



Amélioration de la réactivité des réseaux pair à pair pour les MMOGs

Rapport Bibliographique

Xavier Joudiou

Encadré par: Sergey Legtchenko & Sébastien Monnet

21/05/10



Table des matières

1	Introduction	3
2	Présentation du Laboratoire d' Informatique de Paris 6	4
2.1	Le laboratoire et l'équipe REGAL	4
2.2	L'INRIA Paris - Rocquencourt	5
3	Vers des solutions distribuées pour les MMOG	6
3.1	Les architectures actuelles pour les MMOGs	6
3.2	Le pair à pair, la solution ?	7
4	Traces des utilisateurs dans les MMOGs	9
4.1	Les différentes techniques de récupération de trace	9
4.1.1	Les objectifs et les techniques de collecte de trace	9
4.1.2	Limitations de la collecte des traces	9
4.2	Observations des traces	10
4.2.1	Hotspots	10
4.2.2	Waypoints	10
5	Systèmes P2P exploitables par les MMOGs distribués	11
5.1	Area Of Interest	11
5.2	Découpage de la carte	12
5.3	Overlays	13
5.3.1	Semantic overlay	13
5.3.2	Application-Malleable Overlay	14
5.3.3	Voronoi-based Overlay Network	14
6	Un système P2P pour MMOG : Solipsis	16
6.1	Introduction sur Solipsis	16
6.2	Propriétés de Solipsis	16
6.2.1	Propriétés	16
6.2.2	Maintien des propriétés	17
7	Prise en compte de la mobilité : Blue Banana	18
7.1	Solutions introduites	18
7.1.1	Les états de l'avatar	18
7.1.2	Anticipation des mouvements	18
7.2	Expérimentations et Résultats	19
7.2.1	Le simulateur PeerSim et la description des expérimentations . . .	19
7.2.2	Les résultats	20
8	Conclusion	20

Résumé

Depuis plusieurs années, un nouveau type d'architecture des systèmes est apparu. Il s'agit de l'architecture pair à pair, cette architecture est devenue populaire grâce à des applications de partage de fichiers. Nous allons nous intéresser aux jeux vidéos massivement multijoueur (MMOG pour Massively Multiplayer Online Games) qui sont de plus en plus populaires et qui font ressortir des problèmes que l'architecture pair à pair doit pouvoir corriger. Le problème du passage à l'échelle sera l'un des plus importants à résoudre pour permettre à un grand nombre de joueurs de participer simultanément. Nous verrons comment l'architecture pair à pair peut être une des solutions.

Pour remédier à cela, une solution consiste à remplacer le modèle client/serveur par un réseau logique pair à pair (overlay). Malheureusement, les protocoles pair à pair existants sont trop peu réactifs pour assurer la faible latence nécessaire à ce genre d'applications. Néanmoins, quelques travaux ont déjà été menés pour adresser ce problème. L'idée est d'adapter le voisinage de chaque pair afin que toute l'information dont il aura besoin dans l'avenir se trouve proche de lui dans le réseau. Il est alors nécessaire de correctement évaluer les futurs besoins de chaque pair, et de faire évoluer son voisinage à temps. Dans ce rapport bibliographique, nous expliquerons les différentes solutions existantes.

1 Introduction

A travers ce rapport, nous allons observer les différents travaux déjà réalisés sur les applications pair à pair, nous nous intéresserons dans le détail aux jeux vidéos massivement multijoueur. Ces applications impliquent que différentes propriétés soient vérifiées si l'on veut qu'elles soient utilisées par un grand nombre de personnes en même temps et sur une longue durée. L'architecture pair à pair peut répondre efficacement à certaines des différentes propriétés nécessaires au bon fonctionnement des applications mais elle pose un problème au niveau de la latence. Le but du stage est donc d'améliorer les liens du réseau pair à pair pour que l'utilisateur puisse avoir dans son voisinage les données qui lui seront nécessaires aux itérations suivantes, pour cela il faudra anticiper au mieux les mouvements du joueur.

Tout d'abord nous commencerons par une présentation du Laboratoire d'informatique de Paris 6, qui m'accueille pendant ce stage de fin d'étude et nous présenterons brièvement l'INRIA Paris - Rocquencourt qui travaille conjointement avec l'équipe REGAL à laquelle je suis rattaché. Nous expliquerons ensuite pourquoi l'approche pair à pair est celle qui paraît la plus adaptée pour répondre aux différentes problématiques qu'induisent ces applications, nous en profiterons pour rappeler rapidement les caractéristiques des différentes architectures (cf 3, page 6). Les différentes études des traces des avatars dans les environnements virtuels seront ensuite expliquées (cf 4, page 9). Ensuite, les mécanismes permettant une meilleure mise en place de l'architecture pair à pair seront expliqués, nous rentrerons un peu plus dans les détails pour Solipsis car Blue Ba-

nana l'utilise comme base (cf 6, page 16). Après avoir parlé des différents mécanismes déjà existants, nous parlerons alors du travail Blue Banana qu'il faudra essayer d'améliorer lors des mois restants du stage (cf 7, page 18).

2 Présentation du Laboratoire d' Informatique de Paris 6

2.1 Le laboratoire et l'équipe REGAL

Le LIP6 est un laboratoire de recherche sous tutelle de l'Université Pierre et Marie Curie, et du CNRS (UMR 7606). Avec 184 chercheurs permanents et 258 doctorants, il est l'un des principaux laboratoires de recherche en informatique en France. Le Directeur du laboratoire est Patrick Gallinari et le directeur adjoint est Pierre Sens. Le laboratoire vient de revenir à Jussieu après plusieurs années d'exil sur le site de Radio France Avenue de Président Kennedy dans le 16^e arrondissement.

Le laboratoire couvre un large spectre d'activités regroupées au sein de cinq départements : Calcul Scientifique, Décision, Systèmes Intelligents Recherche opérationnelle, Données et Apprentissage Artificiel, Réseaux et Systèmes Répartis, Systèmes Embarqués sur Puce. En complément de la recherche académique, le LIP6 a une longue tradition de coopération avec des partenaires industriels dans de très nombreux projets nationaux, européens ou internationaux.

Deux centres Recherche et Développement ont été créés : le CERME, Centre Européen de Recherche en Micro-Electronique sur les systèmes embarqués, et Euronetlab, sur l'internet et les réseaux de télécommunication. Le LIP6 est également impliqué dans les pôles de compétitivité de l'Ile-de-France : Cap Digital sur le contenu numérique et System@tic sur les systèmes embarqués.

Il a également des équipes communes avec l'INRIA sur les thématiques du calcul formel et des systèmes répartis, dont l'équipe REGAL qui m'accueille pour ce stage. La coopération internationale est une constante pour les activités du laboratoire. Le LIP6 est membre de plusieurs réseaux d'excellence et développe également des relations suivies avec des universités au Brésil, aux États-Unis, au Japon, et dans de nombreux pays européens. Le laboratoire est largement ouvert aux projets de coopération et à l'accueil de visiteurs scientifiques. Le laboratoire est impliqué dans des enseignements liés à la recherche, qui sont dispensés au Master « Sciences et technologie » à l'Université Pierre et Marie Curie. L'EDITE de Paris (Ecole Doctorale d'Informatique, Télécommunication et Electronique de Paris) accueille nos doctorants.

L'équipe REGAL fait partie du département "Réseaux et Systèmes Répartis" qui est dirigé par Bertil Folliot. Le département "Réseaux et Systèmes Répartis" se concentre sur la conception de solutions pour construire et gérer les réseaux et systèmes du futur. Il est constitué de quatre équipes : MoVe (Modélisation et Vérification), REGAL (Répartition et Gestion des Applications à Large Echelle), NPA (Networks and Performances Analysis) et Phare. L'équipe REGAL est une équipe commune avec l'INRIA Rocquencourt, le responsable de cette équipe est Pierre Sens. L'objectif de l'équipe est la gestion de

ressources dans le cadre très dynamique de grands réseaux. L'équipe REGAL s'intéresse aux techniques de déploiement d'applications (code et données) adaptées aux environnements extrêmement distribués de grande taille (nombre de processeurs, distances), fortement dynamiques, hétérogènes, sans possibilité simple de gestion centralisée et/ou instantanée de la connaissance mutuelle. L'approche de REGAL repose sur des techniques de réplication et d'adaptation dynamique dans lesquelles un code applicatif et ses données sont dupliqués sur plusieurs sites ce qui permet de tolérer les fautes, d'augmenter la disponibilité et réduire les temps d'accès du service rendu par l'application. Nous étudions la façon dont les systèmes peuvent garantir une qualité de service en termes de fiabilité, disponibilité et cohérence.

2.2 L'INRIA Paris - Rocquencourt

L'INRIA Paris - Rocquencourt est l'un des huit centres de recherche de l'Institut National de Recherche en Informatique et Automatique (INRIA), organisme de recherche spécialisé dans le domaine des Sciences et Technologies de l'Information et de la Communication. C'est un établissement public placé sous la double tutelle du ministère de la recherche et du ministère de l'économie, des finances et de l'industrie, l'INRIA accueille environ 3 800 personnes dont 2 800 scientifiques.

Combiner l'excellence scientifique et le transfert technologique est le fondement de la stratégie de l'institut, dans laquelle s'inscrit en particulier le centre de recherche INRIA Paris - Rocquencourt. Ainsi, ses objectifs prioritaires de recherche - réseaux et systèmes de communication - logiciels fiables et sécurité - modélisation du vivant et de l'environnement - sont conduits avec le souci de concilier recherche en amont du meilleur niveau international, applications et valorisation par le biais d'interactions constantes avec le monde socio-économique.

Ils mènent ses activités scientifiques en développant des partenariats étroits avec les équipes internationales de pointe, le monde de l'industrie et des services, en particulier dans le cadre des pôles de compétitivité, et de nombreux établissements d'enseignement supérieur et de recherche d'Île-de-France.

3 Vers des solutions distribuées pour les MMOG

Nous souhaitons étudier les différentes raisons qui nous feraient passer d'une architecture Client/Serveur à une architecture pair à pair. Pour cela nous mettrons en évidence les différences des deux solutions, les avantages et les inconvénients d'une approche pair à pair. Il nous a été possible d'observer qu'il est devenu important, pour les éditeurs de MMOGs, d'étudier les différentes architectures en amont de la réalisation du jeu pour ainsi gérer au mieux les différentes dépenses [2] (maintenance des serveurs, ajout de add-ons, etc).

3.1 Les architectures actuelles pour les MMOGs

Dans la plupart des jeux en ligne massivement multi joueur, l'architecture est de type client/serveur (voir page 8). Dans cette architecture, il y a une forte distinction entre le client, qui envoie des requêtes au serveur et attend les réponses, et le serveur qui est à l'écoute de requêtes des clients. Cette approche simplifie la sécurité et le fonctionnement global des jeux. Par exemple, pour effectuer des mises à jour sur l'état global du jeu, il suffit de le faire sur une seule machine (le serveur principal) et il n'y a pas de problème d'incohérence entre les données. De même pour la l'administration et le contrôle des tricheurs, toutes les données étant regroupées sur une seule machine, le contrôle sera plus simple que dans des systèmes distribués.

Un des problèmes que peut rencontrer l'architecture pair à pair est qu'elle aura plus de difficultés pour passer à l'échelle sans mettre de gros moyens en terme de machines et donc financier. Le serveur peut devenir un goulot d'étranglement et si un trop grand nombre de joueurs se connecte, le serveur pourrait avoir des difficultés à tenir [18]. Ce problème est résolu en ayant des serveurs de très grandes capacités ou en mettant en place des clusters de serveur. Ces solutions induisent un gros investissement dès le début de la mise en service du jeu et le coût de maintenance est élevé. Il est donc difficile de mettre cela en place pour des applications *open source* ou ayant peu de moyen, c'est surtout pour celles-ci que les architectures pair à pair peuvent être intéressantes dans l'immédiat. Un autre problème est la disponibilité du système en cas de panne du serveur, si le serveur tombe en panne alors plus personne n'aura accès à l'application que ce dernier faisait fonctionner, à la différence d'une architecture répartie.

Au vue du nombre croissant de participant à ce genre de jeux vidéos massivement multi joueur, le passage à l'échelle devient un sujet très important et c'est pour cela que les recherches sur des architectures distribuées sont de plus en plus importantes (voir Schéma 1).

Année	Nombre de publications *	Nombre de publications **
2002	0	30
2003	0	109
2004	2	235
2005	9	489
2006	9	745
2007	21	1159
2008	19	1464
2009	24	1516

Statistiques récupérées en effectuant des recherches sur le site <http://portal.acm.org/> :

* pour la recherche sur: P2P MMOGs

** pour la recherche sur: P2P overlays

FIGURE 1 – Tableau montrant l'évolution des recherches dans le domaine du P2P (MMOGs et Overlay)

3.2 Le pair à pair, la solution ?

Comme il est dit précédemment, l'augmentation croissante des recherches sur le sujet atteste du fait que des limites ressortent des solutions existantes. Le problème du passage à l'échelle est sûrement le plus important, et il est l'une des raisons principales de toutes ces recherches. Les architectures pair à pair ne font plus ressortir d'entité serveur et client comme nous le connaissions dans l'architecture client/serveur. Chaque nœud sera client et serveur en fonction du temps et en fonction de ses besoins (voir page 8). Les systèmes pair à pair peuvent avoir une multitude d'utilisations, que ce soit dans le partage de fichier [21, 24, 23], la communication [26], les jeux vidéos [28], le calcul scientifique [25, 30, 20], le militaire [22], etc.

L'architecture pair à pair est faite telle qu'il n'y a pas de goulot d'étranglement, car nous passons d'un système où tout passait par un point unique à un système qui comporte un grand nombre d'entités qui peuvent toutes avoir le même rôle. L'utilisation de l'architecture pair à pair peut entraîner un grand nombre de communications. Elle nécessite des synchronisations des entités, de la gestion des ressources partagées et d'autres problèmes liés à la répartition des données. Il faut donc trouver des solutions à tous ces problèmes éventuels. Le pair à pair peut donc être plus adapté, dans certains cas, à des applications massivement multi joueur mais il faudra pouvoir garantir les mêmes propriétés que les systèmes client/serveur.

Les systèmes pair à pair sont plus difficiles à surveiller, les phénomènes de tricherie

sont donc plus difficiles à détecter, de même pour la sécurité. Nous avons pu voir qu'il existe trois types de tricherie : par Confidentialité, c'est à dire obtenir des informations non autorisées sur d'autres utilisateurs ; par Intégrité, si il y des modifications du monde, des lois physiques ou les lois du jeu non autorisées ; par Disponibilité, c'est le fait de provoquer des ralentissements ou des arrêts de partie du jeu (référence vers Challenges in P2P gaming). Il faut que le système soit aussi fiable sur le long terme et qu'il soit tolérant aux connexions et déconnexions (churn).

Les jeux vidéos sont des applications distribuées qui ne sont pas dites "critiques" (temps réel "mou"), le fait que le jeu ralentisse légèrement de manière très ponctuelle n'est pas gênant. Certaines propriétés des applications distribuées peuvent être retardées ou sautées ponctuellement. Les jeux vidéos ont des avantages qui font qu'il sera moins ardu de réaliser une distribution [4] :

- Les jeux vidéos tolèrent une consistance faible pour les différents états de l'application.
- Il peut être possible de prédire les écritures et les lectures grâce à l'ensemble des règles définies dans le jeu. Le but du stage sera l'amélioration de la prise en compte de la mobilité dans les MMOGs, dont Blue Banana [13] sera un point de départ possible.

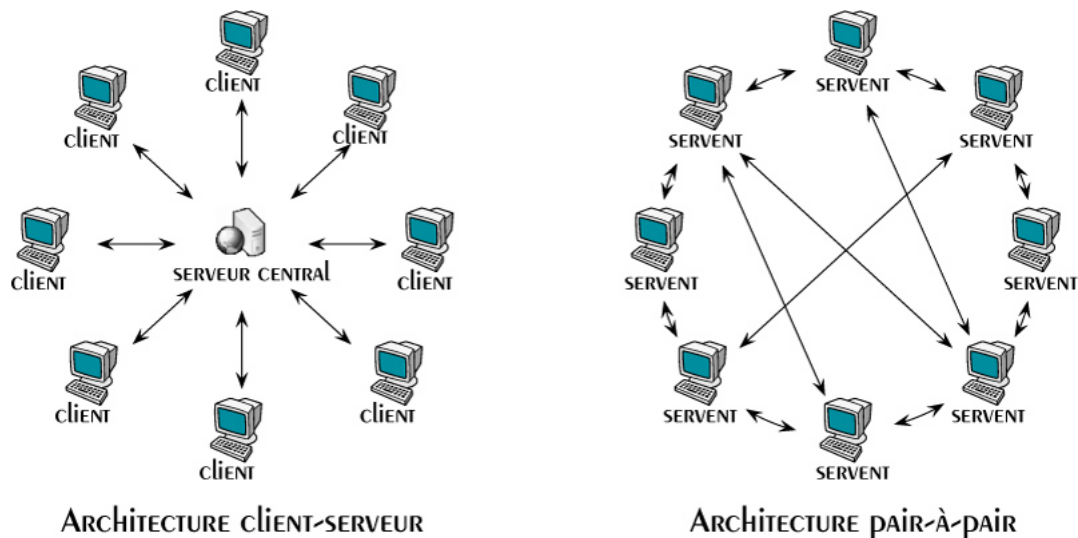


FIGURE 2 – Schéma des architectures pair à pair et client/serveur

4 Traces des utilisateurs dans les MMOGs

Nous allons expliquer les différentes techniques de collecte de traces et expliquer pourquoi ce travail de collecte est important pour améliorer les performances des solutions pair à pair pour les MMOGs.

4.1 Les différentes techniques de récupération de trace

4.1.1 Les objectifs et les techniques de collecte de trace

Dans la littérature, il y a plusieurs études des traces des utilisateurs dans des différents environnements virtuels [19, 10]. La plupart vont récupérer les traces des avatars sur des jeux vidéos tel que World Of Warcraft [29] et Second Life [27]. Nous avons pu voir qu'il peut y avoir des différences entre des MMOGs. Par exemple dans Second Life l'environnement est beaucoup plus interactif (possibilités plus étendues de modification de l'environnement) que dans World Of Warcraft, ce qui peut affecter des différences de résultats des solutions en fonction du jeu [15, 14].

Ces travaux vont permettre de bien comprendre les différents comportements des joueurs, ces travaux nous permettront de détecter les différents comportements en fonction des zones plus ou moins peuplées. Grâce à ces travaux, il sera possible de faire ressortir des modèles montrant le comportement des avatars dans le monde virtuel. Ces modèles permettent de mettre en place une distribution spatiale des avatars, qui sera plus cohérente que dans les recherches anciennes où les distributions étaient uniformes [11]. Différentes mesures vont apparaître comme : le nombre de joueurs, le nombre d'arrivées et de départs, le temps moyen d'une session, la distribution des joueurs, etc. L'étude des traces des joueurs permet aussi de détecter les tricheurs qui utilisent des bots.

Certaines techniques utilisent un bot qui est introduit dans le jeu et qui va récupérer des informations sur les autres joueurs. Le bot va récupérer des informations à intervalle régulier, nous pourrons ainsi analyser les déplacements et vérifier les modèles. Par exemple dans [15], une librairie open source *libsecondlife* a été développée pour mettre en place le bot dans Second Life. Pour vérifier que les données récupérées sont cohérentes, une technique de positionnement de sept bots dans une région ressort. Quatre bots statiques sont placés à chaque coin de la région, un autre statique va se placer au centre et deux autres vont bouger suivant un schéma défini. Nous pouvons alors enregistrer les positions des avatars se trouvant dans la région et nous aurons sept fois les informations sur les positions. Nous comparerons alors les données pour voir si des déviations existent entre les valeurs et si elles sont importantes.

4.1.2 Limitations de la collecte des traces

Des limitations existent pour la collecte des traces. Une des premières est que les mondes virtuels sont très souvent découpés en région ou en île or les bots que nous

introduisons ne peuvent pas traverser les régions et ils ne peuvent pas, par exemple, suivre des avatars. Dans [15], la possibilité de différencier un avatar qui va dans une autre région et un qui quitte le jeu n'est pas possible. Nous ne pouvons pas savoir ce que va faire l'avatar en dehors de la zone. Il y aussi un problème avec la détection des avatars qui se trouvent sur des objets et il n'y pas de prise en compte de la coordonnée z . Une autre limitation de la collecte des traces est qu'elle se fait en très grande majorité de façon manuelle, il est donc très long et fastidieux de récupérer un nombre de traces suffisant pour effectuer des bonnes analyses. La collecte de trace peut aussi avoir des problèmes de passage à l'échelle car les systèmes de collecte existants travaillent à une petite échelle. Par exemple dans Second Life, le monde étant découpé en îles indépendantes, l'addition des traces collectées sur chaque île pour en faire une étude globale, n'est pas cohérente.

4.2 Observations des traces

La collecte de toutes ces traces a permis de faire beaucoup d'observations sur les déplacements des avatars dans les mondes virtuels. Cela a aussi permis de faire valider ou invalider des modèles qui avaient été mis en place [15].

4.2.1 Hotspots

Une des observations qui est ressortie de ces études est l'existence de différentes zones dans le monde, une autre observation est que les mouvements des avatars étaient très différents en fonction de la zone où ils se trouvent. Deux types de zone peuvent se dégager : des zones très peuplées avec des avatars ayant des mouvements très aléatoires et lents, et des zones qui sont entre les zones peuplées, où les avatars se déplacent rapidement et suivant souvent une trajectoire rectiligne. Des phénomènes de déplacement en groupe ont aussi pu être repérés [16].

4.2.2 Waypoints

Comme nous pouvons le voir dans le schéma 3, il y a des "routes" entre les différents points de regroupement. Ces routes vont nous permettre de mieux anticiper les déplacements des avatars entre les *Hotspots*.

L'étude des traces a aussi permis de détecter les habitudes des joueurs, comme le fait qu'ils jouent à certaines heures plutôt que d'autres et avec des durées de connexion différentes en fonction de l'heure de la journée.

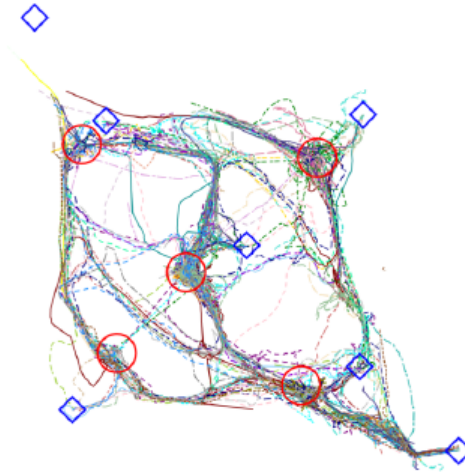


FIGURE 3 – Battle 980 movement paths

5 Systèmes P2P exploitables par les MMOGs distribués

5.1 Area Of Interest

Dans plusieurs articles, la notion de *Area Of Interest* [3, 4, 5] va apparaître ce qui montre l'utilité de ce concept. Nous pouvons déjà retrouver ce principe sous le nom de *Local Awareness* dans Sollipsis, l'idée principale de ce mécanisme est qu'une entité n'a pas besoin de connaître l'ensemble du monde virtuel à chaque instant. Alors nous allons mettre en place une zone dans laquelle l'entité sera tenue informée des différentes modifications qui seront faites sur les objets et les autres entités qui se trouvent dans la zone (voir schéma 4). Si une entité modifie sa représentation virtuelle, seulement les entités qui sont dans sa zone seront directement informées.

Dans Mercury [5], le concept d'Area Of Interest va pouvoir être utile avec le mécanisme de publish-subscribe qui est mis en place. Ce mécanisme de publish-subscribe permet l'abonnement et le désabonnement aux mises à jour d'un objet, et il permet donc d'envoyer un objet vers les nœuds qui sont abonnés.

Colyseus [4] est un travail postérieur à Mercury, les principes sont les mêmes avec l'utilisation de *range-queriable* DHT ou de DHTs pour stocker les informations. Les *range-queriable* DHTs vont s'organiser en un overlay circulaire où chaque nœud adjacent est responsable d'une suite continue de clés. Grâce à cela, il sera possible de prendre les coordonnées "x" comme clé et ainsi les performances seront bien meilleures qu'avec des DHTs normales (aléatoire). Les différents résultats réalisés montrent bien que les *range-queriable* DHTs utilisent moins de bande passante que les DHTs normales.

Dans Donnybrook [3], le concept d'Area Of Interest fait référence au travail réalisé

dans Colyseus. Comme chaque joueur envoie ses mises à jour à chaque joueur qui se trouve dans sa zone, il faut une limitation du nombre de joueurs dans une même zone. Donnybrook introduit la notion de *player's interest set*, un joueur va pouvoir se concentrer sur un nombre fixe de joueurs (cinq dans l'article), à la différence du nombre d'objet dans l'AOI qui peut varier. Un mécanisme d'abonnement sera mis en place pour s'échanger les informations entre les joueurs. Un mécanisme d'*intérêt estimé* est mis en place pour définir les joueurs qui feront partis de son *interest set*. Plusieurs critères sont pris en compte pour choisir les joueurs qui seront sélectionnés. Trois principales propriétés sont prises en compte : la proximité spatiale entre les joueurs, les objectifs des joueurs et les interactions récentes que les joueurs peuvent avoir eu entre les deux joueurs.

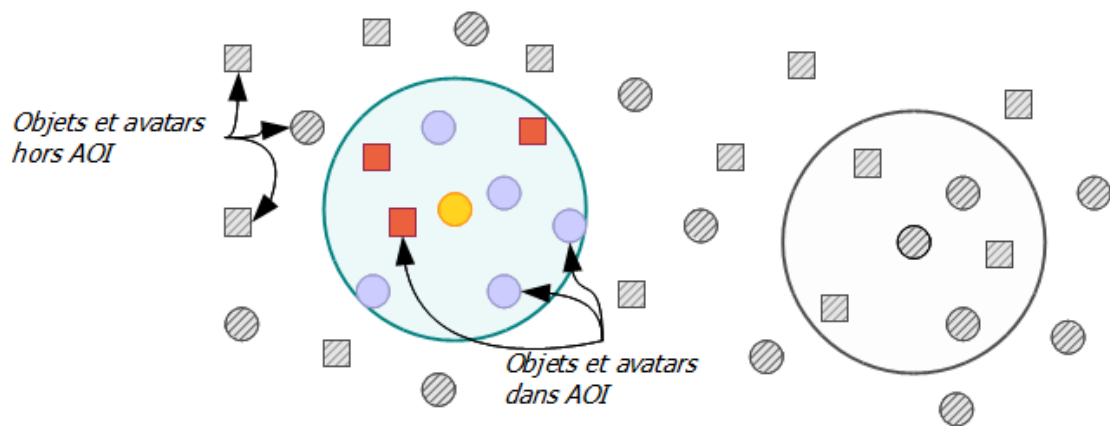


FIGURE 4 – Principe de l'Area Of Interest

5.2 Découpage de la carte

Pour plusieurs raisons, dont l'équilibrage de charge, le monde a été découpé en plusieurs zones. Différentes techniques pour le découpage ont été étudiées, l'une des plus courantes est d'utiliser le découpage grâce au découpage de Voronoi [8], c'est de cette technique que la triangulation de Delaunay s'inspire. Les découpages suivant ces

principes sont répandus car un découpage aléatoire ne prendrait pas en compte la densité des objets dans le monde qui peut être très variable entre les régions. Le principe est de découper la monde en zone en fonction de la distance entre les différents objets qui se trouvent dans l'environnement. Dans une région, on aura un objet qui sera entouré de ses voisins. La frontière entre la région de deux voisins se trouve au milieu d'une droite séparant les deux voisins. Comme il est possible de voir sur le schéma 5, les zones ont des formes et des tailles différentes. Nous avons ainsi des zones qui comprennent un seul nœud. La différence entre la triangulation de Delaunay et le découpage de Voronoi est que pour la première, les arrêtes seront les droites entre les sommets si ils sont voisins et dans le deuxième cas, les arrêtes seront formées grâce aux médiatrices des arrêtes de la triangulation de Delaunay.

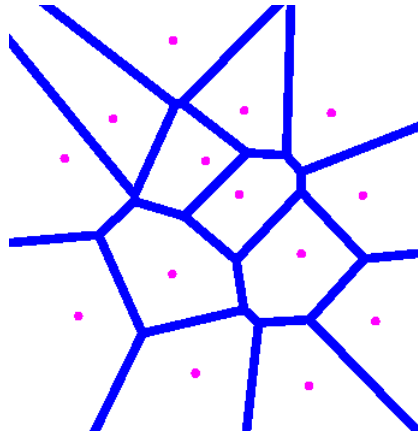


FIGURE 5 – Principe du découpage de Voronoi

5.3 Overlays

Plusieurs travaux sur la modification de l'overlay ont été fait [1, 17, 7], le but principal de ces travaux est de mettre en place un overlay qui puisse facilement s'adapter aux besoins des nœuds. Un overlay est un réseau informatique formant un graphe au sein duquel le voisinage de chaque nœud est déterminé selon un critère logique. Dans l'architecture pair à pair, il n'y a pas de connaissance globale, il n'y a qu'une connaissance locale (voisinage). Le but est de faire des liens entre les nœuds qui sont intelligents, nous verrons des overlays avec différents degrés de prise en compte de l'application.

5.3.1 Semantic overlay

Le but principal de cet overlay sémantique [1] est de créer des liens entre des nœuds qui s'intéressent aux mêmes documents, car la recherche de fichiers est devenue très

importante au moment de l'arrivée des logiciels de partage de fichiers tel que Napster, Gnutella, KaZaA, etc. L'approche consiste à constituer des groupes sémantiques avec les documents et de construire un overlay au dessus du réseau pour chaque groupe. Le problème est donc d'identifier et de constituer des groupes efficacement. Trois stratégies ont été mises en place :

- *LRU* : Stratégie basée sur les éléments les plus récemment utilisés
- *History* : On maintient les liens sémantiques vers la même classe de préférence, cette technique oblige le maintien d'un nœud *counter*. C'est une technique lourde en nombre de messages et en stockage, de plus il peut y avoir des problèmes avec les *counters*.
- *Popularity* : Création de liens entre les nœuds du même type, introduction de deux paramètres : Numrep (nombre de réponse positive pour obtenir le document) et Lastreply (date de la dernière réponse)

Les résultats montrent que Popularity est l'algorithme le plus adapté, cette stratégie a un bon compromis entre efficacité et complexité de mise en place.

5.3.2 Application-Malleable Overlay

Il y a plusieurs types d'overlay, certains ne prennent pas en compte l'application, d'autres le font seulement au démarrage, d'autres essayent de réagir aux événements de l'application. MOve [17] souhaite mettre en place un overlay malléable, c'est à dire que les communications de l'application distribuée vont influencer la structure de l'overlay. Les deux buts principaux sont d'optimiser les performances de l'overlay et de garder les propriétés de tolérance aux fautes et de passage à l'échelle. Les auteurs mettent en place deux types de liens :

- *Lien non-applicatif* : Ils maintiennent l'overlay global proche d'un graphe aléatoire avec un faible degré de clustering (regroupement).
- *Lien applicatif* : Ils permettent de créer un graphe fortement connecté, pour regrouper les nœuds. Chaque nœud d'un groupe crée aléatoirement des liens applicatifs vers des membres du groupe. Cela va permettre une rapide propagation des états lors des updates et des messages applicatifs multicast.

Cette solution offre différentes possibilités d'ajout de nœud, de détection de défaillance, de remplacement de lien, etc. *group-based* applications influencent l'overlay sous-jacent en remplaçant les liens inter nodes par des liens entre les applications. L'algorithme maintient une bonne connectivité.

5.3.3 Voronoi-based Overlay Network

VON veut exploiter la localité des intérêts des utilisateurs pour maintenir la topologie pair à pair avec un faible *overhead*. VON utilise un principe d'AOI dynamique, il utilise le découpage de Voronoi et chaque nœud est représenté par un site dans le diagramme. VON introduit trois procédures principales :

- *Join Procedure* : Le textitjoining node contacte le server pour avoir un ID unique, puis envoie une requête avec ses coordonnées à tous les nœuds existants. Création

de la liste des voisins, mis à jour chez les voisins, etc.

- *Move Procedure* : Lorsque un nœud bouge ses coordonnées sont mises à jour chez ses voisins (gestion si nœud est à la limite de la région, si il y a des nouveaux voisins, etc).
- *Leave Procedure* : Le nœud se déconnecte (qu'importe sa raison et la manière) et ses voisins vont se mettre à jour.

VON permet de gérer un grand nombre d'utilisateurs et de garder une topologie consistante. Il y a beaucoup de messages introduits par l'AOI et si la vitesse des utilisateurs est trop importante, on a des problèmes pour notifier les nouveaux voisins.

6 Un système P2P pour MMOG : Solipsis

Solipsis est le "socle" du travail Blue Banana, nous allons présenter les propriétés importantes qui le caractérisent.

6.1 Introduction sur Solipsis

Nous allons expliquer le fonctionnement global et les fonctionnalités importantes de Solipsis. Solipsis est fait pour accepter un nombre illimité d'utilisateurs et pour maintenir une cohérence suffisante. Il peut être accessible par n'importe quel ordinateur, il peut fonctionner sur des ordinateurs peu puissants et avec des connections internet faibles (56Kbs) ou sans fil. Une fois les nœuds connectés, ils peuvent échanger des données telles que de la vidéo, du son, le mouvement d'avatar ou toutes choses affectant la représentation du monde virtuel.

6.2 Propriétés de Solipsis

Le monde de Solipsis est un tore à deux dimensions, chaque entité détermine sa position dans le monde et elle en est responsable. Chaque utilisateur collecte les informations lui permettant de reconstituer son environnement virtuel local. Les connections entre les nœuds sont bidirectionnelles.

6.2.1 Propriétés

Solipsis doit permettre aux utilisateurs de se déplacer à travers le monde, il faut pour cela que les deux propriétés locales suivantes soient respectées.

- *Local Awareness* :

Une entité doit être connectée avec tous ses plus proches voisins. Il est important de noter qu'une entité a la possibilité de connaître des entités en dehors de son environnement virtuel local. En revanche, cette propriété impose que l'entité située à l'intérieur.

- *Global Connectivity* :

Une entité doit connaître toutes les entités se trouvant dans son "champs de vision". Elle doit pouvoir détecter l'arrivée ou le départ d'une entité de son "champs de vision". Cette propriété est basée sur la Géométrie Informatique, elle assure qu'une entité ne "tourne pas le dos" à une partie du monde. L'enveloppe convexe des entités est le plus petit polygone convexe formé par cet ensemble. Un mécanisme pour éviter qu'une partie du graphe soit isolée a aussi été mis en place.

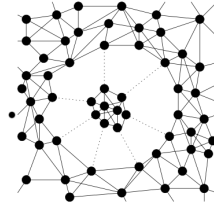


FIGURE 6 – Un composant connexe isolé et les connexions qui évitent la création d’une île

6.2.2 Maintien des propriétés

Solipsis a aussi des mécanismes de collaboration pour maintenir les propriétés précédentes, nous allons voir ces deux propriétés :

- *Spontaneous Collaboration for Local Awareness* :
Pour vérifier la propriété *Local Awareness*, il faut qu’une entité puisse connaître tous ses voisins à chaque instant. Pour faciliter cette connaissance du voisinage, et comme il y a un grand nombre de mouvements dans le monde virtuel, un système de collaboration entre les nœuds a été mis en place. Une entité pourra alors demander régulièrement à ses voisins si ils détectent une nouvelle entité mais cela implique un grand nombre de message inutile et une perte temporelle de consistance. Pour éviter ces problèmes, une entité a va prévenir une entité b si une entité c est en train de se diriger vers la zone de l’entité b .
- *Recursive Query-Response for Global Connectivity* :
Il faut prévoir une mécanisme pour le maintien d’une entité dans l’enveloppe convexe. Quand une entité e détecte deux entités consécutives, elle se lance immédiatement à la recherche d’une ou plusieurs entités dans le même secteur en envoyant des messages dans la zone.

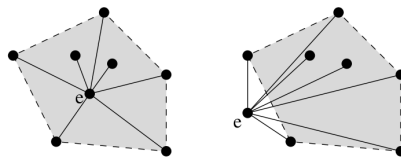


FIGURE 7 – Deux différentes enveloppes convexes des voisins de e . A gauche, e respecte la règle de *Global Connectivity* et à droite non.

7 Prise en compte de la mobilité : Blue Banana

Il nous a été possible de voir, dans les chapitres précédents, des mécanismes qui permettent faire évoluer le système en réaction à des événements. Solipsis ne pourrait pas bien fonctionner dans un monde avec des traces correspondants à la réalité, Blue Banana va donc mettre en place un mécanisme d'anticipation des mouvements des avatars pour mieux s'adapter.

Blue Banana présente une solution aux différents problèmes que peut rencontrer une architecture pair à pair dans les MMOGs. La mobilité des avatars implique de nombreux échanges de données à travers le réseau pair à pair. Comme les overlays de l'état de l'art n'anticipent pas cette mobilité, les données nécessaires ne seront pas chargées à temps, ce qui conduit à des défaillances transitoires au niveau applicatif. Blue Banana a été réalisé pour résoudre ce problème, il modélise et prédit les mouvements des avatars ce qui permet à l'overlay de s'adapter par anticipation aux besoins du jeu.

7.1 Solutions introduites

Blue Banana est implémenté au dessus de Solipsis qu'il a été possible d'étudier dans le chapitre 6. Plusieurs observations ont été faites et différentes optimisations en sont ressorties. Il nous a été possible d'observer plusieurs types de zone (dense ou non, cf. 4) et que les mécanismes d'adaptation sont trop tardifs pour être mis en place dans la réalité (le chargement des données sera trop lent).

7.1.1 Les états de l'avatar

Une des premières innovations qui a été introduite est la distinction de plusieurs états d'un avatar. Comme il a été possible de voir dans le chapitre sur la collecte de trace, un avatar se comporte différemment en fonction des zones du monde. Deux états ont donc été introduit :

- **T**(ravelling) : l'avatar se déplace rapidement sur la carte et il a une trajectoire droite.
- **E**(xploring) : l'avatar est en train d'explorer une zone, sa trajectoire est confuse et sa vitesse est lente.

Le changement d'état de l'avatar se fait en fonction de la vitesse de celui-ci, si la vitesse devient supérieure à une borne définie et que l'avatar est dans l'état E alors l'avatar passe en état T. Ce modèle pourrait être affiné par la suite en prenant en compte l'accélération ou l'historique des mouvements.

7.1.2 Anticipation des mouvements

Un autre mécanisme a été mis en place, il s'agit d'anticiper les mouvements d'un avatar, pour cela deux suppositions sont faites : seulement une prédiction courte est cohérente, et plus l'avatar se déplace rapidement, plus il y a de chance qu'il continue dans la même direction [13]. Comme nous pouvons voir sur la figure 8, en fonction du vecteur

de mouvement de l'avatar, le nœud B, si il est dans l'état **T**, va chercher des nœuds qui se trouvent sur la trajectoire probable de l'avatar, tant que son ensemble de voisins n'est pas plein. Le nœud B va envoyer un message aux voisins qui sont le plus près de lui par rapport au vecteur de mouvement. Un mécanisme pour évaluer si le nœud n'est pas trop près, et donc rapatrier des données ne servirait pas car le temps des communication serait supérieur au temps du déplacement de l'avatar. Un des risques est de rapatrier des nœuds qui seront inutiles si l'avatar va changer de direction ou d'état. L'amélioration de ce point fait parti des futures pistes pour améliorer l'algorithme d'anticipation des mouvements.

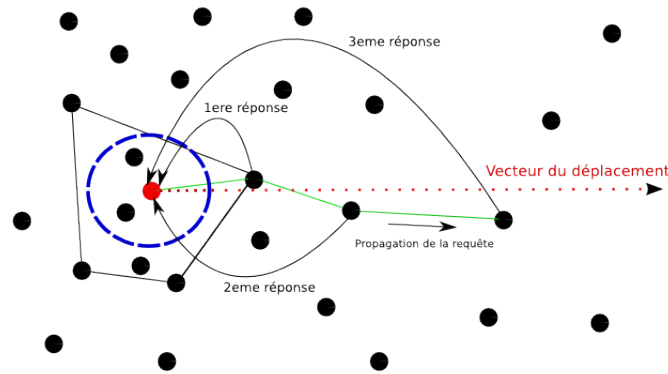


FIGURE 8 – Algorithme de propagation

7.2 Expérimentations et Résultats

Le travail a été testé sur le simulateur à évènement discret PeerSim [9], les expérimentations ont eu pour objectif de comparer Solipsis avec et sans Blue Banana.

7.2.1 Le simulateur PeerSim et la description des expérimentations

PeerSim est un simulateur de réseau pair à pair, qui a deux modes de fonctionnement : par cycles ou par évènements. C'est une API riche et modulaire qui est codée en Java, c'est une composante du projet BISON de l'université de Bologne (Italie). Ce simulateur permet de simuler un large nombre de machines et de tester différentes configurations du réseau. Le simulateur va faire des simplifications sur les couches réseau et les contraintes physiques (latence, pannes, ...). Chaque nœud est considéré comme un module qui va

échanger des messages avec les autres nœuds du système. La plupart des plateformes de simulation est basée sur le modèle à événements discrets. Il est possible de distinguer deux entités : les nœuds et les messages. Le temps va seulement évoluer à chaque nouvel événement sur un nœud.

Au départ de la simulation, une carte initiale des traces est introduite dans le simulateur, la carte provient d'une étude de La et Michiardi [12] dans Second Life. Ensuite, le simulateur va initialiser l'overlay de Solipsis et vérifier que les deux règles de Solipsis sont bien respectées sur chaque nœud, nous insérerons ensuite le reste des traces. Il faut aussi régler les différents paramètres du simulateur (nombre d'avatar, surface du monde, densité, accélération des avatars, vitesse de connection, etc). Plusieurs métriques sont mises en place pour évaluer les résultats :

- *Violation of Solipsis fundamental rules* : Regarde si les propriétés de *Global Connectivity* et de *Local Awareness* sont respectées.
- *Knowledge of nodes ahead of the movement* : Mesure pour les avatars qui se déplacent rapidement, le temps moyen pour qu'il connaisse un nœud qui sera sur sa trajectoire.
- *Exchanged messages count* : Mesure l'impact de Blue Banana sur le réseau, cela va compter le nombre de messages introduits par Blue Banana et Solipsis.

7.2.2 Les résultats

Les résultats les plus intéressants sont que Blue Banana diminue les transitions en échec de 55% à 20%, augmentent la connaissance des prochains nœuds de 270% et cela en créant un overhead de seulement 2%. Les résultats montrent que le mécanisme d'anticipation introduit par Blue Banana aide l'overlay de Solipsis à s'adapter à temps et à réduire significativement le nombre de violation des règles de Solipsis (de 55% ou 80% à 20%).

8 Conclusion

Dans ce document, nous avons étudié les différentes architectures disponibles pour les MMOGs, nous avons observé les limites et les avantages que peuvent avoir les architectures client/serveur et pair à pair. Ensuite grâce à l'étude des traces, nous avons pu dégager différents points permettant l'amélioration de la prise en compte de la mobilité dans les MMOGs. Nous avons vu différents mécanismes permettant d'améliorer la réactivité dans les applications pair à pair, et plus précisément dans les MMOGs, mais la plupart fonctionnait, au mieux, en réaction aux événements [17]. Blue Banana est un mécanisme d'anticipation des mouvements, ce qui permet de mieux faire évoluer le réseau. Nous essayerons dans le reste du stage d'améliorer l'anticipation des mouvements, en nous appuyant sur ce qui a été fait dans Blue Banana.

Les mécanismes d'anticipation des mouvements donnent des résultats satisfaisants mais il sera nécessaire de trouver des améliorations pour mieux anticiper les mouve-

ments en essayant de ne pas handicaper les performances de l'application. Les performances globales ont déjà été bien améliorées grâce aux mécanismes mis en place dans Blue Banana et cela sans trop affecter le réseau. Des améliorations sur les déplacements en groupe pourraient être une piste d'amélioration des performances. Un article, parlant des déplacements en groupe des fourmis, explique que leurs différentes observations ont servi dans l'informatique [6].

Références

- [1] Exploiting semantic proximity in peer-to-peer content searching. In *FTDCS '04 : Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems*, pages 238–243, Washington, DC, USA, 2004. IEEE Computer Society.
- [2] Reis Tiago Alves and Licinio Roque. Because players pay : The business model influence on mmog design. In Baba Akira, editor, *Situated Play : Proceedings of the 2007 Digital Games Research Association Conference*, pages 658–663, Tokyo, September 2007. The University of Tokyo.
- [3] Ashwin Bharambe, John R. Douceur, Jacob R. Lorch, Thomas Moscibroda, Jeffrey Pang, Srinivasan Seshan, and Xinyu Zhuang. Donnybrook : enabling large-scale, high-speed, peer-to-peer games. *SIGCOMM Comput. Commun. Rev.*, 38(4) :389–400, 2008.
- [4] Ashwin Bharambe, Jeffrey Pang, and Srinivasan Seshan. Colyseus : a distributed architecture for online multiplayer games. In *NSDI'06 : Proceedings of the 3rd conference on Networked Systems Design & Implementation*, pages 12–12, Berkeley, CA, USA, 2006. USENIX Association.
- [5] Ashwin R. Bharambe, Mukesh Agrawal, and Srinivasan Seshan. Mercury : supporting scalable multi-attribute range queries. *SIGCOMM Comput. Commun. Rev.*, 34(4) :353–366, 2004.
- [6] Eric Bonabeau Guy Theraulaz and Marco Dorigo. Intelligence collective des fourmis et des nouvelles techniques d'optimisation.
- [7] Shun-Yun Hu, Jui-Fa Chen, and Tsu-Han Chen. Von : a scalable peer-to-peer network for virtual environments. *IEEE Network*, 20(4) :22–31, July 2006.
- [8] Shun-Yun Hu and Guan-Ming Liao. Scalable peer-to-peer networked virtual environment. In *NetGames '04 : Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, pages 129–133, New York, NY, USA, 2004. ACM.
- [9] Márk Jelasity, Alberto Montresor, Gian Paolo Jesi, and Spyros Voulgaris. The Peersim simulator. <http://peersim.sourceforge.net/>.
- [10] Zhi-Qiang Jiang, Wei-Xing Zhou, and Qun-Zhao Tan. Online-offline activities and game-playing behaviors of avatars in a massive multiplayer online role-playing game. *EPL (Europhysics Letters)*, 88(4) :48007, 2009.
- [11] Björn Knutsson, Massively Multiplayer Games, Honghui Lu, Wei Xu, and Bryan Hopkins. Peer-to-peer support for massively multiplayer games, 2004.
- [12] Chi-Anh La and Pietro Michiardi. Characterizing user mobility in Second Life. In *SIGCOMM 2008, ACM Workshop on Online Social Networks, August 18-22, 2008, Seattle, USA*, August 2008.
- [13] Sergey Legtchenko, Sébastien Monnet, and Gaël Thomas. Blue Banana : resilience to avatar mobility in distributed MMOGs. Research Report RR-7149, INRIA, 2009.

- [14] Huiguang Liang, Ransi Nilaksha Silva, Wei Tsang Ooi, and Mehul Motani. Avatar mobility in user-created networked virtual worlds : measurements, analysis, and implications. *Multimedia Tools Appl.*, 45(1-3) :163–190, 2009.
- [15] Huiguang Liang, Ian Tay, Ming Feng Neo, Wei Tsang Ooi, and Mehul Motani. Avatar mobility in networked virtual environments : Measurements, analysis, and implications. *CoRR*, abs/0807.2328, 2008.
- [16] John L. Miller and Jon Crowcroft. Avatar movement in world of warcraft battlegrounds. In *Netgames 2009*. IEEE, November 2009.
- [17] Sebastien Monnet, Ramses Morales, Gabriel Antoniu, and Indranil Gupta. Move : Design of an application-malleable overlay. *Reliable Distributed Systems, IEEE Symposium on*, 0 :355–364, 2006.
- [18] Christoph Neumann, Nicolas Prigent, Matteo Varvello, and Kyoungwon Suh. Challenges in peer-to-peer gaming. *SIGCOMM Comput. Commun. Rev.*, 37(1) :79–82, 2007.
- [19] Daniel Pittman and Chris GauthierDickey. A measurement study of virtual populations in massively multiplayer online games. In *NetGames '07 : Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, pages 25–30, New York, NY, USA, 2007. ACM.
- [20] Chord. <http://pdos.csail.mit.edu/chord/>.
- [21] Gnutella. <http://www.gnutella.fr/>.
- [22] Jxta. <https://jxta.dev.java.net/>.
- [23] KaZaA. <http://www.kazaa.com/>.
- [24] Napster. www.napster.com/.
- [25] Pastry. <http://research.microsoft.com/en-us/um/people/antr/Pastry/>.
- [26] Skype. www.skype.com/.
- [27] Second Life. <http://secondlife.com/>.
- [28] Star Wars Galaxie. <http://www.starwarsgalaxie.com/>.
- [29] World of Warcraft. <http://www.worldofwarcraft.com/>.
- [30] Xtremweb. <http://www.xtremweb.net/>.