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

Les différentes idées d'amélioration

Xavier Joudiou

Encadré par: Sergey Legtchenko & Sébastien Monnet

17/06/10

Table des matières

1 Introduction **3**

2 Mise en place d'un cache pour les zones peuplées **4**

3 Mécanismes de connaissance des routes entre les Hotspots **5**

4 Amélioration du prefetch de Blue Banana **6**

5 Les mouvements de groupe **7**

6 Autres **8**

7 Conclusion **9**

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 résoudre 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 document, nous allons présenter différentes idées pour tenter d'améliorer la réactivité des réseaux pair à pair dans les MMOG.

1 Introduction

Ce document va récapituler les différentes solutions d'amélioration de la réactivité des réseaux pair à pair pour les MMOG. Il faudra tenir compte de la difficultés et du temps de mise en place de chaque solution, pour choisir la solution qui sera réalisable avant la fin du stage et qui puisse apporter des améliorations non négligeables. Plusieurs pistes sont envisagées dont celle menant aux mouvements de groupe, que l'on peut aussi retrouver de façon plus précise dans un rapport bibliographique qui lui est dédié.

2 Mise en place d'un cache pour les zones peuplées

La mise en place d'un cache pour les zones peuplées permettrait d'améliorer la réactivité dans un état (**W**) qui n'est pas encore améliorée. L'avantage de cette solution, même si l'issue n'est pas certaine, est de s'intéresser à une partie que l'on n'a pas encore fait évoluée.

Le but de cette solution serait de mettre en place un cache qui garderait un certain nombre de nœuds qui vient de sortir de la liste des voisins du nœud. Ainsi comme les mouvements de l'avatar sont désordonnés, il est possible qu'il retourne vers des nœuds qu'il vient de quitter. Sur la figure 1, il est possible de se faire une idée de cette solution.

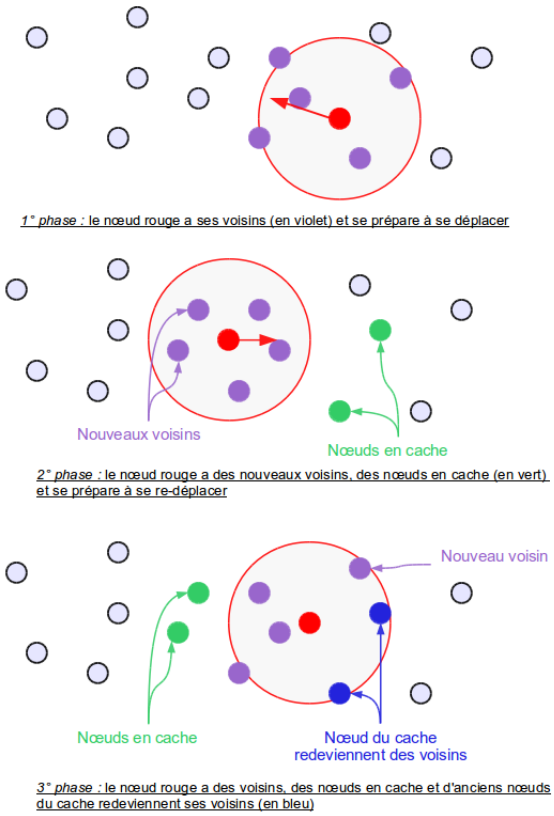

FIGURE 1 – Exemple de gain possible pour le prefetching

Sur la figure 1, un exemple de l'utilisation du cache est mis en avant. Nous pouvons voir trois phases durant lesquelles un nœud (rouge) va bouger et modifier ses voisins à chaque instant. Les nœuds, qui étaient voisins et qui ne le sont plus à l'instant suivant, sont placés dans le cache. Ainsi lorsque le nœud revient vers eux, ce dernier n'aura pas à refaire une recherche de voisins avec les deux nœuds verts, il aura juste à demander aux nœud verts si ils sont toujours au même endroit, etc.

Cette solution pourrait économiser des messages de découverte des voisins dans le cas de changements de direction fréquents. Mais l'efficacité de cette méthode n'est pas

sûre.

3 Mécanismes de connaissance des routes entre les Hotspots

Dans cette solution, nous voudrions permettre aux avatars de suivre des routes, pour le contournement d'un obstacle par exemple. Il faudrait modifier le modèle pour ajouter des obstacles dans l'environnement. Deux solutions apparaissent rapidement pour créer ses routes. La première solution serait de définir des chemins pour contourner les obstacles, et de conserver ces chemins dans l'environnement. La deuxième solution consisterait à mettre en place un mécanisme d'apprentissage des routes par les avatars. Les avatars pourraient apprendre les routes au fur et à mesure des passages, et laisser des indications pour les avatars suivants. Cette solution permettrait de simuler un comportement réel, et ainsi d'essayer d'améliorer cette situation.

Les modifications sur le modèle ne seront peut être pas très simples à mettre en place. Il faudrait aléatoirement, comme il a été fait pour les Hotspots, définir des zones où les avatars ne pourraient pas passer ou seraient ralentis. Sur la figure 2, nous pouvons voir des trajectoires permettant d'éviter l'obstacle. Il faut définir si ces trajectoires se font par apprentissage ou si elles sont données avec l'initialisation de la carte.

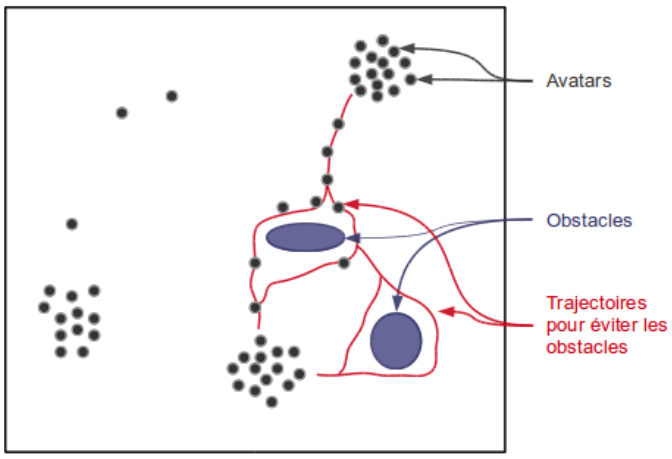

FIGURE 2 – Exemple de trajectoire d'évitement d'un obstacle

4 Amélioration du prefetch de Blue Banana

Une des solutions est d'essayer de continuer le travail déjà réalisé dans Blue Banana [3]. L'objectif serait de rapatrier plus finement les données. Pour le moment, nous regardons juste les nœuds qui sont dans notre champs de vision (un cône) et qui ne sont ni trop loin, ni trop près. Mais il serait intéressant d'avoir plus d'informations sur les nœuds, comme leur direction ou leur vitesse par exemple.

Actuellement un nœud, qui est dans l'état **T**(ravelling), va chercher des nœuds qui se trouvent sur la trajectoire probable de l'avatar, tant que son ensemble de voisins n'est pas plein. Ce mécanisme va donc rapatrier des données qui sont à bonne distance (pas trop près à cause des temps de communication). Un des risques est de rapatrier des nœuds qui sont inutiles si l'avatar, dont nous rapatrions les données, change de direction ou d'état. Le mécanisme existant ne va pas observer les différentes propriétés des nœuds (vitesse, direction, état, etc) et dans certains cas, il est possible qu'il rapatrie des nœuds qui viennent vers lui très rapidement et qui ne seront donc pas utiles.

L'idée est donc en plus de la distance avec le nœud de tester les différentes propriétés. Une solution serait peut être d'essayer de déterminer de façon simpliste les futures positions des nœuds et de tenir compte du couple vitesse/direction, pour mieux rapatrier les données. De plus, il faudrait chercher parmi des nœuds qui peuvent être hors du cône, mais sans trop chercher sinon trop de messages seraient émis.

Nous pouvons voir un exemple de l'avantage de cet ajout (voir figure 3), qui pourrait, par exemple, permettre l'ajout du nœud en rouge au lieu du vert.

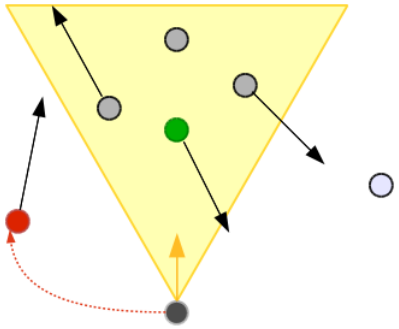

FIGURE 3 – Exemple de gain possible pour le prefetching

5 Les mouvements de groupe

Cette partie est décrite de façon plus précise dans le rapport bibliographique qui lui est consacrée. Il nous a été possible d'observer, à travers différentes études [2, 2, 1], que les joueurs se déplacent fréquemment en groupe. Ces déplacements se font la plupart du temps entre les joueurs d'un même guilde. Nous pouvons observer sur la figure 4 qu'en grande majorité les joueurs font partie de groupe, c'est pour cela que cette solution peut être intéressante.

Pour mettre en place cette solution, il faudra refaire un modèle de mobilité qui met en place des déplacements de groupe. Ce travail pourrait prendre plus de temps que les autres solutions car il nécessite un bon nombre de changement dans l'existant.

FIGURE 4 – Schéma récapitulatif des activités de groupe

L'utilisation des mouvements de groupe pourraient nous permettre de réorganiser le prefetching des données, et ne plus considérer les nœuds indépendamment mais comme formant un groupe. Ces groupes vont devoir avoir une organisation flexible et efficace. Les nœuds se trouvant en avant du groupe, selon la direction, pourrait être les seuls à rapatrier des données, ainsi nous économiserions des messages. Il faudrait ensuite transmettre les données vers les autres membres du groupe, plusieurs méthodes de diffusion peuvent être mises en place. La formation du groupe pourrait aussi permettre de mettre entre parenthèse la recherche de voisins (pour certains des nœuds en tout cas).

La figure 5 montre un exemple de ce à quoi pourrait ressembler la solution. Les nœuds, en gris foncé, vont rapatrier les données qui peuvent être intéressantes. Les autres nœuds du groupe n'auront pas rapatrier les données, mais il faudra ensuite diffuser ces données vers le reste du groupe.

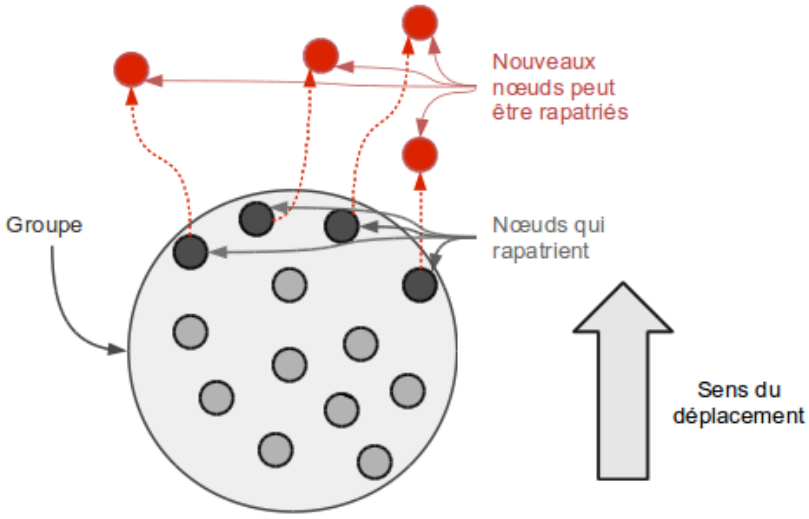

FIGURE 5 – Une piste pour les déplacements en groupe

6 Autres

Dans cette partie, nous allons faire le tour d'autres idées d'amélioration, mais sans rentrer dans les détails comme nous avons pu le faire pour les autres. Des nœuds pourraient partager des liens encore plus forts que ceux du voisinage. Si deux nœuds vont dans la même direction et qu'ils ne sont pas trop éloignés (pas obligatoirement parmi les voisins), alors un lien pourrait se créer pour échanger des données.

Une autre amélioration pourrait être d'insérer des nœuds qui seraient fixe dans l'environnement, ils pourraient servir de "relais" dans certaines zones. Ces nœuds pourraient avoir plusieurs utilisations, nous pouvons imaginer qu'il pourrait servir de référent dans sa zone, ce mécanisme reviendrai à mixer un peu les solutions client/serveur et pair à pair. (FLOU)

Possibilité de créer des liens entre des nœuds qui sont proches dans le réseau. Ainsi deux nœuds qui sont proches pourrait échanger des données rapidement et ces données pourrait servir au nœud ultérieurement ou il pourrait les faire partager à d'autres nœuds sur le réseau virtuel.

De la même façon que l'idée d'instaurer un cache, les nœuds pourrait se souvenir, en fonction de la zone géographique où ils se trouvent, d'anciens nœuds déjà rencontrés et ainsi tenter de communiquer avec eux prioritairement. (Trop Comme Cache)

7 Conclusion

Nom Solution	Tps Implem	Ajout princpal	Intérêt	Gains
Cache	Moyen	Hotspots	+	+ ?
Routes	Long	Entre Hotspots	+ (routes ajoutent réalisme)	+ ?
Prefetch	Moyen	Existant	+ Meilleur prefetch	? ?
Groupe	+ Long	Entre Hotspots	+ (groupe ajoutent réalisme)	+ ?

Just Ro It ! ! ! !

Références

[1] Nicolas Ducheneaut and Robert J. Moore. The social side of gaming : a study of interaction patterns in a massively multiplayer online game. In *CSCW '04 : Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 360–369, New York, NY, USA, 2004. ACM.

[2] Nicolas Ducheneaut, Nicholas Yee, Eric Nickell, and Robert J. Moore. "alone together ?" : exploring the social dynamics of massively multiplayer online games. In *CHI '06 : Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 407–416, New York, NY, USA, 2006. ACM.

[3] Sergey Legtchenko, Sébastien Monnet, and Gaël Thomas. Blue Banana : resilience to avatar mobility in distributed MMOGs. Research Report RR-7149, INRIA, 2009.