme des surfaces des zones de conscience et de zones d'effet. Cette condition permet de trouver la solution optimale qui minimise le nombre d'interactions inutiles entre les entités. La surface à minimiser est la suivante :

$$\begin{aligned} AZ &= \pi \cdot \sum_{n \in S} C(n)^2 + E(n)^2 \\ &= \pi \cdot (C(Normale)^2 + E(Normale)^2 + C(Grande)^2 + E(Grande)^2 + C(Grande\&Avantagée)^2 \\ &\quad + E(Grande\&Avantagée)^2 + C(Avantagée)^2 + E(Avantagée)^2) \end{aligned}$$

La résolution du systèmes d'inéquations nous a donné les zones de conscience et d'effet suivantes :

	Zone de conscience	Zone d'effet
Normale	250	250
Grande	250	1000
Grande & Avantagée	1000	1000
Avantagée	1000	250

3.2.1.2 Entités hétérogènes

Les entités hétérogènes sont caractérisées par des tailles et des capacités visuelles assez différentes. L'écart entre les tailles des entités est plus grand que celui qui existe entre les entités homogènes (de même pour la capacité visuelle). Ainsi, une entité de grande taille est assez importante pour être vue par la majorité des entités de l'environnement et une entité ayant une capacité visuelle avantagée est capable de voir des entités très distantes.

Les expériences avec les entités hétérogènes ont deux intérêts : d'une part l'étude des environnements ayant des entités fortement conscientes les unes des autres et d'autre part l'étude des environnements où les interactions asymétriques sont très présentes. En effet, l'augmentation des différences entre les capacités de perception et de perceptibilité augmente le besoin d'établissement de relations asymétriques puisque les entités grandes sont désormais très grandes alors

que la vue des entités normales, par exemple, n'a pas changée.

Pour agrandir l'écart entre les entités, nous avons gardé la capacité visuelle des entités avantagées qui sont toujours capables de voir jusqu'à un pixel. Par contre, nous avons diminué la capacité visuelle des entités normales qui étaient capables de voir jusqu'à deux pixels et qui sont devenus capables de ne voir que jusqu'à quatre pixels. Ainsi la capacité visuelle de l'entité avantagée est toujours égale à :

$$\begin{aligned} V_{Avantagée} &= 1/\tan(n_{Avantagée} \cdot \theta/2) \\ &= 1/\tan(1 \cdot 0.117/2) \\ &= 1/\tan(0.058) \\ &= 977.84 \end{aligned}$$

Alors que la capacité visuelle d'une entité normale devient égale à :

$$\begin{aligned} V_{Normale} &= 1/\tan(n_{Normale} \cdot \theta/2) \\ &= 1/\tan(4 \cdot 0.117/2) \\ &= 1/\tan(0.234) \\ &= 244.85 \end{aligned}$$

D'autre part, les entités normales ont gardé leurs tailles mais les entités grandes qui étaient 2 fois plus grandes que les entités normales sont désormais 10 fois plus grandes qu'elles. Les distances maximales de perception des entités avantagées sont donc les suivantes :

$$\begin{aligned} Dmax(Avantagée, Normale) &= R_{Normale} \cdot V_{Avantagée} \\ &= 1.02 \cdot 977.84 \\ &= 1000\ mètres \end{aligned}$$

et

$$\begin{aligned} Dmax(Avantagée, Grande) &= R_{Grande} \cdot V_{Avantagée} \\ &= 10.2 \cdot 977.84 \\ &= 10000\ mètres \end{aligned}$$

Et les distances maximales de perception des entités normales sont égales à :

$$\begin{aligned} Dmax(Normale, Normale) &= R_{Normale} \cdot V_{Normale} \\ &= 1.02 \cdot 244.85 \\ &= 250\ mètres \end{aligned}$$

et

$$\begin{aligned} Dmax(Normale, Grande) &= R_{Grande} \cdot V_{Normale} \\ &= 10.2 \cdot 244.85 \\ &= 2500\ mètres \end{aligned}$$

Le tableau 3.2 décrit les distances maximales de perception qu'ont les couples de types d'entités hétérogènes.

D_{max}	Normale	Grande	Grande & Avantagée	Avantagée
Normale	250	2500	2500	250
Grande	250	2500	2500	250
Grande & Avantagée	1000	10000	10000	1000
Avantagée	1000	10000	10000	1000

FIG. 3.2 – Les distances maximales de perception des entités hétérogènes

La détermination des tailles des zones d'effet et de conscience en se basant sur la distance maximale de perception (cf. 1.2.5) nous donne le système d'inéquations suivant :

$C(Normale) + E(Normale) \geq 250$

$C(Normale) + E(Grande) \geq 2500$

$C(Normale) + E(Grande\&Avantagée) \geq 2500$

$C(Normale) + E(Avantagée) \geq 250$

$C(Grande) + E(Normale) \geq 250$

$C(Grande) + E(Grande) \geq 2500$

$C(Grande) + E(Grande\&Avantagée) \geq 2500$

$C(Grande) + E(Avantagée) \geq 250$

$C(Grande\&Avantagée) + E(Normale) \geq 1000$

$C(Grande\&Avantagée) + E(Grande) \geq 10000$

$C(Grande\&Avantagée) + E(Grande\&Avantagée) \geq 10000$

$C(Grande\&Avantagée) + E(Avantagée) \geq 1000$

$C(Avantagée) + E(Normale) \geq 1000$

$C(Avantagée) + E(Grande) \geq 10000$

$C(Avantagée) + E(Grande\&Avantagée) \geq 10000$

$C(Avantagée) + E(Avantagée) \geq 1000$

En minimisant la somme des surfaces des zones d'effet et de conscience, L'application de la méthode de Newton nous a donné les zones de conscience et d'effet du tableau suivant :

	Zone de conscience	Zone d'effet
Normale	125	125
Grande	125	5000
Grande & Avantagée	5000	5000
Avantagée	5000	125

3.2.2 Résultats

Pour évaluer la gestion d'effet, nous avons comparé le fonctionnement des simulations sans filtrage et avec la gestion d'effet. On a fait varier le nombre de clients dans les deux cas. On a collecté à la fin de chaque expérience le nombre de messages échangés, la charge du serveur et la latence des messages. On a répété chaque expérience quatre fois et calculé la moyenne des valeurs résultantes.

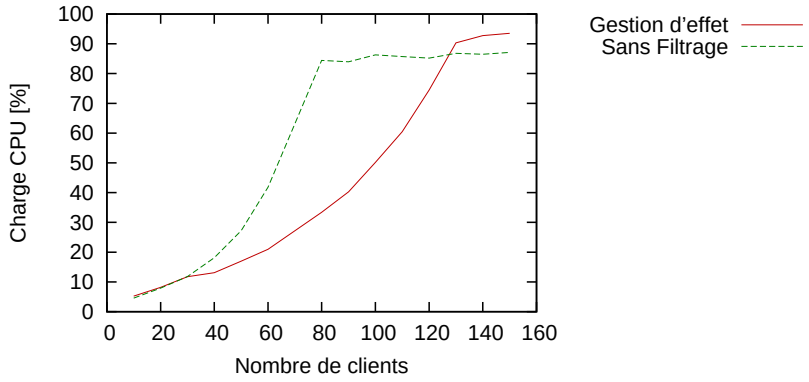

FIG. 3.3 – Charge du serveur

3.2.2.1 Expériences avec les entités homogènes

Sans filtrage, le serveur atteint sa charge maximale et devient un goulot d'étranglement à partir de 80 clients (figure 3.3). Par contre avec la gestion d'effet, le serveur arrive à gérer jusqu'à 130 clients, ce qui permet d'en doubler approximativement le nombre (passage à l'échelle). Le goulot d'étranglement provient de la charge de calcul du serveur et non du réseau parce que le réseau liant les différentes machines possède une bande passante de 3 Gbits/s. Si la bande passante était limitée, le goulot d'étranglement serait arrivé plus vite sans filtrage et il aurait été dû à la surcharge de la bande passante et non du serveur.

L'atteinte rapide du goulot d'étranglement sans l'utilisation du filtrage est due au coût élevée de la transmission de messages par le serveur. Dans la figure 3.4, on voit que le nombre de messages envoyés par un serveur sans filtrage est largement supérieur au nombre de messages envoyés en utilisant la gestion d'effet. Ceci permet à la gestion d'effet de repousser le goulot d'étranglement qui ne dépend plus du coût de l'envoi des messages uniquement mais aussi du coût du filtrage.

L'atteinte de la charge maximale chez le serveur se traduit systématiquement par une perte des messages reçus par lui (figures 3.5, 3.6). En fait, le nombre de messages envoyés par un client est toujours constant quel que soit le nombre total de clients de l'application et pour cette raison le nombre de messages reçus par le serveur est toujours constant quelles que soient les méthodes et les architectures utilisées.

On a aussi mesuré la latence de la réception des messages de mise à jour. L'atteinte de la charge maximale du serveur affecte cette latence qui augmente rapidement dès que la charge

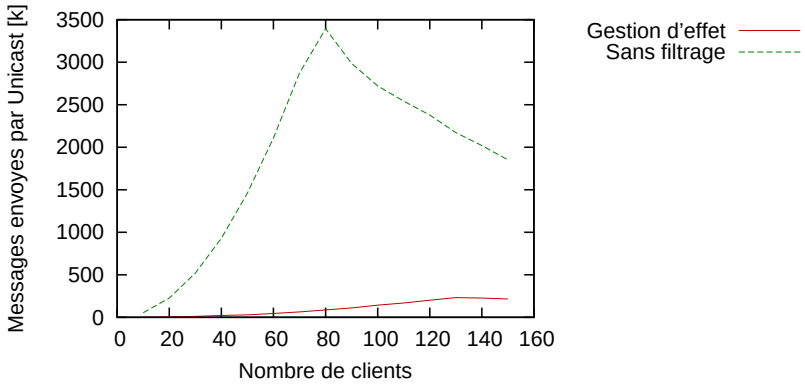

FIG. 3.4 – Les messages envoyés par le serveur

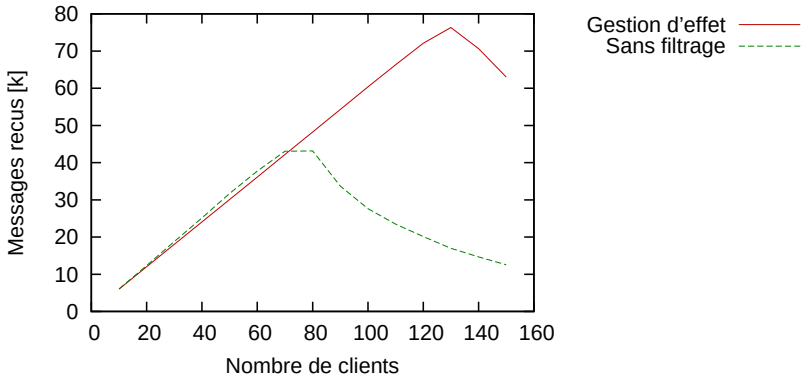

FIG. 3.5 – Les messages reçus par le serveur

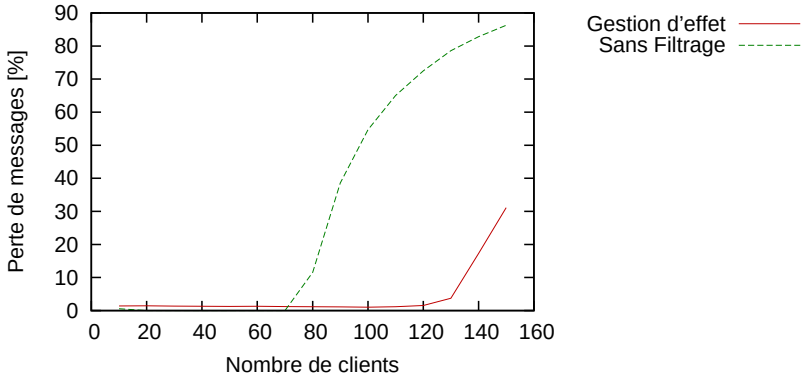

FIG. 3.6 – Le taux de perte de messages chez le serveur

maximale est atteinte (figure 3.7). Cette augmentation de la latence est normale puisque quand le serveur devient surchargé, il fait attendre les messages reçus dans sa file d'attente jusqu'à ce qu'il ait le temps de les traiter. Pendant ce temps là, la latence augmente progressivement et l'application devient de moins en moins interactive.

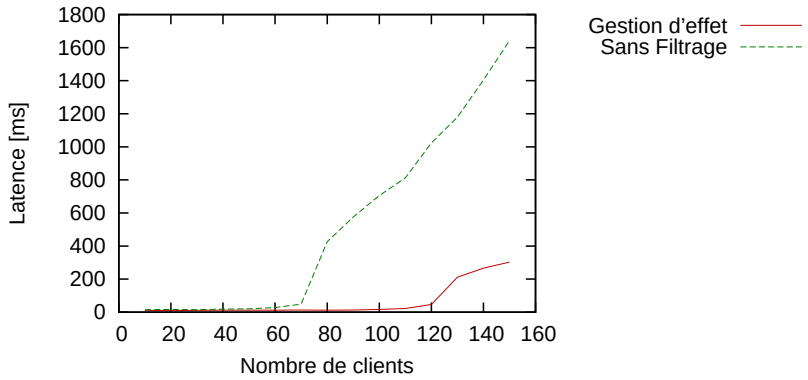

FIG. 3.7 – La latence des messages de mise à jour

D'autre part, dans la figure 3.8 on voit que le taux élevé d'envoi de messages chez le serveur sans l'utilisation de filtrage nécessite une bande passante sortante assez importante (de l'ordre de 150 Mbits/s pour 60 clients par exemple). Ceci est assez coûteux pour les applications commerciales hébergeant des serveurs. En utilisant la gestion d'effet, la bande passante sortante nécessaire est beaucoup moins élevée (de l'ordre de 3 Mbits/s pour 60 clients).

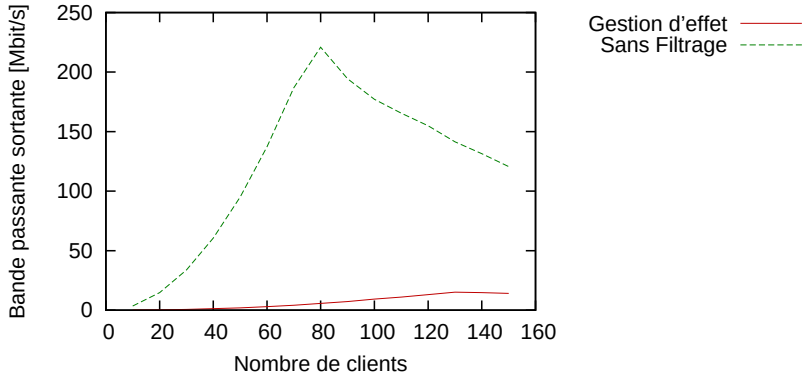

FIG. 3.8 – La bande passante sortante du serveur

3.2.2.2 Expériences avec les entités hétérogènes

Les entités hétérogènes sont plus conscientes les unes des autres que les entités homogènes puisqu'elles ont des tailles et des capacités visuelles plus élevées. Cette augmentation de la conscience entraîne une augmentation de la nécessité d'échange de messages entre les entités. Ceci implique une augmentation du taux d'envoi de messages chez le serveur avec la gestion d'effet (figure 3.9).

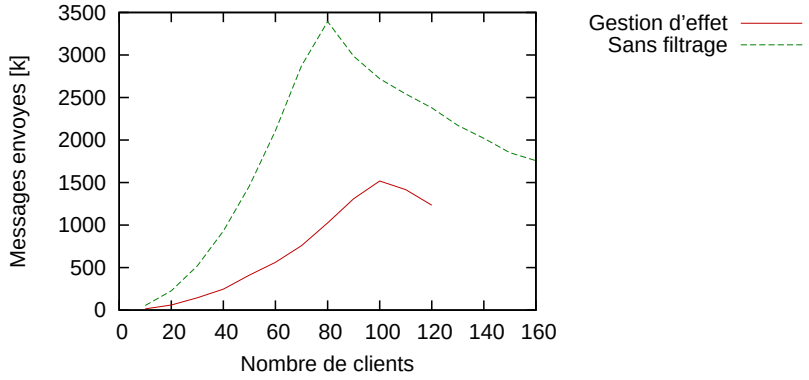

FIG. 3.9 – Les messages envoyés par le serveur

Il faut noter que sans filtrage, le serveur se comporte de la même manière avec les entités homogènes et hétérogènes parce qu'il livre à chaque entité les messages de mise à jour de toutes les autres sans prendre en considération sa capacité visuelle ou les tailles des autres.

L'augmentation du coût d'envoi de messages avec les entités hétérogènes en utilisant la gestion d'effet fait saturer la charge du serveur plus vite (figure 3.10). Ainsi le passage à l'échelle est plus limité, on atteint 100 clients au lieu de 130 avec les entités homogènes.

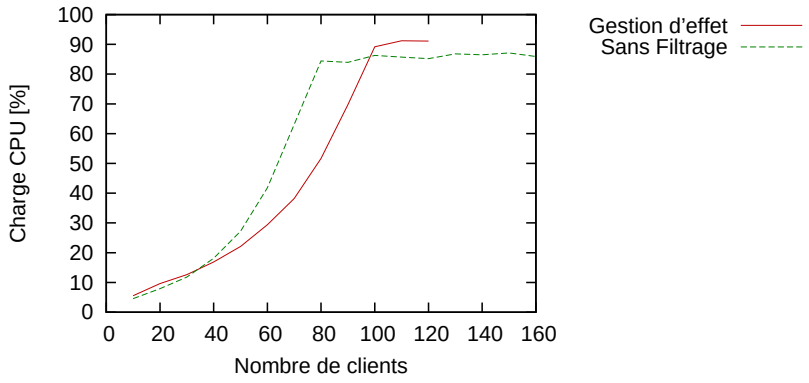

FIG. 3.10 – La charge chez le serveur

La perte des messages arrive par la suite plus rapidement (figures 3.11, 3.12) et la latence augmente également plutôt (figure 3.13).

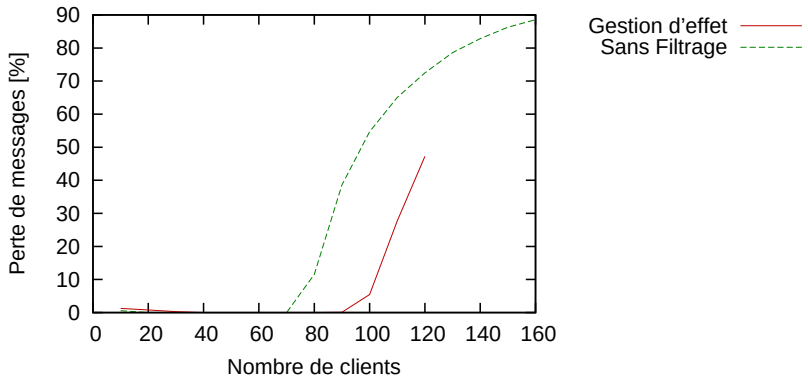

FIG. 3.11 – Le taux de perte chez le serveur

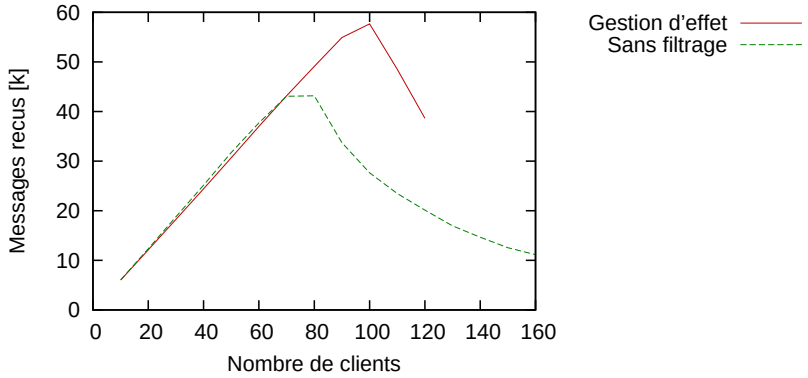

FIG. 3.12 – Les messages reçus par le serveur

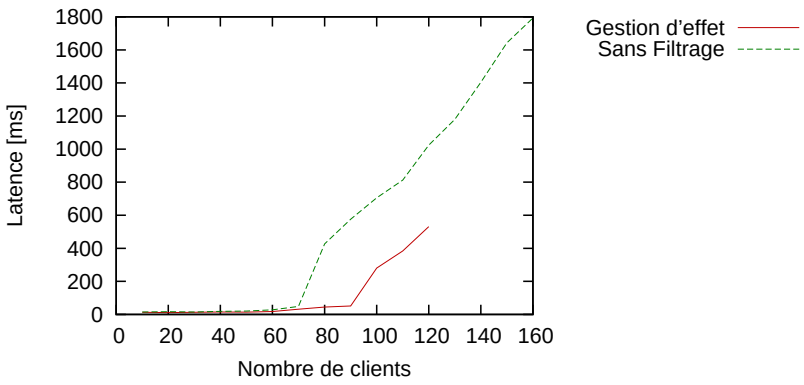

FIG. 3.13 – La latence des messages de mise à jour

Malgré cette diminution du passage à l'échelle avec les entités fortement conscientes les unes des autres, la gestion d'effet permet un meilleur passage à l'échelle par rapport aux simulations qui n'utilisent pas de filtrage.

Par contre, l'économie de la bande passante sortante utilisée avec la gestion d'effet reste assez importante (figure 3.14).

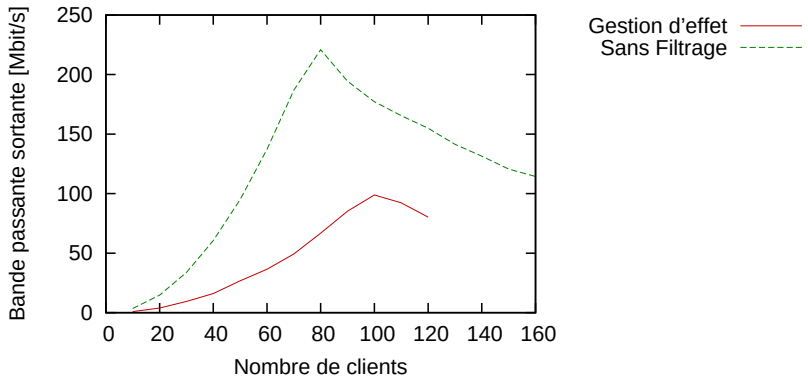

FIG. 3.14 – La bande passante sortante du serveur

3.3 Comparaison entre la gestion d'effet et les autres méthodes de gestion d'intérêt

Pour compléter notre évaluation de la gestion d'effet, on a comparé son fonctionnement à celui des autres méthodes de gestion d'intérêt utilisant des zones comme le modèle spatial d'interaction et la gestion par zone d'intérêt. Nous avons comparé avec le modèle spatial d'interaction dans sa version simple qui n'utilise pas des foci et des nimbi parce que le coût du filtrage avec deux étapes au lieu d'une seule sera très élevé. C'est pour cette raison que les systèmes qui ont utilisé le modèle l'ont utilisé dans sa version simple. Pour mener une comparaison équitable entre les méthodes, on a utilisé des paramètres de filtrage compatibles pour les trois méthodes. Ainsi ces trois méthodes de gestion d'intérêt fournissent un niveau constant de réalisme.

Le modèle spatial d'interaction et la gestion par zone d'intérêt définissent les tailles de leurs zones d'une manière empirique. Pour pouvoir effectuer notre évaluation d'une manière équitable, on a établi les mécanismes qui permettent la définition des tailles des zones de la même manière que celle avec laquelle on a défini nos zones de conscience et d'effet. Ces mécanismes se basent

sur les règles de perception qui permettent à l'utilisateur de voir une entité lorsque sa taille sur son écran est supérieure ou égale à un pixel.

De cette manière, la comparaison entre les fonctionnements des trois méthodes est faite avec un niveau constant de réalisme grâce aux règles établies par la perception.

3.3.1 Définition des tailles des zones

3.3.1.1 La gestion par zone d'intérêt

La gestion par zone d'intérêt permet à une entité de percevoir les entités qui sont à l'intérieur de sa zone d'intérêt. La zone d'intérêt d'une entité doit donc couvrir toutes les entités qu'elle est capable de voir avec les règles prédéfinies de perception. Ainsi elle doit couvrir toutes les entités situées à une distance inférieure ou égale à sa distance maximale de perception (figure 3.15). Par conséquent, le rayon de la zone d'intérêt doit être supérieur ou égal à toutes les distances maximales de perception de l'entité. On attribue donc à l'entité une zone d'intérêt de rayon égal à la plus grande distance maximale de perception qu'elle a d'une autre entité.

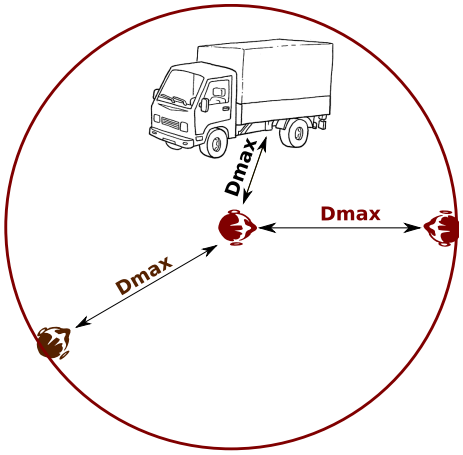

FIG. 3.15 – Détermination du rayon de la zone d'intérêt

D'où la règle générale :

$$AOI(A) = max\{D_{max}(A, n), \forall n \in S\}$$

3.3.1.2 Le modèle spatial d'interaction

Nous comparons également le fonctionnement de la gestion d'effet avec celui du modèle spatial d'interaction pris dans sa forme la plus simple et la plus utilisé (c'est-à-dire en utilisant les auras sans les foci et les nimbi). Pour définir des tailles d'auras qui satisfassent les règles de perception, notre règle est la suivante :

R' : "Quand A voit B dans le filtrage basé sur la perception, A doit voir B dans le modèle spatial d'interaction"

A voit B dans le filtrage basé sur la perception si la distance entre A et B est plus petite que la distance maximale de perception correspondante :

$$\mathrm{D(A,B)} \leq D_{max}\mathrm{(A,B)}$$

Et A voit B dans le modèle spatial d'interaction si les auras de A et B chevauchent :

D(A,B) $\leq$ Aura(A)+Aura(B) tel que Aura(N) est le rayon de l'aura de l'entité N.

Une condition qui rend **R'** vraie est :

$$\textbf{C'} : \mathrm{Aura(A)+Aura(B)} \geq D_{max}\mathrm{(A,B)}$$

parce que :

$$\mathrm{D(A,B)} \leq D_{max}\mathrm{(A,B)} \text{ et } D_{max}\mathrm{(A,B)} \leq \mathrm{Aura(A)+Aura(B)} \Rightarrow \mathrm{D(A,B)} \geq \mathrm{Aura(A)+Aura(B)}$$

Donc la condition **C'** permet à une entité qui voit l'autre dans le filtrage basé sur la perception de la voir avec le modèle spatial d'interaction. Ceci fait de C' la base de la traduction des règles de perception dans le modèle spatial d'interaction.

D'autre part, les auras sont utilisées pour exprimer la perception et la perceptibilité en même temps. Alors le calcul de la perception de B de A nous donne une équation qui contient également Aura(A) et Aura(B) :

$$\mathrm{Aura(B)+Aura(A)} \geq D_{max}\mathrm{(B,A)}$$

Donc le rayon de l'aura d'une entité dépend aussi des rayons des auras des autres entités. Ainsi les deux inéquations suivantes :

$$\text{Aura(A)+Aura(B)} \geq D_{max}\text{(A,B)}$$

et

$$\text{Aura(B)+Aura(A)} \geq D_{max}\text{(B,A)}$$

ont comme solution :

$$\text{Aura(B)+Aura(A)} \geq max(D_{max}(A,B), D_{max}(B,A))$$

Reprenons l'exemple de l'application qui comprend trois types d'entités A, B et C. Les conditions de la traduction de la perception sont :

$Aura(A) + Aura(B) \geq max(D_{max}(A,B), D_{max}(B,A))$ (Pour que A voit B et B voit A)

$Aura(A) + Aura(C) \geq max(D_{max}(A,C), D_{max}(C,A))$

$Aura(B) + Aura(C) \geq max(D_{max}(B,C), D_{max}(C,B))$

$2 * Aura(A) \geq D_{max}(A,A)$ (pour que deux entités de type A puissent se voir quand il faut)

$2 * Aura(B) \geq D_{max}(B,B)$

$2 * Aura(C) \geq D_{max}(C,C)$

Donc, S étant l'ensemble des types d'entités de l'application, le règle générale est :

$$\forall a \in S, \forall b \in S, Aura(a) + Aura(b) \geq max(D_{max}(a,b), D_{max}(b,a))$$

Ceci définit un système d'inéquations linéaires générant un ensemble de solutions. La solution optimale correspond au nombre minimal d'interactions, par suite au nombre minimal de collisions entre les auras. Et la minimisation du nombre de collisions exige la minimisation des aires des auras :

$$AA = \pi \cdot \sum_{n \in S} Aura(n)^2$$

Donc la solution minimale sera obtenue en résolvant le système d'inéquations en essayant de minimiser AA.

3.3.2 Paramètres du filtrage

Les paramètres de nos expériences sont identiques à celles utilisés avec la gestion d'effet : on utilise une architecture client/serveur, le média visuel et des entités homogènes ou hétérogènes. On a fait varier le nombre de clients dans chacune des méthodes. Chaque expérience est répétée quatre fois pour calculer les moyennes des valeurs résultantes.

3.3.2.1 Entités homogènes

Pour calculer les tailles des zones pour la gestion par zone d'intérêt et le modèle spatial d'interaction, on reprend les distances maximales de perception utilisées pour les entités homogènes avec la gestion d'effet (tableau 3.16).

D_{max}	Normale	Grande	Grande & Avantagée	Avantagée
Normale	500	1000	1000	500
Grande	500	1000	1000	500
Grande & Avantagée	1000	2000	2000	1000
Avantagée	1000	2000	2000	1000

FIG. 3.16 – Les distances maximales de perception des entités homogènes

La gestion par zone d'intérêt

On se base ainsi sur la règle suivante de définition des tailles de zones d'intérêt (cf. 3.3.1.1) :

$$AOI(A) = max\{D_{max}(A, n), \forall n \in S\}$$

Nos entités auront donc des zones d'intérêt ayant les tailles suivantes :

$$\begin{aligned} AOI(Normale) &= max\{D_{max}(A, n), \forall n \in S\} \\ &= max\{D_{max}(Normale, Normale), D_{max}(Normale, Grande), \\ &\quad D_{max}(Normale, Grande\&Avantagée), D_{max}(Normale, Avantagée)\} \\ &= max\{500, 1000, 1000, 500\} \\ &= 1000\ mètres \end{aligned}$$

Avec une procédure équivalente, on obtient :

$AOI(Grande) = 1000\ mètres$

$AOI(Grande\&Avantagée) = 2000\ mètres$

$AOI(Avantagée) = 2000\ mètres$

Le modèle spatial d'interaction

La détermination des tailles des auras en se basant sur les mêmes règles de perception (cf. 3.3.1.2) suit la règle générale suivante :

$$\text{Aura(B)+Aura(A)} \geq max(D_{max}(A,B), D_{max}(B,A))$$

Ceci nous donne le système d'inéquations suivant :

$Aura(Normale) + Aura(Grande) \geq max(500, 1000)$ (Pour qu'une entité normale voit une entité grande quand il le faut)

$Aura(Normale) + Aura(Grande\&Avantagée) \geq max(1000, 1000)$

$Aura(Normale) + Aura(Avantagée) \geq max(500, 1000)$

$Aura(Grande) + Aura(Grande\&Avantagée) \geq max(2000, 1000)$

$Aura(Grande) + Aura(Avantagée) \geq max(500, 2000)$

$Aura(Avantagée) + Aura(Grande\&Avantagée) \geq max(2000, 1000)$

$2 * Aura(Normale) \geq 500$ (pour que deux entités de type normales puissent se voir quand il faut)

$2 * Aura(Grande) \geq 1000$

$2 * Aura(Grande\&Avantagée) \geq 2000$

$2 * Aura(Avantagée) \geq 1000$

Ce système d'inéquation est résolu en minimisant la somme des surfaces des auras :

$$\begin{aligned} AA &= \pi \cdot \sum_{n \in S} Aura(n)^2 \\ &= \pi \cdot (Aura(Normale)^2 + Aura(Grande)^2 + Aura(Grande\&Avantagée)^2 \\ &\quad + Aura(Avantagée)^2) \end{aligned}$$

La solution optimale obtenue est la suivante :

$Aura(Normale) = 250\ mètres$

$Aura(Grande) = 1000\ mètres$

$Aura(Grande\&Avantagée) = 1000\ mètres$

$Aura(Avantagée) = 1000\ mètres$

3.3.2.2 Entités hétérogènes

Le but de l'utilisation des entités hétérogènes en comparant la gestion d'effet aux autres méthodes est de voir comment chacune de ces méthodes réagit vis-à-vis de l'augmentation de la conscience des entités au sein de l'environnement et de l'augmentation du besoin de relations asymétriques.

On utilise les distances maximales des entités hétérogènes utilisées pour la gestion d'effet (tableau 3.17) pour calculer les tailles des zones dans les deux méthodes.

D_{max}	Normale	Grande	Grande & Avantagée	Avantagée
Normale	250	2500	2500	250
Grande	250	2500	2500	250
Grande & Avantagée	1000	10000	10000	1000
Avantagée	1000	10000	10000	1000

FIG. 3.17 – Les distances maximales de perception des entités hétérogènes

La gestion par zone d'intérêt

En utilisant la même procédure que celle des entités homogènes, on obtient les valeurs suivantes :

$AOI(Normale) = 2500\ mètres$

$AOI(Grande) = 2500\ mètres$

$AOI(Grande\&Avantagée) = 10000\ mètres$

$AOI(Avantagée) = 10000\ mètres$

Le modèle spatial d'interaction

En résolvant le système d'inéquations associé aux entités hétérogènes, on obtient la solution suivante :

$Aura(Normale) = 125\ mètres$

$Aura(Grande) = 5000\ mètres$

$Aura(Grande\&Avantagée) = 5000\ mètres$

$Aura(Avantagée) = 5000\ mètres$

3.3.3 Résultats

3.3.3.1 Entités homogènes

La figure 3.18 montre le taux d'envoi de messages par le serveur sans filtrage et avec les méthodes de gestion d'intérêt. On constate que les trois méthodes de gestion d'intérêt ont réussi à réduire significativement le taux de messages envoyés par le serveur. Ces méthodes repoussent le goulot d'étranglement qui est dû avec ces méthodes au coût du filtrage et non au coût de l'envoi de messages comme avec l'application qui n'utilise pas de filtrage.

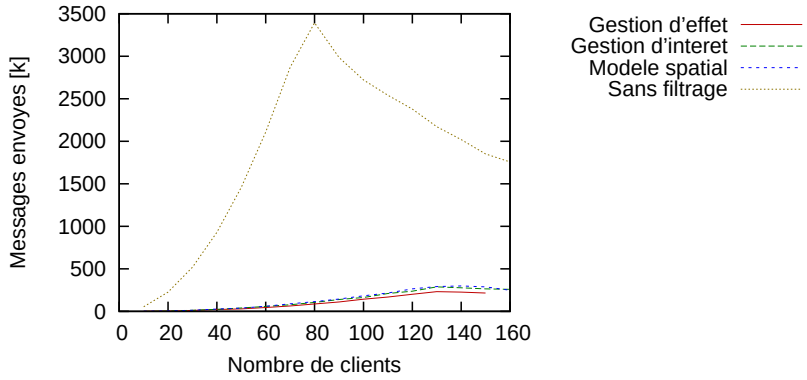

FIG. 3.18 – Les messages envoyés par le serveur

Cette diminution du taux d'envoi de messages permet à la charge du serveur d'être acceptable jusqu'à 130 clients au lieu de 80 sans filtrage (figure 3.19).

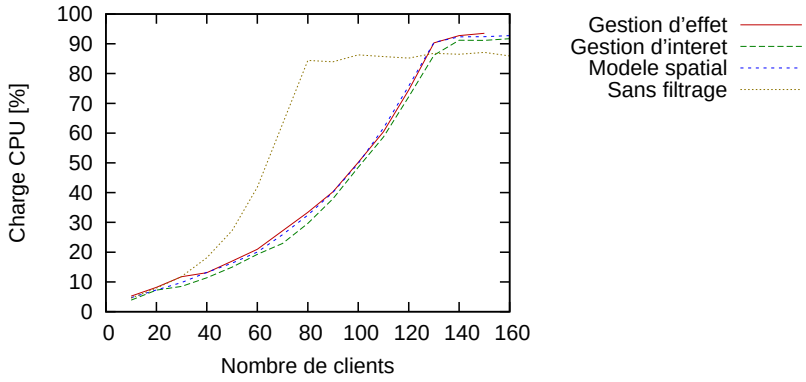

FIG. 3.19 – La charge chez le serveur

L'atteinte de la charge maximale du serveur se manifeste en une perte ascendante de messages (figures 3.20, 3.21).

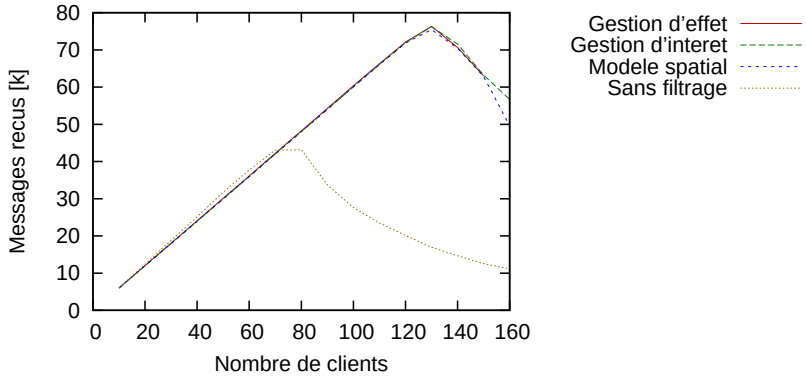

FIG. 3.20 – Les messages reçus par le serveur

L'atteinte de la charge maximale du serveur implique aussi une augmentation de la latence (figure 3.22).

Étudions maintenant la différence de fonctionnement entre les trois méthodes. Ces trois méthodes donnent des résultats identiques vis-à-vis de la charge du serveur parce que le goulot d'étranglement est dû au coût du filtrage qui est équivalent dans les trois méthodes. Par contre, le taux de messages envoyés par le serveur est différent dans les trois méthodes (figure 3.23). La gestion d'effet fournit un filtrage plus important de données que la gestion par zone d'intérêt et le modèle spatial d'interaction. Ainsi le serveur de la gestion d'effet a besoin d'une bande passante sortante moins élevée que les deux autres méthodes (figure 3.24)

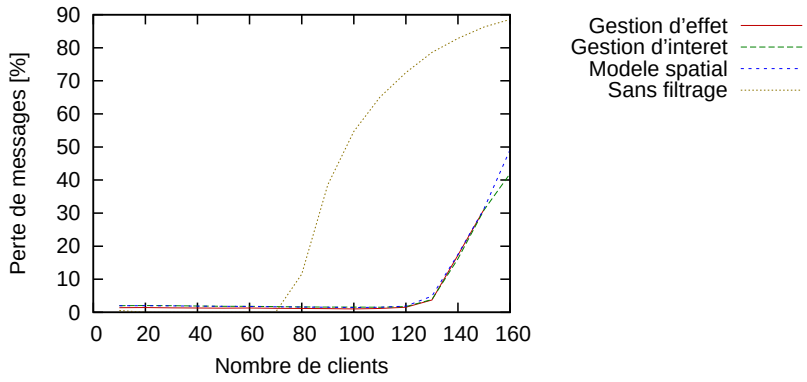

FIG. 3.21 – Le taux de perte de messages chez le serveur

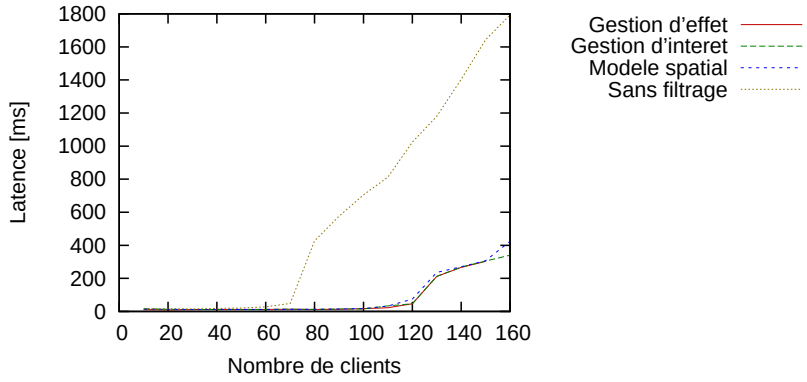

FIG. 3.22 – La latence des messages de mise à jour

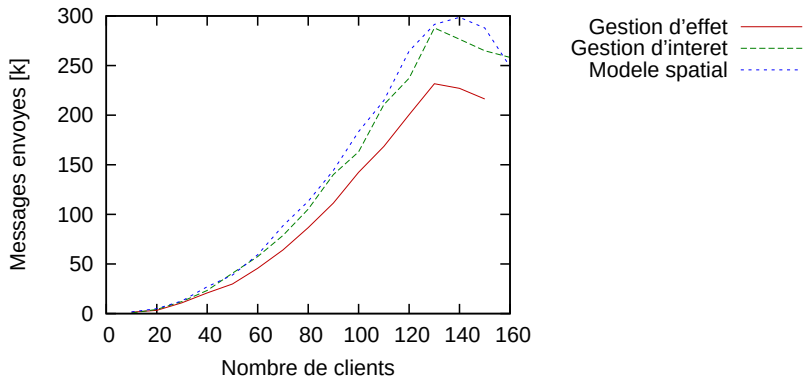

FIG. 3.23 – Les messages envoyés par les serveur avec les méthodes de filtrage

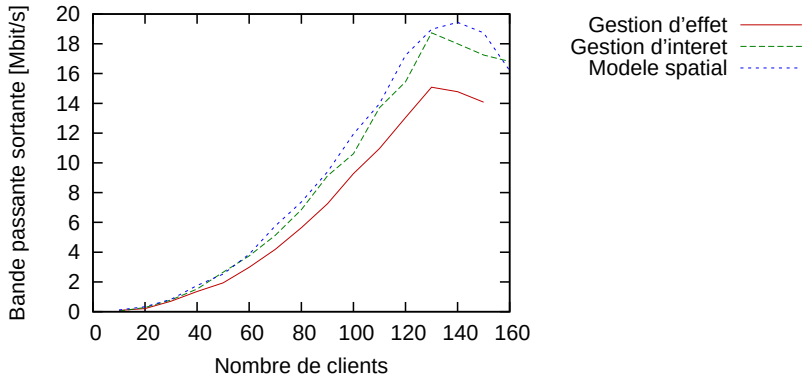

FIG. 3.24 – La bande passante sortante du serveur avec les méthodes de filtrage

Voyons dans ce qui suit les raisons pour lesquelles la gestion d'effet permet un meilleur filtrage que les deux autres méthodes de gestion d'intérêt.

Comparaison entre la gestion d'effet et la gestion par zone d'intérêt

La gestion d'effet permet l'expression individuelle de la manifestation d'une entité à travers sa zone d'effet alors que la gestion par zone d'intérêt ne permet pas cette expression. Pour cette raison, la taille de la zone d'intérêt d'une entité dépend de toutes les entités de la scène. Une entité possède à la fin une zone d'intérêt qui lui permet de voir toutes les entités qui l'intéresse mais aussi de petites entités qui ne l'intéressent pas mais sa zone d'intérêt est trop grande pour pouvoir faire la distinction.

Dans la figure 3.25, prenons l'exemple de deux entités parmi nos entités homogènes : une entité normale (représentée par une personne), une entité avantagée (représentée par une personne portant des jumelles) et une entité grande (représentée par un camion). Nous allons étudier les messages reçus par la personne ayant une vue normale.

En se basant sur notre exemple des entités homogènes, on sait que :

$D_{max}(Personne, Jumelles) = 500$ mètres

et

$D_{max}(Personne, Camion) = 1000$ mètres

La personne observatrice de la figure 3.25 est a une distance de 1000 mètres de l'autre personne et du camion. Elle est capable de voir le camion parce qu'elle est à sa distance maximale

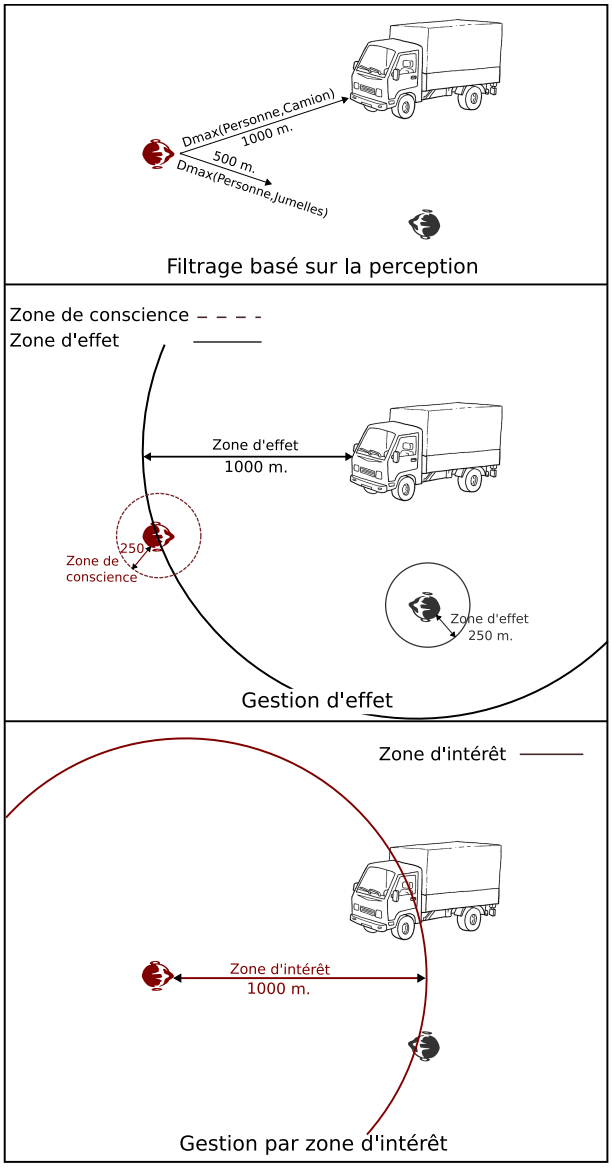

FIG. 3.25 – Comparaison entre gestion d'effet et gestion par zone d'intérêt

de perception du camion mais normalement elle n'est pas intéressée par l'autre personne parce qu'elle est à une distance supérieure de sa distance maximale de perception d'elle. Alors normalement, elle reçoit les messages de mise à jour du camion mais pas celles de l'autre personne.

Prenons le même scénario mais avec la zone d'effet. On a vu que les zones de conscience et d'effet résultantes des distances maximales de perception sont :

Zone de conscience (Personne) = 250 mètres

Zone d'effet (Jumelles) = 250 mètres

Zone d'effet (Camion) = 1000 mètres

Ainsi, la personne recevra les messages de mise à jour du camion à cause du chevauchement de sa zone de conscience avec la zone d'effet du camion et elle ne recevra pas les messages de mise à jour de la personne avantagée qu'elle n'est pas capable de voir.

Par contre avec la gestion par zone d'intérêt, une personne possède une zone d'intérêt de 1000 mètres. Sa zone d'intérêt inclura dans notre exemple le camion et la deuxième personne, alors elle recevra les mises à jour des deux autres entités alors qu'elle n'est intéressée que par le camion.

Cet exemple illustre l'inconvénient de la non expression de manifestation dans la gestion par zone d'intérêt et explique l'avantage de la gestion d'effet par rapport à la gestion par zone d'effet. Pour cette raison, on voit dans notre courbe que le serveur envoie plus de messages avec la gestion par zone d'intérêt, ce qui produit plus de coût d'envoi, consomme de la bande passante sortante et consomme plus de bande passante entrante et de charge chez les clients qui reçoivent des messages qui ne les intéressent pas.

Comparaison de fonctionnement avec le modèle spatial d'interaction

La gestion d'effet permet l'établissement de relations asymétriques alors que le modèle spatial d'interaction ne permet pas l'établissement de ce genre de relations s'il est utilisé sans les foci et les nimbi. Ceci impose aux entités la réception de messages qui ne les intéressent pas.

La figure 3.26 reprend le même scénario déjà étudié mais avec le modèle spatial d'interaction.

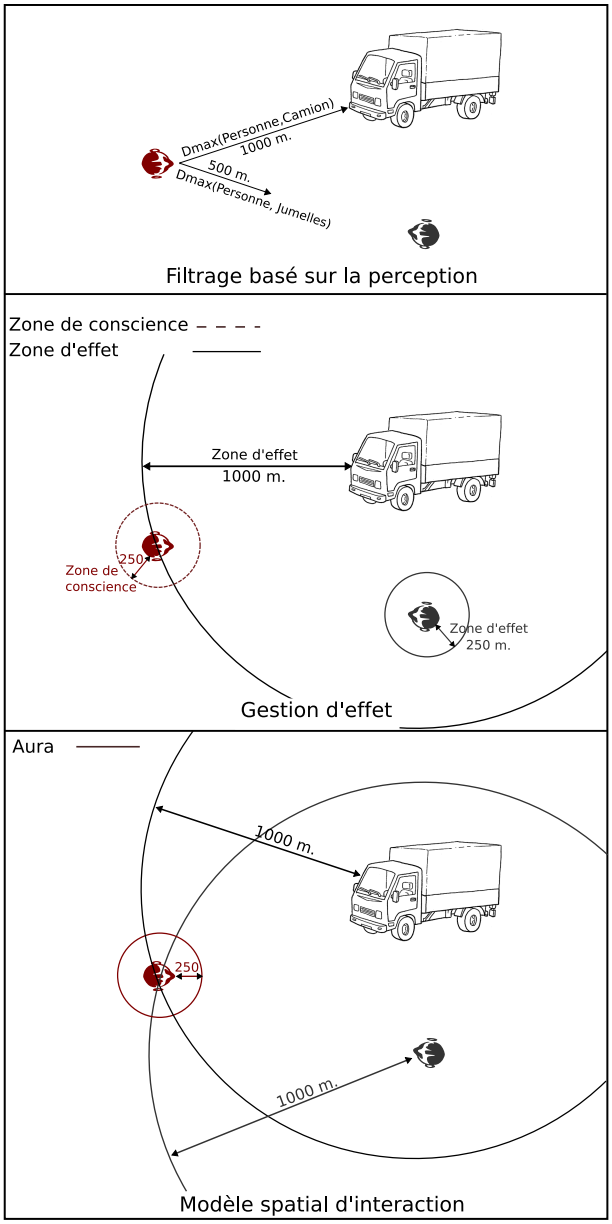

FIG. 3.26 – Comparaison entre gestion d'effet et modèle spatial d'interaction

Le modèle spatial d'interaction définit pour une entité une seule zone "l'aura" qui exprime en même temps ses intérêts et sa manifestation. Quand les auras de deux entités chevauchent, une relation symétrique bilatérale est établie entre ces deux entités et elles seront donc conscientes l'une de l'autre. Une entité importante recevra ainsi les messages de toutes les entités qui sont intéressées par elle à cause de l'établissement exclusif de relations symétriques.

Dans notre exemple, la personne ayant une vue normale possèdent une aura de rayon égal à 250 mètres, la personne ayant une vue avantagée possède une aura de 1000 mètres et le camion possède également une aura de 250 mètres. On voit que l'aura de la personne observatrice chevauche les auras de l'autre personne et du camion, elle recevra donc les messages de mises à jour des deux entités. Ainsi, dans des cas pareils le modèle spatial d'interaction laisse passer des messages inutiles aux clients à cause de l'impossibilité d'établissement des relations asymétriques. Cet exemple illustre donc l'effet des relations symétriques du modèle spatial d'interaction et explique la cause de la réception de messages inutiles par les clients. Ce qui impose une charge supplémentaire sur le serveur, les clients et le réseau.

3.3.3.2 Entités hétérogènes

Avec les entités hétérogènes, les entités sont plus conscientes les unes des autres parce que les entités avantagées ont des capacités de perception beaucoup plus élevées que celles des entités homogènes et les grandes entités sont beaucoup plus importantes que les grandes entités des entités homogènes. Ainsi les entités sont capables de voir des entités qui sont à l'extrémité de l'environnement. Le taux de messages échangés augmente parce que les besoins des entités augmentent. D'autre part, les écarts entre les capacités de perception et de perceptibilité augmentent et donc le besoin de l'établissement de relations asymétriques augmente également.

Par contre, toutes les méthodes de gestion d'intérêt ne réagissent pas d'une manière identique. La gestion par zone d'intérêt est handicapée par l'impossibilité de l'expression de manifestation et le modèle spatial d'interaction est handicapé par l'établissement des relations symétriques. Ces raisons font que la gestion d'effet gère mieux l'augmentation de la conscience et l'augmentation de l'écart entre les capacités des entités. Dans la figure 3.27, on voit que le taux de messages envoyés par la gestion par zone d'intérêt et par le modèle spatial d'interaction se rapproche du taux de messages envoyés sans filtrage. Ce qui fait que le goulot d'étranglement ne dépend plus

que du coût du filtrage mais aussi du coût d'envoi de messages. En même temps, la gestion d'effet envoie moins de messages que les deux autres méthodes, elle atteint donc le goulot d'étranglement plus tard qu'elles.

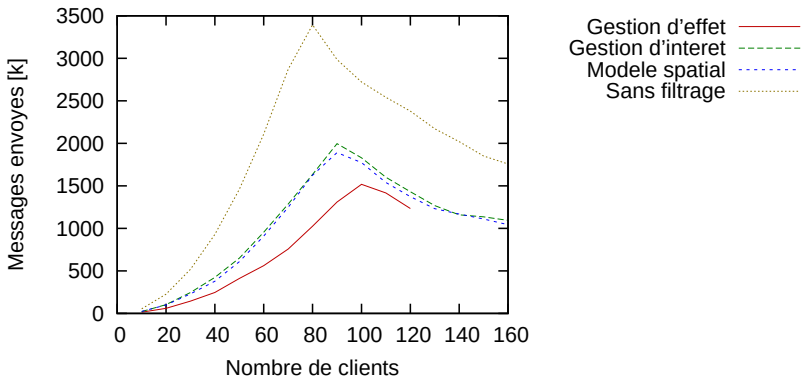

FIG. 3.27 – Les messages envoyés par le serveur

Les serveurs de ces deux méthodes deviennent un goulot d'étranglement avec 90 clients alors que le serveur de la gestion d'effet réussit à atteindre les 100 clients (figure 3.28) grâce à son économie d'envoi de messages.

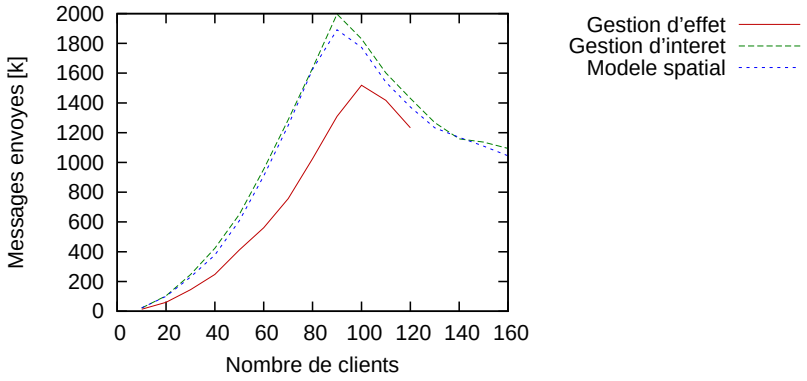

FIG. 3.28 – Les messages envoyés par le serveur

Ce repoussement du goulot d'étranglement repousse également la perte de messages (figures 3.29, 3.30)

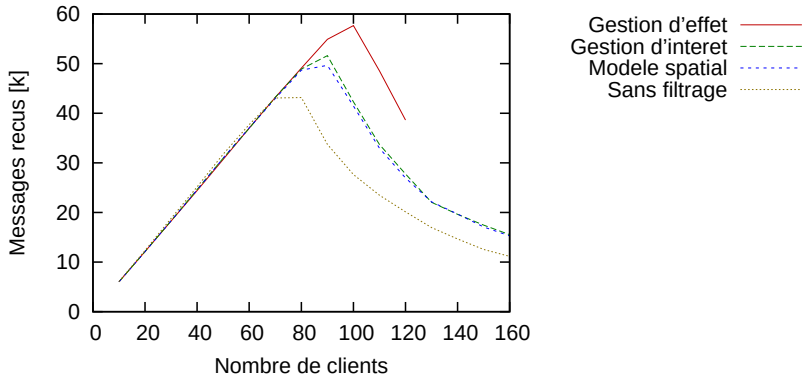

FIG. 3.29 – Les messages reçus par le serveur

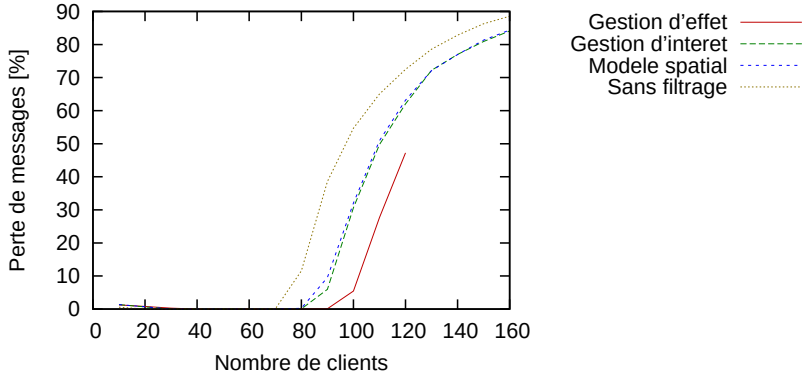

FIG. 3.30 – Le taux de perte de messages chez le serveur

L'augmentation de la latence est également repoussée (figure 3.31) avec la gestion d'effet.

Le gain de la consommation des ressources réseau est aussi considérable. Un serveur d'une application commerciale aura besoin de moins de bande passante sortante avec la gestion d'effet qu'avec les autres méthodes de gestion d'intérêt (figure 3.32).

Finalement, l'utilisation de la gestion d'effet dans des environnements hébergeant des entités ayant des intérêts et des manifestations hétérogènes permet un meilleur passage à l'échelle et la consommation d'une plus petite bande passante sortante par rapport aux autres méthodes de gestion d'intérêt.

3.3.4 Conclusion

La gestion d'effet est donc capable de réaliser un filtrage plus exact et donc plus performant que les autres méthodes de gestion d'intérêt. Ceci est vrai avec des environnements hébergeant

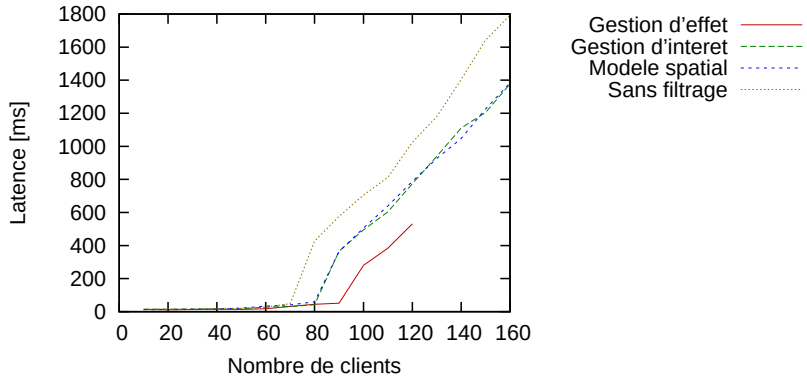

FIG. 3.31 – La latence des messages de mise à jour

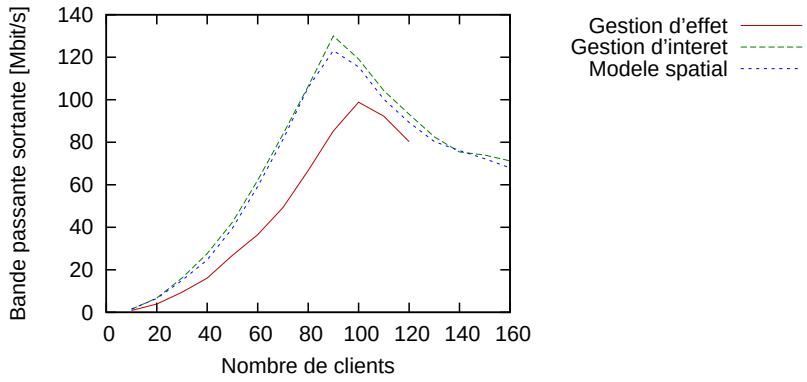

FIG. 3.32 – La bande passante sortante du serveur

des entités ayant des capacités homogènes et hétérogènes. Le problème du passage à l'échelle devient lié à la saturation de la charge du serveur, pour cette raison nous avons remplacé l'architecture client/serveur par une architecture client/multi-serveurs. Nous avons évalué cette architecture dans la section suivante.

3.4 Évaluation de l'architecture client/multi-serveurs

L'utilisation d'une méthode de filtrage ne permet que de repousser le problème de goulot d'étranglement sans le résoudre. Malgré le gain de ressources présenté par la gestion d'effet, le goulot d'étranglement finit par arriver. Pour pouvoir passer à l'échelle d'une manière continue, on a remplacé l'architecture client/serveur par une architecture client/multi-serveurs. Le rôle de cette architecture est de rajouter un serveur dès que le goulot d'étranglement est atteint. Le défi est que cette solution reste efficace avec l'augmentation du nombre serveurs au sein

d'un environnement ouvert. Ceci est difficile parce que quand le nombre de clients (et par suite de serveurs) devient élevé dans un environnement ouvert, le taux de communication entre les serveurs devient plus important à cause de l'augmentation du nombre de clients intéressés par des entités situées dans d'autres régions. Nous présentons dans la suite nos solutions à ce problème.

3.4.1 Paramètres du filtrage

On a utilisé le même environnement que pour les expériences précédentes. Cet environnement est divisé en régions égales au nombre de serveurs. Les simulations utilisent le média visuel et les entités homogènes. Les serveurs communiquent avec leurs clients et entre eux en unicast. Les entités sont également réparties dans l'espace dû à leurs comportements.

On a fait varier le nombre de clients dans quatre séries d'expériences qui utilisent 1, 2, 3 ou 4 serveurs. On a répété chaque expérience quatre fois et calculé la moyenne des résultats.

3.4.2 Résultats

La figure 3.33 montre le nombre de messages envoyés par un seul serveur et le point d'atteinte du goulot d'étranglement en variant le nombre de serveurs. On observe que l'ajout d'un serveur réussit à établir un passage à l'échelle continu qui se manifeste par le rajout d'un nombre constant de clients quelque soit le nombre de serveurs. On déduit que notre algorithme réussit à gérer l'augmentation d'intérêt entre les serveurs qui suit l'augmentation de leurs nombres de clients.

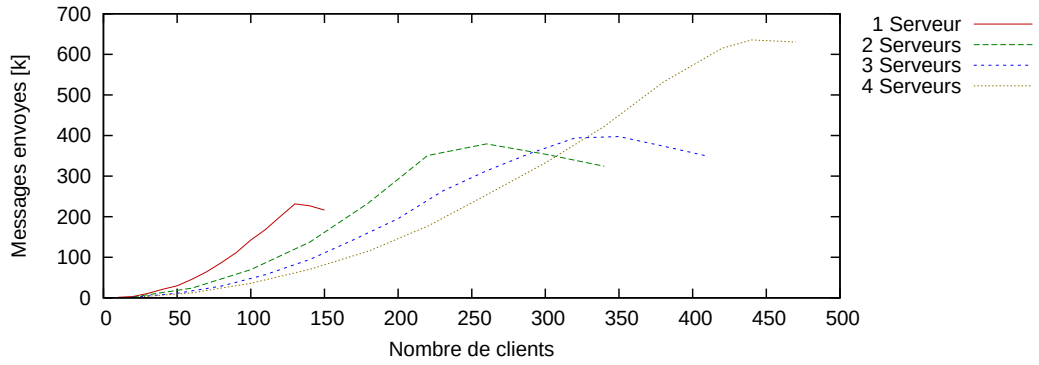

FIG. 3.33 – Les messages envoyés par un serveur dans les 4 cas

Le goulot d'étranglement est toujours dû à l'atteinte de la charge maximale par les serveurs (figure 3.34) et implique une perte des messages par ces serveurs (figures 3.35, 3.36).

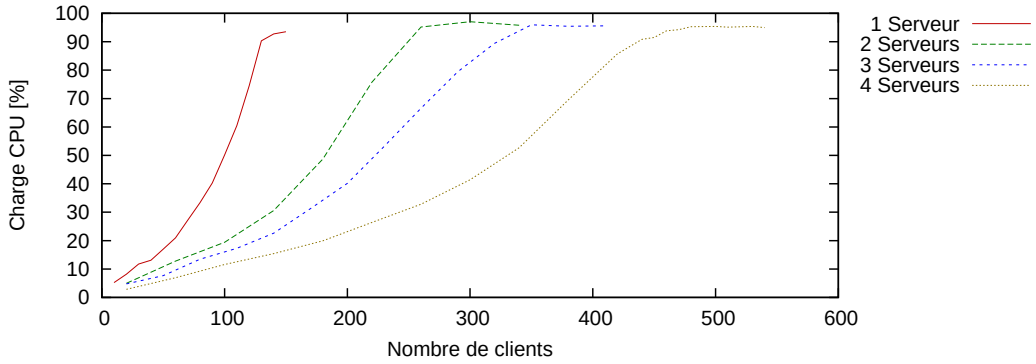

FIG. 3.34 – La charge d'un serveur

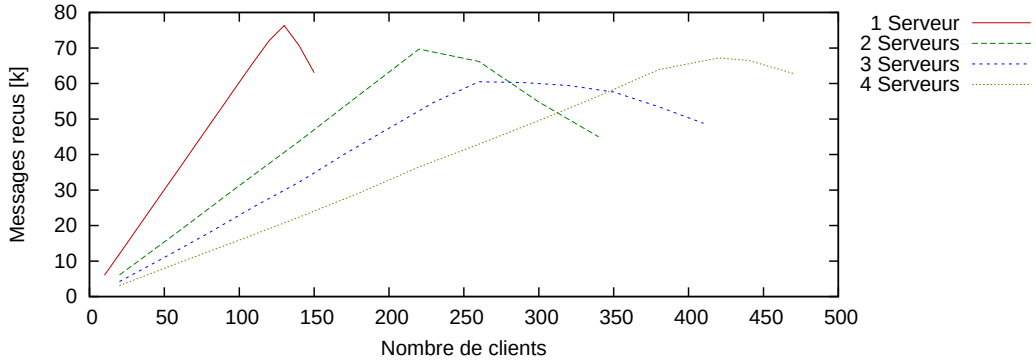

FIG. 3.35 – Les messages reçus par un serveur dans les quatre cas

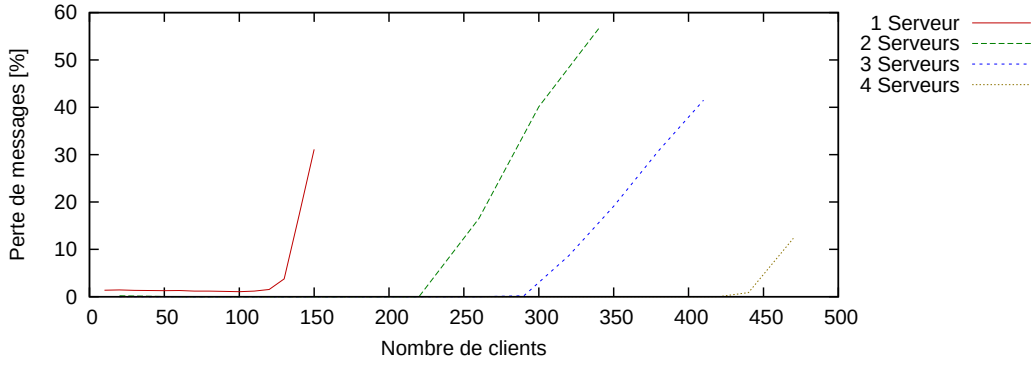

FIG. 3.36 – Le taux de perte de messages d'un serveur

D'autre part, l'utilisation d'une architecture client/multi-serveurs rajoute des points de relais supplémentaires au transfert des messages de mise à jour entre des entités gérées par des serveurs différents. Pour ceci, on a mesuré dans la figure 3.37 la latence des messages avec l'utilisation d'un nombre différent de serveurs. On remarque que la latence reste de l'ordre de 12 millisecondes tant que le système n'est pas surchargé quelque soit le nombre de serveurs utilisés. L'ajout d'un serveur maintient plus longtemps un niveau acceptable de latence sans présenter une augmentation problématique de la latence.

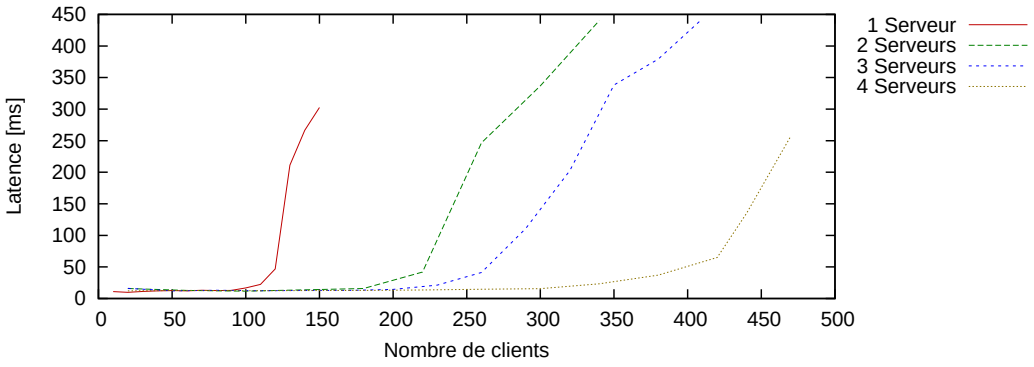

FIG. 3.37 – La latence des messages de mise à jour dans les quatre cas

Il faut noter que nos serveurs sont hébergés sur le même LAN, ce qui permet d'avoir un niveau acceptable de latence inter-serveur. On peut remarquer que la plupart des applications massivement multi-utilisateurs (MMOGs essentiellement) utilisent également des LAN comme réseau connectant leurs serveurs pour la même raison.

3.4.3 Conclusion

On a vu que l'architecture client/multi-serveurs résout efficacement le problème de passage à l'échelle et reste efficace avec l'augmentation du nombre serveurs. L'application passe à l'échelle d'une manière régulière malgré l'augmentation des intérêts que partagent les serveurs entre eux.

3.5 Les modes de transmission

Pour étudier l'utilité des divers modes de transmission, nous avons échangé l'unicast et le multicast dans les communications client/serveur et serveur/serveur (cf. 2.4). Pour utiliser le multicast, nous avons attribué les groupes multicast aux régions de l'environnement et par suite

aux serveurs qui gèrent ces régions. Les communications multicast client/serveur signifient que les clients joignent le groupe multicast de leur serveur. Le serveur envoie à tous ses clients les messages de mise à jour qui les intéressent à travers ce groupe. D'autre part, les communications multicast entre serveurs signifient qu'un serveur joint le groupe multicast d'un autre serveur lorsqu'il est intéressé par une entité de l'autre serveur. Ainsi un serveur envoit les mises à jour des entités qui intéressent d'autres serveurs à travers ce groupe multicast.

Ceci nous donne quatre variations de notre architecture client/multi-serveurs :

1. architecture avec une communication unicast client/serveur et multicast serveur/serveur ;
2. architecture avec de l'unicast entre les clients et les serveurs et entre les serveurs ;
3. architecture avec du multicast client/serveur et de l'unicast serveur/serveur ;
4. architecture avec du multicast client/serveur et serveur/serveur.

Nous avons évalué le fonctionnement de ces variations pour déduire les avantages et les inconvénients de chacune et les cas où elle est utile.

3.5.1 Paramètres du filtrage

On a expérimenté les quatre architectures dans un environnement divisé en trois régions égales, gérées par trois serveurs. Les interactions ont lieu dans le média visuel. Nos expériences ont été menées avec les entités homogènes et hétérogènes.

3.5.2 Résultats

3.5.2.1 Entités homogènes

Charge sur les serveurs

Les serveurs des différentes architectures atteignent leurs charges maximales avec le même nombre de clients (figure 3.38). Le changement du mode de transmission n'a donc pas affecté le passage à l'échelle parce que le goulot d'étranglement est dû à la charge de calcul et non à la charge du réseau. Cette charge évolue d'une manière identique dans toutes les architectures puisqu'elle est due au coût du filtrage.

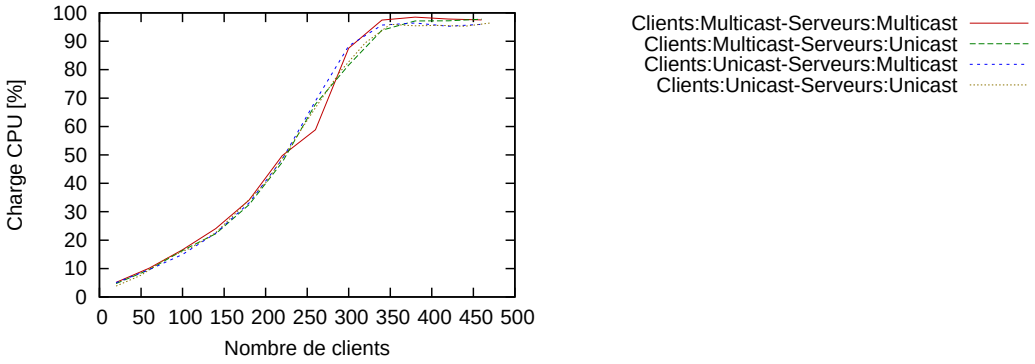

FIG. 3.38 – La charge chez le serveur dans les différentes architectures

Charge du réseau chez le serveur

Au cas où la bande passante aurait été limitée, le goulot d'étranglement serait arrivé plus tôt dans certaines architectures que dans d'autres, ce qui aurait sûrement affecté le passage à l'échelle. Mais puisque notre serveur a une bande passante de 3 Gbits/s, le goulot d'étranglement ne provient pas du réseau. Mais les applications commerciales qui ont des clients distribués géographiquement partout dans le monde sont très loin de jouir d'une bande passante équivalente. Ces applications ont intérêt à minimiser la bande passante payante qu'elles utilisent.

Bande passante entrante nécessaire Il faut noter que lorsque les serveurs communiquent entre eux en unicast, ils ne reçoivent des autres serveurs que les messages dont ils ont besoin. Mais lorsqu'ils communiquent en multicast, un serveur peut recevoir les messages d'une entité qui ne l'intéresse pas mais intéresse un autre serveur puisque tous les messages de mise à jour sont envoyés à travers le même groupe multicast.

Ainsi on peut voir dans la figure 3.39 que le nombre de messages reçus par les applications dont les serveurs communiquent par multicast est plus élevé que les autres. Et plus le nombre de clients augmente, plus les intérêts entre serveurs augmentent ainsi que la réception de messages inutiles. Ainsi l'utilisation du multicast entre les serveurs n'a pas d'avantages vis-à-vis de l'utilisation de la bande passante entrante (figure 3.40).

Bande passante sortante nécessaire Dans la figure 3.41, on voit que les serveurs qui communiquent avec leurs clients en multicast envoient beaucoup moins de messages que ceux

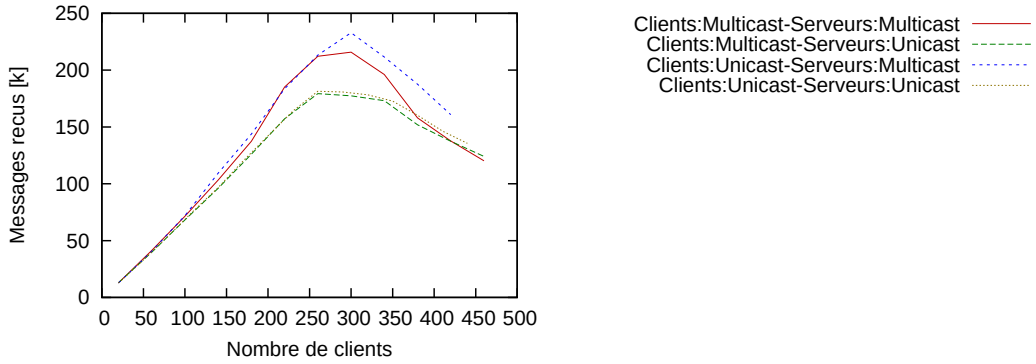

FIG. 3.39 – Le nombre de messages reçus par le serveur dans les différentes architectures

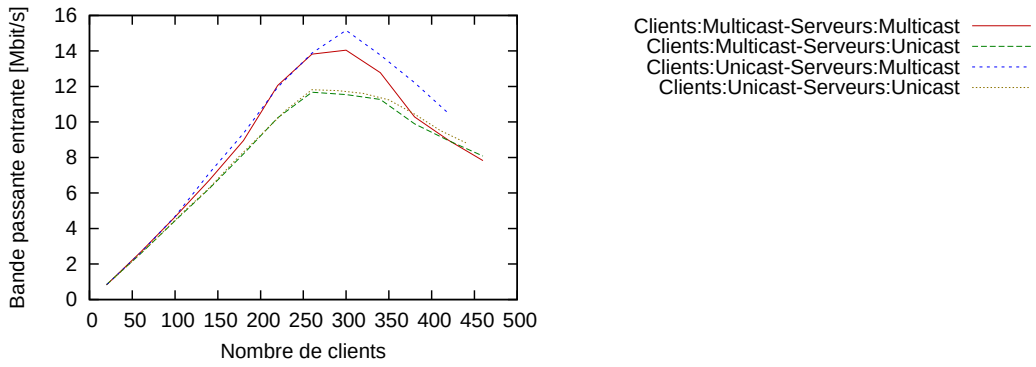

FIG. 3.40 – La bande passante entrante nécessaire pour un serveur

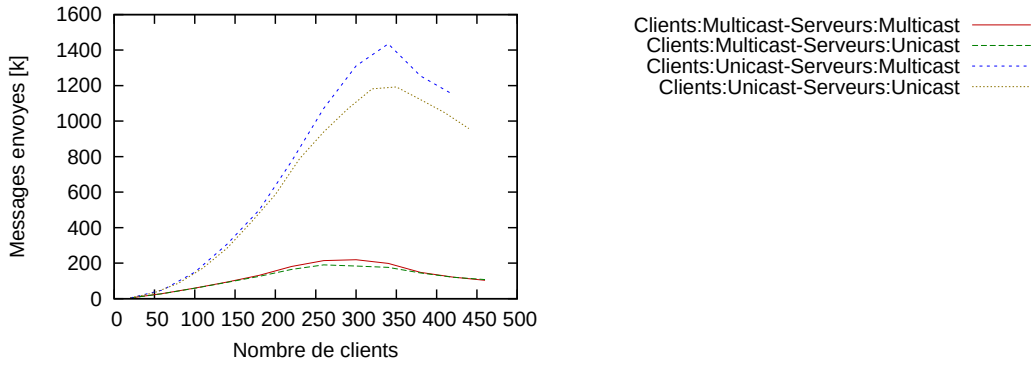

FIG. 3.41 – Le nombre de messages envoyés par chaque serveur

qui communiquent avec leurs clients en unicast. En particulier, les serveurs qui communiquent entre eux en unicast envoient encore moins de messages. Ceci est dû au fait que la communication multicast inter-serveur nécessite que les serveurs échangent des messages supplémentaires concernant les notifications d'abonnement et de désabonnement, ce qui s'est avéré assez coûteux au niveau communication.

Les serveurs utilisant l'unicast pour communiquer avec les clients consomment donc plus de ressources réseau. En particulier, l'architecture qui utilise le multicast entre serveur a utilisé le plus de ressources réseau.

Enfin, les architectures ayant des serveurs qui communiquent par multicast utilisent plus de bande passante entrante (figure 3.40) par rapport aux autres. En plus, elles surchargent les serveurs avec des messages qui ne les intéressent pas.

D'autre part, les architectures utilisant l'unicast pour envoyer les messages à leurs clients utilisent également beaucoup plus de bande passante sortante pour leur communiquer les messages qui les intéressent (figure 3.42).

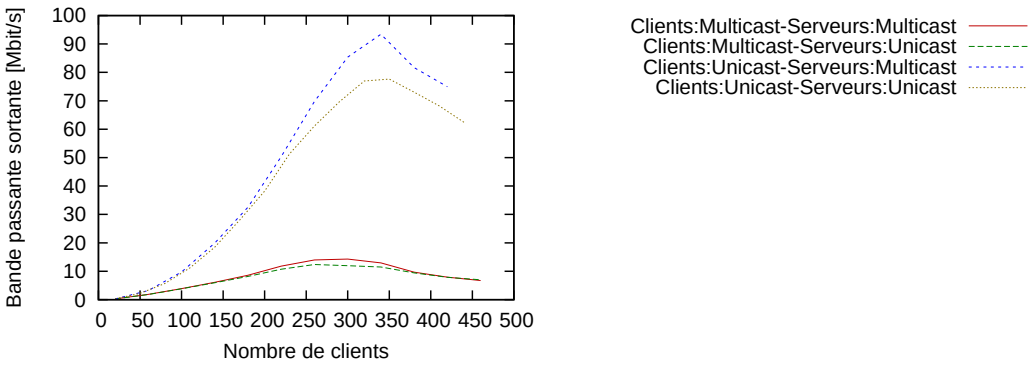

FIG. 3.42 – La bande passante sortante nécessaire au serveur

Charge du réseau chez les clients

Le nombre de messages envoyés par un client est toujours constant quel que soit le nombre total de clients de l'application. La bande passante sortante nécessaire n'est pas très élevée puisque le pas de simulation est de 200 millisecondes.

Par contre, avec l'augmentation du nombre de participants, un client devient intéressé par plus d'entités et reçoit donc plus de messages de mise à jour. L'augmentation des messages

de mise à jour n'est pas très rapide quel que soit le mode de transmission adopté. Mais ce qui est intéressant à étudier est la différence des taux de réception de messages entre l'unicast et le multicast. En fait, lorsqu'un serveur envoie à ses clients les mises à jour en unicast, il envoie à chaque client les messages qui l'intéressent. Un client ne risque alors pas de recevoir un message qui ne l'intéresse pas. Par contre, quand le serveur envoie les messages à travers son groupe multicast, un client reçoit les messages qui l'intéresse mais reçoit aussi les messages qui intéressent les autres clients de sa région (figure 3.43).

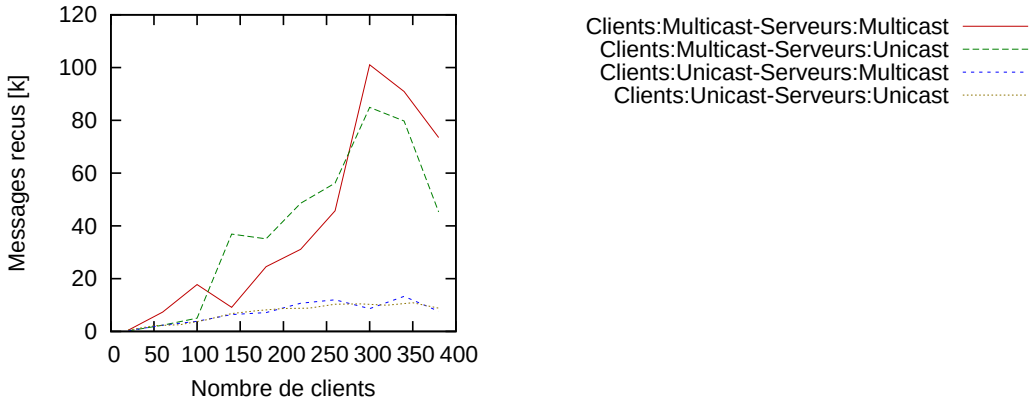

FIG. 3.43 – Les messages reçus par un client

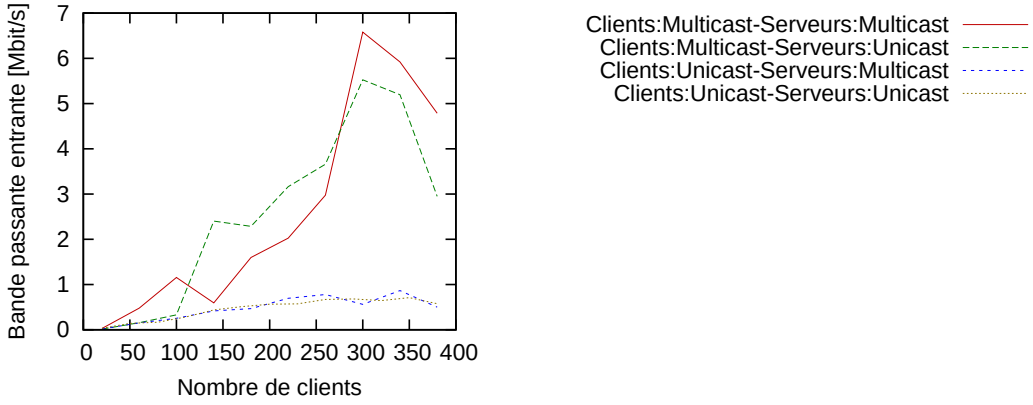

FIG. 3.44 – La bande passante entrante nécessaire pour un client

Ainsi un client a besoin de plus de bande passante entrante quand son serveur utilise le multicast (figure 3.44). La bande passante sortante maximale nécessaire pour le multicast est de 6.5 Mbits/s, ce qui n'est pas trop problématique pour une connexion haut débit. Mais dans le cas où le client a une connexion bas débit, sa bande passante sera vite saturée et son application

sera moins interactive.

Charge de calcul chez les clients

La charge chez les clients devient plus élevée lorsqu'ils reçoivent leurs messages via multicast parce qu'ils ne reçoivent pas que les messages qui les intéressent. Cependant, le taux d'augmentation de messages n'est pas contraignant, il permet à la charge CPU de rester dans des proportions raisonnables. Par exemple, avec une simulation de 300 participants, un client reçoit en moyenne 8600 messages par unicast, il aura une charge de 5%. Sinon il reçoit pas multicast 101000 messages et aura une charge de 15%, ce qui est tout à fait acceptable.

Conclusion

Nous avons vu qu'avec les entités homogènes, l'utilisation du multicast entre serveurs est plus coûteuse que l'utilisation de l'unicast. Elle impose plus de charge sur les serveurs et nécessite plus de bande passante entrante et sortante.

D'autre part, l'utilisation du multicast pour la communication client/serveur permet d'économiser beaucoup de bande passante chez les serveurs. Mais elle nécessite que les utilisateurs aient une connexion haut débit pour pouvoir recevoir d'une manière interactive les messages de mise à jour. Le choix de l'utilisation du multicast comme mode de transmission des serveurs vers les clients dépend donc des moyens et des besoins de l'application : si l'application cible des utilisateurs ayant une connexion haut débit, elle pourra économiser sa bande passante sortante en utilisant le multicast. Par contre, si l'application vise des utilisateurs ayant des connexions bas débit, elle ne pourra pas bénéficier des économies du multicast. Une solution hybride pourrait être considérée, c'est-à-dire un utilisateur ayant une connexion haut débit joint le groupe multicast de son serveur alors qu'un utilisateur ayant une connexion bas débit reçoit ses messages de mise à jour de son serveur via unicast.

3.5.2.2 Entités hétérogènes

L'intérêt des expériences utilisant des entités hétérogènes est de fournir des conditions optimales pour le multicast. En fait, avec les entités hétérogènes les grandes entités sont tellement

importantes qu'elles intéressent des utilisateurs de toutes les régions. Alors les serveurs externes ont plus de chance d'être intéressés par les mêmes entités, alors ils recevront moins de messages non pertinents. De même, les clients d'une région seront intéressés par plus d'entités communes, ils recevront également moins de messages non intéressants. Voyons donc si dans ces conditions optimales le multicast sera plus avantageux qu'il l'était avec les entités homogènes.

On peut voir dans la figure 3.45 que le nombre de messages reçus par les applications dont les serveurs communiquent par multicast continue à être plus élevé que les autres même avec les entités hétérogènes. Donc l'utilisation du multicast entre serveurs n'a pas d'avantages vis-à-vis de la consommation de la bande passante entrante dans tous les cas possibles (figure 3.46).

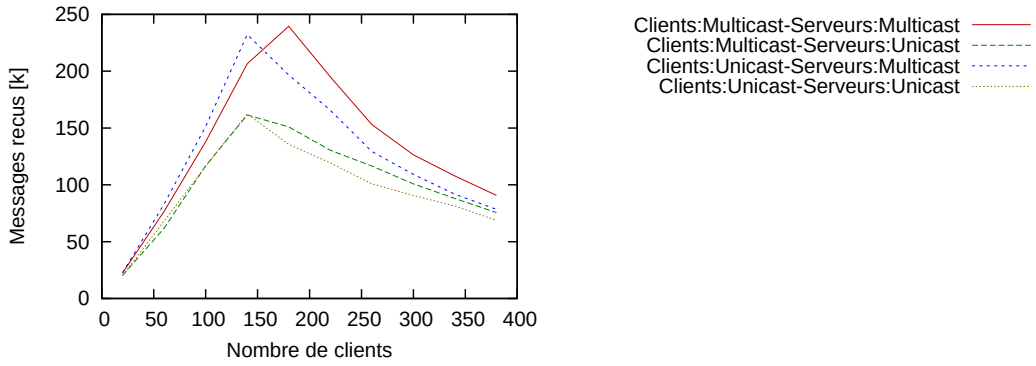

FIG. 3.45 – Les messages reçus par le serveur

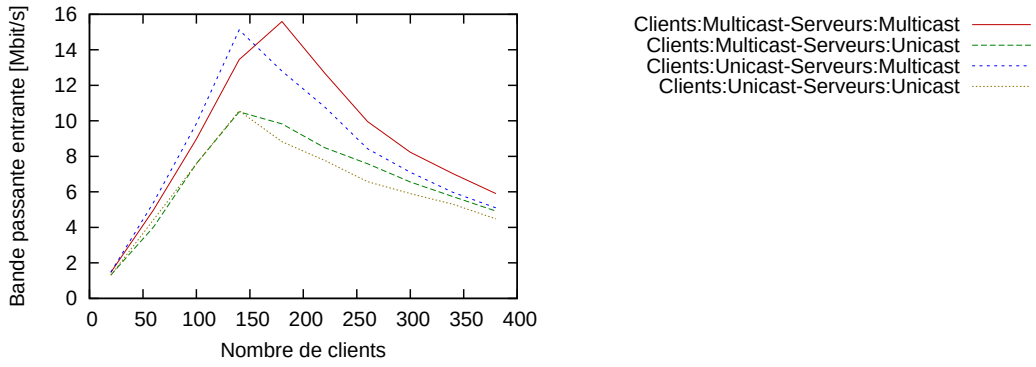

FIG. 3.46 – La bande passante entrante nécessaire pour un serveur

Bande passante sortante nécessaire On voit dans la figure 3.47, qu'avec les entités hétérogènes, l'écart entre les messages envoyés aux clients par unicast et multicast devient encore plus important. En particulier, l'architecture qui utilise le multicast entre serveur a consommé

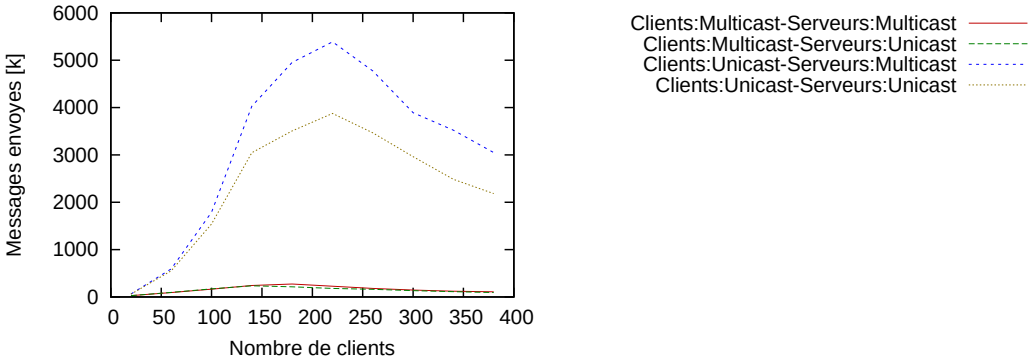

FIG. 3.47 – Les messages envoyés par un serveur

beaucoup plus de ressources réseau.

Ainsi les architectures utilisant le multicast pour communiquer entre les serveurs restent non avantageuses vis-à-vis de la communication de la bande passante entrante (figure 3.46), en plus de la surcharge des serveurs. Et les architectures utilisant l'unicast pour communiquer avec les clients continuent à consommer beaucoup plus de bande passante sortante chez les serveurs (figure 3.48).

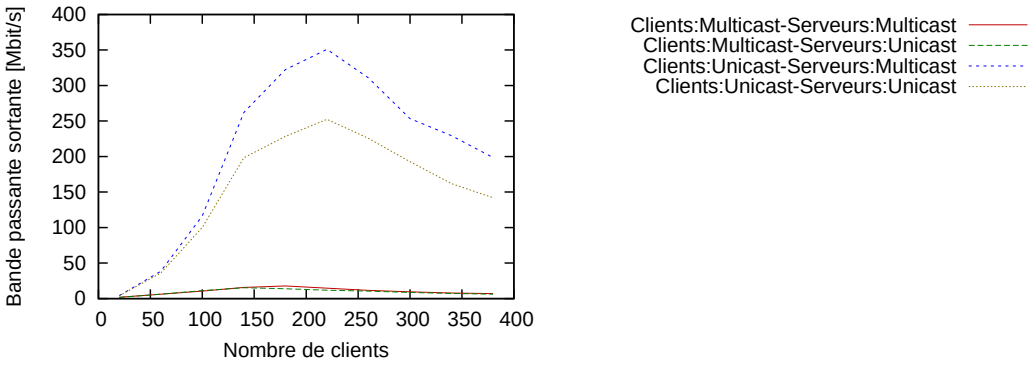

FIG. 3.48 – La bande passante sortante nécessaire pour un serveur

3.5.3 Conclusion

Dans le cas où l'utilisation du multicast est possible (avec l'utilisation de l'IPv6 prochainement) ou du multicast applicatif, l'utilisation de multicast comme mode de communication entre les serveurs et leurs clients est assez intéressante pour économiser de la bande passante réseau du côté du serveur. Mais ceci n'est possible que si les utilisateurs possèdent une connexion haut débit qui leur permet de recevoir plus de messages qu'avec l'unicast. Dans le cas où tous les

clients ne possèdent pas une connexion haut débit, une architecture hybride peut être établie. Dans cette architecture, le serveur envoie les messages en unicast à ses clients ayant un bas débit et en multicast aux clients ayant un haut débit.

Par contre, l'utilisation du multicast comme mode de communication entre les serveurs n'a pas d'avantages vis-à-vis de la consommation de la bande passante ni de la charge de calcul.

4

Conclusion et perspectives

Vivre une expérience immersive et satisfaisante au sein d'un environnement virtuel distribué exige de l'interactivité, de la cohérence et un passage à l'échelle performant. Ces exigences sont contradictoires parce qu'elles nécessitent à la fois d'augmenter et de baisser le taux d'échange de messages. La gestion d'échange de messages est donc un élément essentiel pour l'évolution des environnements virtuels distribués. Pour ces raisons, nous avons étudié les méthodes de filtrage existantes essayant de répondre aux besoins des environnements virtuels distribués.

Nous avons constaté que chacune de ces méthodes possède un fonctionnement différent qui lui permet de répondre à des aspects importants pour l'adaptation aux différents domaines d'application. Ces aspects sont l'utilisation de plusieurs médias de communication, l'expression de la manifestation en plus des intérêts des participants, l'établissement de relations asymétriques, l'adaptation aux environnements ouverts ainsi qu'aux architectures emboîtées. D'autres aspects essentiels pour le passage à l'échelle tel que l'utilisation du multicast et la continuité du passage à l'échelle ont également été abordés.

Notre objectif était de présenter une méthode de filtrage qui répond à tous ces aspects en même temps et d'une manière efficace et simple. Ceci a été réalisé à travers "la gestion d'effet" qui présente comme atouts :

- d'être une extension du modèle spatial d'interaction permettant l'établissement de relations asymétriques en une seule étape au lieu de deux étapes ;
- la définition des intérêts et des manifestations visuels des participants est effectuée d'une manière exacte à l'aide d'un mécanisme basé sur les règles de la perception graphique ;

- le système de mise en œuvre de la gestion d'effet utilise une architecture client/multi-serveurs relié à une division de l'espace en régions. Cette structure gère les intérêts des participants entre les régions d'une manière optimisée. Ceci permet d'adapter l'application aux environnements ouverts ainsi qu'aux architectures emboîtées ;
- l'utilisation d'une architecture client/multi-serveurs permet un passage à l'échelle continu ;
- l'intégration du multicast fournit une économie supplémentaire de la bande passante.

Dans le cadre de l'évaluation de la gestion d'effet, nous avons comparé la gestion d'effet aux autres méthodes de gestion d'intérêt. Pour établir une comparaison équitable entre les méthodes, nous avons conçu des mécanismes de définition des intérêts des participants dans les autres méthodes d'intérêt en se basant sur les règles de perception. L'évaluation de la gestion d'effet a montré qu'elle définit d'une manière plus exacte la manifestation et l'intérêt de chaque participant dans chaque média, ce qui lui permet d'effectuer un filtrage plus exact et donc plus économe des données que les autres méthodes de gestion d'intérêt.

Le système a également réussi à prouver sa capacité à passer à l'échelle d'une manière régulière malgré l'augmentation des intérêts entre les serveurs accompagnant l'augmentation du nombre de clients et par suite de serveurs. De plus, l'utilisation du multicast par les serveurs a prouvé son utilité pour minimiser l'utilisation de la bande passante sortante des serveurs.

En perspective de ces travaux de thèse, nous pensons qu'il est nécessaire de supporter le changement dynamique des intérêts et des manifestations des participants. Ceci est assez essentiel parce que les environnements virtuels peuvent offrir aux participants des outils virtuels ou des dispositifs d'interaction leur permettant de changer leurs capacités durant la simulation. La difficulté de ce changement dynamique provient de l'obligation du changement des zones d'effet et de conscience de toutes les entités après le changement de capacité d'un seul participant. Ceci est dû au fait que les zones d'effet et de conscience des entités sont inter-dépendantes et leurs valeurs résultent de la solution d'un système global d'inéquations.

Malgré ses inconvénients, le filtrage basé sur la perception paraît une méthode de filtrage assez intéressante en elle-même. Cette méthode n'est pas encore assez mûre, elle a besoin de plus d'approfondissement, de développement et d'évaluation.

En ce qui concerne la détermination des zones de conscience et d'effet, il nous semble important de concevoir un mécanisme qui détermine la taille des zones de conscience et d'effet

sonores.

La conception d'une division dynamique de l'espace est aussi essentielle pour l'équilibrage de charge et la résolution des problèmes de regroupement. Cette division doit s'adapter aux intérêts des participants, à leur distribution géographique, à la capacité des serveurs et à l'architecture adoptée.

Bibliographie

[AFT06] Bruno Arnaldi, Philippe Fuchs, and Jacques Tisseau. Introduction à la Réalité Virtuelle . In Philippe Fuchs and Guillaume Moreau, editors, *Traité de le Réalité Virtuelle : L'homme et l'environnement virtuel*, volume 1, pages 1–21. Les Presses de l'Ecole des Mines de Paris, 3 edition, mars 2006. ISBN 2-911762-62-2.

[AWZ98] Howard Abrams, Kent Watsen, and Michael Zyda. Three-tiered interest management for large-scale virtual environments. In *VRST '98 : Proceedings of the ACM symposium on Virtual reality software and technology*, pages 125–129, New York, NY, USA, 1998. ACM Press.

[BF93] Steve Benford and Lennart E. Fahlen. A spatial model of interaction in large virtual environments. In *ECSCW*, pages 107–, 1993.

[BFV06a] Alain Berthoz, Philippe Fuchs, and Jean-Louis Vercher. Immersion et présence. In Philippe Fuchs and Guillaume Moreau, editors, *Traité de le Réalité Virtuelle : L'homme et l'environnement virtuel*, volume 1, pages 309–337. Les Presses de l'Ecole des Mines de Paris, 3 edition, mars 2006. ISBN 2-911762-62-2.

[BFV06b] Alain Berthoz, Philippe Fuchs, and Jean-Louis Vercher. Latence et temps de transmission de l'information. In Philippe Fuchs and Guillaume Moreau, editors, *Traité de le Réalité Virtuelle : L'homme et l'environnement virtuel*, volume 1, pages 339–355. Les Presses de l'Ecole des Mines de Paris, 3 edition, mars 2006. ISBN 2-911762-62-2.

[BG97] Steve Benford and Chris Greenhalgh. Introducing third party objects into the spatial model of interaction. In *ECSCW'97 : Proceedings of the fifth conference on European Conference on Computer-Supported Cooperative Work*, pages 189–204, Norwell, MA, USA, 1997. Kluwer Academic Publishers.

[BM05] A. Boukerche and N. J. McGraw. A grid-filtered region-based approach to support synchronization in large-scale distributed interactive virtual environments. In *ICPPW '05 : Proceedings of the 2005 International Conference on Parallel Processing Workshops (ICPPW'05)*, pages 525–530, Washington, DC, USA, 2005. IEEE Computer Society.

[BMDL05] Azzedine Boukerche, Nathan J. McGraw, Caron Dzermajko, and Kaiyuan Lu. Grid-filtered region-based data distribution management in large-scale distributed simulation systems. In *ANSS '05 : Proceedings of the 38th annual Symposium on Simulation*, pages 259–266, Washington, DC, USA, 2005. IEEE Computer Society.

[BMH98] Dirk Bartz, Michael Meißner, and Tobias Huttner. Extending graphics hardware for occlusion queries in opengl. In *ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*. ACM Press, 1998.

[BR00] A. Boukerche and A. Roy. In search of data distribution management in large scale distributed simulations. In *SCSC : Proceedings of the Summer Computer Simulation Conference*. The Society for Modeling and Simulation International (SCS), IEEE, 2000.

[BWA96] John W. Barrus, Richard C. Waters, and David B. Anderson. Locales : Supporting large multiuser virtual environments. *IEEE Comput. Graph. Appl.*, 16(6) :50–57, 1996.

[CBC06] Sabine Coquillart, Jean-Marie Burkhardt, and Collectif. In Philippe Fuchs and Guillaume Moreau, editors, *Traité de le Réalité Virtuelle : L'interfaçage, l'immersion et l'interaction en environnement virtuel* , volume 2. Les Presses de l'Ecole des Mines de Paris, 3 edition, mars 2006. ISBN 2-911762-63-0.

[CDD+05] Franck Cappello, Frederic Desprez, Michel Dayde, Emmanuel Jeannot, Yvon Jegou, Stephane Lanteri, Nouredine Melab, Raymond Namyst, Pascale Primet, Olivier Richard, Eddy Caron, Julien Leduc, and Guillaume Mornet. A large scale, reconfigurable, controlable and monitorable grid platform. In *6th IEEE/ACM International Workshop on Grid Computing*, 2005.

[CDG+93] J. Calvin, A. Dickens, B. Gaines, P. Metzger, D. Miller, and D. Owen. the simnet

virtual world architecture. In *Virtual Reality Annual International Symposium*, pages 450–455. IEEE Computer Society, 1993.

[CLC99] Wentong Cai, Francis B. S. Lee, and L. Chen. An auto-adaptive dead reckoning algorithm for distributed interactive simulation. In *PADS '99 : Proceedings of the thirteenth workshop on Parallel and distributed simulation*, pages 82–89, Washington, DC, USA, 1999. IEEE Computer Society.

[Com94] Steering Committee. The dis vision, a map to the future of distributed simulation. Technical report, 1994.

[CW96] J.O Calvin and R. Weatherly. An introduction to the high level architecture (hla) runtime infrastructure (rti). In *14th DIS Workshop*, 1996.

[DG03] Thomas P. Duncan and Denis Gračanin. Algorithms and analyses : pre-reckoning algorithm for distributed virtual environments. In *WSC '03 : Proceedings of the 35th conference on Winter simulation*, pages 1086–1093. Winter Simulation Conference, 2003.

[EHTJ07] Souad Elmerhebi, Jean-Christophe Hoelt, Patrice Torguet, and Jean-Pierre Jessel. Perception based network messages filtering for massively multiplayer online games. In *Cybergames 2007, Manchester, UK, 10/09/07-11/09/07*, pages 125–132. Manchester Metropolitan University, septembre 2007.

[Eri94] Hans Eriksson. Mbone : The multicast backbone. *Commun. ACM*, 37(8) :54–60, 1994.

[ETJ06] Souad Elmerhebi, Patrice Torguet, and Jean-Pierre Jessel. Realism and communication evaluation of effect management in distributed virtual environments. In *Eurographics Symposium on Virtual Environments, Lisbon, Portugal, 08/05/06-10/05/06*, http ://www.eg.org/, 2006. Eurographics.

[ETJH07] Souad Elmerhebi, Patrice Torguet, Jean-Pierre Jessel, and Jean-Christophe Hoelt. Asset : A scalable multicast multi-server based distributed virtual environments system. In Franz-Erich Wolter and Alexei Sourin, editors, *International Conference on Cyberworlds, Hanovre, Allemagne, 24/10/07-27/10/07*, pages 179–186, http ://www.computer.org, octobre 2007. IEEE Computer Society.

[ETRJ06] Souad Elmerhebi, Patrice Torguet, Nancy Rodriguez, and Jean-Pierre Jessel. The effect management in distributed virtual environments. In *IEEE International Symposium and School on Advance Distributed Systems (ISSADS), Guadalajara, México, 25/01/06-27/01/06*, http ://www.springerlink.com/, 2006. Springer-Verlag.

[Far99] Nicolas Farcet. *Perception et filtrage spatial dans les environnements virtuels distribués* . Thèse de doctorat, Université Paris 6, Paris, France, 1999.

[FT98] Nicolas Farcet and Patrice Torguet. Space-scale structure for information rejection in large-scale distributed virtual environments. In *IEEE Virtual Reality Annual International Symposium 1998 (VRAIS 98) , Atlanta, Géorgie,*, pages 276–283. ., mars 1998. Dates de conférence : mars 1998 1998.

[Fun95] Thomas A. Funkhouser. Ring : a client-server system for multi-user virtual environments. In *SI3D '95 : Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 85–ff., New York, NY, USA, 1995. ACM Press.

[Fun96] Thomas Funkhouser. Network topologies for scalable multi-user virtual environments. In *VRAIS '96 : Proceedings of the 1996 Virtual Reality Annual International Symposium (VRAIS 96)*, page 222, Washington, DC, USA, 1996. IEEE Computer Society.

[GB95] Chris Greenhalgh and Steve Benford. MASSIVE : A distributed virtual reality system incorporating spatial trading. In *International Conference on Distributed Computing Systems*, pages 27–34, 1995.

[GB97] C. Greenhalgh and S. Benford. Boundaries, awareness and interaction in collaborative virtual environments, 1997.

[GPS00] Chris Greenhalgh, Jim Purbrick, and Dave Snowdon. Inside massive-3 : flexible support for data consistency and world structuring. In *CVE '00 : Proceedings of the third international conference on Collaborative virtual environments*, pages 119–127, New York, NY, USA, 2000. ACM Press.

[Gre94] C. Greenhalgh. An experimental implementation of the spatial model. In *6th ERCIM workshop*, pages 53–71, 1994.

[Gre96] Chris Greenhalgh. Dynamic, embodied multicast groups in massive-2. Technical Report NOTTCS-TR-96-8, Department of Computer Science, The University of Nottingham, December 1996.

[HLL00] Seunghyun Han, Mingyu Lim, and Dongman Lee. Scalable interest management using interest group based filtering for large networked virtual environments. In *VRST '00 : Proceedings of the ACM symposium on Virtual reality software and technology*, pages 103–108, New York, NY, USA, 2000. ACM Press.

[HRC96] Daniel J. Van Hook, Steven J. Rak, and James O. Calvin. Approaches to rti implementation of hla data distribution management services. In *15th DIS workshop*, 1996.

[JT06] Jean-Pierre Jessel and Patrice Torguet. Réalité Virtuelle Distribuée . In Philippe Fuchs and Guillaume Moreau, editors, *Traité de le Réalité Virtuelle : Outils et modèles informatiques des environnements virtuels* , volume 3, pages 411–423. Les Presses de l'Ecole des Mines de Paris, 3 edition, février 2006. ISBN 2-911762-64-9.

[Kaz93] Rick Kazman. Making WAVES : On the design of architectures for low-end distributed virtual environments. In *VR*, pages 443–449, 1993.

[Man00] T. Manninen. Interaction in networked virtual environments as communicative action - social theory and multi-player games. In *CRIWG conference*, pages 154–157. IEEE Press, 2000.

[MKZB03] Don McGregor, Andrzej Kapolka, Michael Zyda, and Don Brutzman. Requirements for large-scale networked virtual environments. In *the 7th International Conference on Telecommunications ConTel 2003*, pages 353–358, 2003.

[ML03] G. Morgan and F. Lu. Predictive interest management : An approach to managing message dissemination for distributed virtual environments. In *Richmedia2003 : Proceedings of the First International Workshop on Interactive Rich Media Content Production : Architectures, Technologies, Applications, Tools (Richmedia2003)*, 2003.

[Mor96] Katherine L. Morse. Interest management in large-scale distributed simulations. Technical Report ICS-TR-96-27, 1996.

[MSL04] Graham Morgan, Kier Storey, and Fengyun Lu. Expanding spheres : A collision detection algorithm for interest management in networked games. In *ICEC*, pages 435–440, 2004.

[MZ01] Katherine Morse and Michael Zyda. Multicast grouping for data distribution management. In *Computer Simulation Methods and Applications Conference*, 2001.

[MZP+95] M. R. Macedomia, M. J. Zyda, D. R. Pratt, D. P. Brutzman, and P. T. Barham. Exploiting reality with multicast groups : a network architecture for large-scale virtual environments. In *VRAIS '95 : Proceedings of the Virtual Reality Annual International Symposium (VRAIS'95)*, page 2, Washington, DC, USA, 1995. IEEE Computer Society.

[PG00] James Purbrick and Chris Greenhalgh. Extending locales : Awareness management in massive-3. In *VR '00 : Proceedings of the IEEE Virtual Reality 2000 Conference*, page 287, Washington, DC, USA, 2000. IEEE Computer Society.

[Psh70] B.N. Pshenichnyi. Newton's method for the solution of systems of equalities and inequalities. *Mathematical Notes, USSR*, 8(5) :827–830, 1970.

[RJT02] Nancy Rodriguez, Jean-Pierre Jessel, and Patrice Torguet. A virtual reality tool for teleoperation research. *Virtual Reality Society Journal*, 6(2) :57–62, 2002.

[Rod03] Nancy Rodriguez. *ASSET : une architecture générale pour la télérobotique* . Thèse de doctorat, Université Paul Sabatier, Toulouse, France, janvier 2003.

[SLM04] Kier Storey, Fengyun Lu, and Graham Morgan. Determining collisions between moving spheres for distributed virtual environments. In *CGI '04 : Proceedings of the Computer Graphics International (CGI'04)*, pages 140–147, Washington, DC, USA, 2004. IEEE Computer Society.

[Ste96] W. Richard Stevens. *TCP/IP illustrated (vol. 3) : TCP for transactions, HTTP, NNTP, and the Unix domain protocols.* Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1996.

[TTS07] Samir Torki, Patrice Torguet, and Cédric Sanza. Adaptive classifier system-based dead reckoning. In *Eurographics Symposium on Virtual Environments (EGVE), Wei-*

mar, Allemagne, 15/07/2007-18/07/2007, volume virtual environments 2007, pages 101–108, http ://www.eg.org/, juillet 2007. Eurographics.

[YC06] M. Ye and L. Cheng. System-performance modeling for massively multiplayer online role-playing games. *IBM Syst. J.*, 45(1) :45–58, 2006.

[YLE04] C. K. Yeo, B. S. Lee, and M. H. Er. A survey of application level multicast techniques. *Computer Communications*, 27(15) :1547–1568, 2004.