Stage de fin d'études
France Telecom Recherche et Développement

Jeux vidéos sur réseaux ad-hoc

Rapport de Stage final

Du 6 février au 4 août 2006

Auteur :
Pierre CHECA

Tuteur entreprise :
Sidi-Mohammed SENOUCI

Tuteur enseignant :
Vincent RICORDEL

Avec la collaboration de :
Khaled BOUSSETTA et Nadjib ACHIR,
maîtres de conférences de l'Université Paris XIII

Soutenance de stage le 13 juillet 2006 en vue de l'obtention du diplôme d'ingénieur
Ecole Polytechnique de L'Université de Nantes, SILR3 option Systèmes et Réseaux
Année scolaire 2005-2006

Table des matières

Remerciements

Tout d'abord, J'aimerais remercier Sidi-Mohammed Senouci mon responsable de stage, pour m'avoir accueilli et encadré de façon efficace et sympathique, me permettant ainsi de réaliser un stage formateur. A ce titre, je remercie aussi vivement Khaled Boussetta et Nadjib Achir, pour leur encadrement ainsi que leur apport technique et humain.
Je remercie également Vincent Ricordel, mon tuteur école, pour sa visite et son soutien.
Enfin, je remercie toute l'équipe R2A pour leur accueil et leur sympathie, et en particulier Yvon Gourhant le responsable, pour m'avoir accepté et accueilli.

Un grand merci aussi à Elise Morin la responsable des stages, ainsi qu'aux stagiaires et thésards m'ayant permis d'évoluer dans un environnement agréable où règne une bonne ambiance générale.

Introduction

Dans le cadre de ma dernière année d'études à l'Ecole Polytechnique de l'Université de Nantes, j'ai dû effectuer un stage d'un minimum de 5 mois en entreprise. En parallèle à cette dernière année, j'ai également suivi un master recherche. Le stage que j'ai réalisé chez France Télécom a donc fait double emploi : valider à la fois la dernière année d'ingénieur ainsi que le DEA, en mariant ces deux aspects d'ingéniérie et de recherche.

Lors de ce présent ouvrage, nous allons donc rapporter en détail le déroulement ce stage. Pour cela, nous allons tout d'abord évoquer le stage de façon générale, à travers une présentation de l'entreprise ainsi que du contexte technique qui entoure ce projet. Ensuite, nous entrerons dans le détail technique via un état de l'art global du domaine dans lequel s'inscrit le stage : les "Jeux vidéos sur réseaux ad-hoc".
A partir de là, nous présenterons les modélisations et projets de conception que nous avons émis, débouchant ainsi sur une phase de réalisation. Le rapport se conclura sur les résultats que nous avons obtenu, sur la gestion de projet qui a été mise en place, ainsi que sur un bilan final.

Chapitre 1

Le contexte et les objectifs du stage

Avant de passer sur les aspects techniques du stage, nous allons tout d'abord présenter le groupe France Telecom, tout comme sa filière R&D où j'ai pu évoluer pendant ce stage. Nous évoquerons également le contexte économique et technique du stage, expliquant ainsi pourquoi celui-ci a été proposé.

1.1 Présentation de l'entreprise

Voici les caractéristiques principales du groupe France Telecom, à travers quelques chiffres et une vue globale de ces activités.

1.1.1 Le Groupe France Telecom

France Telecom est l'opérateur de télécommunications historique en France. C'est une Société Anonyme (SA) qui a été créée en 1996, après la privatisation de l'entreprise. Avec un chiffre d'affaires 2005 de plus de 49 milliards d'euros, France Telecom est le plus grand groupe de Télécommunications européen et un des plus importants du monde.

Le Groupe emploie 203 000 personnes dans 220 pays à travers le monde, dont :

- 60,1% en France
- 16,5% en Pologne
- 11,9% au Royaume-Uni
- 2% en Espagne

40% de l'effectif travaille donc à l'international, pour servir plus de 147 millions de clients à travers le monde, dont 62 en France. Les différents métiers de l'entreprise sont :

- La téléphonie (mobile et fixe)
- L'accès Internet

- Les systèmes d'information et les réseaux
- Les services liés aux nouvelles technologies

La téléphonie France Telecom, en tant qu'opérateur historique, dispose de l'ensemble du réseau téléphonique câblé du territoire français. En France, le groupe dispose ainsi de plus de 26 millions de lignes fixes grand public, touchant un nombre de clients de près de 34 millions.

En ce qui concerne la téléphonie mobile, France Telecom est, à travers sa marque Orange, le plus grand opérateur français, avec près de 22 millions de clients. La téléphonie mobile est à l'heure actuelle le principal secteur d'activité du groupe, puisque parmis ces 147 millions de clients à travers le monde, 86 millions sont des clients des services mobiles.

L'Internet Le groupe est le premier opérateur européen en nombre de lignes ADSL, avec 8,1 millions de clients haut débit ADSL Grand Public en Europe. Il est 2ème opérateur mondial de l'ADSL derrière China Telecom, et devant les plus grands groupes américains et japonais. Le domaine de l'Internet, via les services Wanadoo, représentent ainsi près de 12 millions de clients à travers le monde.

La stratégie NExT (Nouvelle Experience des Télécoms) A l'heure actuelle, le monde des Télécoms est en plein bouleversement, avec l'avênement d'un grand nombre de nouvelles technologies, mais aussi celui de la convergence, notamment la convergence fixe / mobile. La stratégie Next s'inscrit donc dans la continuité de la logique du groupe qui est de s'assumer comme un opérateur intégré. Les objectifs sont multiples :

- Avoir un réseau et un système d'information intégré et mondial
- Une relation client harmonisée
- Un portail unique pour les clients, offrant un accès simplifié aux services
- Une culture commune, des valeurs partagées

Dans cette logique d'harmonisation et de convergence des services, le groupe France Telecom a donc tout intérêt à se regrouper à travers une marque unique. Pour cela, elle compte s'appuyer sur sa marque commerciale la plus forte et la plus représentée à l'étranger : la marque Orange. C'est ainsi qu'au premier juin 2006, le groupe France Telecom est devenu le groupe Orange, permettant de construire une identité forte très orientée vers l'international.

Pour affirmer encore plus sa dimension internationale et dans cette logique d'harmonisation, le groupe France Telecom est devenu le groupe Orange au 1er juin 2006. A travers cette nouvelle dynamique, le groupe compte donc renforcer sa suprématie européenne en terme d'innovation technologique.

1.1.2 France Telecom et sa branche R&D

En investissant pas moins de 1,5% de son chiffre d'affaires annuel dans la recherche, le groupe possède le plus grand centre de recherche européen dans son domaine des télécommunications. France Telecom compte ainsi plus de 3900 ingénieurs, scientifiques et chercheurs au sein de sa division R&D, cela sur trois continents : l'Europe, l'Asie et l'Amérique. Le groupe possède ainsi 16 sites de recherche à travers le monde :

En France, Orange dispose donc de 8 centres de recherche :

- Issy-les-Moulineaux (siège)
- Lannion
- Rennes
- Caen
- Grenoble
- Sofia-Antipolis
- Belfort
- La Turbie

Issy-les-Moulineaux et Lannion sont les deux plus gros centres. Chaque centre a sa spécialité, Grenoble est par exemple spécialisée dans ce qui concerne la télémédecine, et Caen dans la cybernetique. Issy-les-Moulineaux et Lannion quant à eux touchent à tous les aspects de la recherche du groupe, et sont réellement polyvalents.

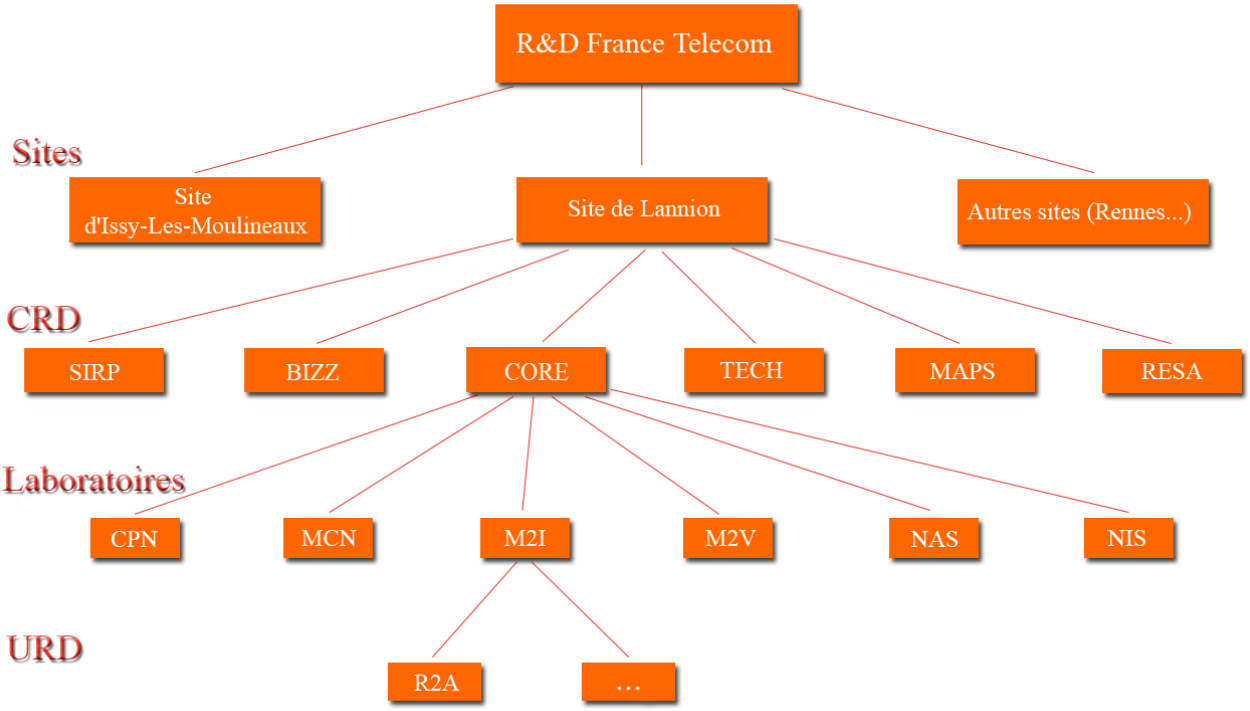
1.1.3 Le site de Lannion

France Telecom R&D est implantée à Lannion depuis 1963, au coeur de la technopole Anticipa, sous l'impulsion du ministre des PTT de l'époque : monsieur Pierre Marzin. En une quarantaine d'années, Lannion est devenue l'un des premiers lieux du développement des télécommunications dans le monde. Dans ce centre ont par exemple été inventés le minitel, la carte à puce, le GSM ou encore l'ATM. Située dans les Côtes d'Armor, proche de la côte de granit rose, Lannion est une ville de plus de dix-sept milles habitants. Avec l'implantation de nombreuses entreprises dans le Trégor, la région compte aujourd'hui plus de trois mille chercheurs et ingénieurs.

Le site de France Telecom Lannion est fort de près de 1600 employés, dont plus de 1200 scientifiques.

1.1.4 L'organisation de France Telecom R&D

France Telecom R&D a une organisation pyramidale assez simplement descriptible. Voici un organigramme de cette structure :



Tout d'abord, la R&D est divisée en **CRD** (Centres de R&D). Un CRD correspond à une grande spécialité, telle que le Coeur de Réseau ou encore les Réseaux d'Accès. A Lannion et à Issy-les-Moulineaux, tous les CRD sont représentés.[1]
Ensuite, chaque CRD est divisé en différents **laboratoires**. Le stage que j'ai effectué s'est par exemple déroulé dans le laboratoire **M2I** : **M**ultimédia networks for non-conversational fixed/mobile services : **I**mage, **I**nternet. Au sein d'un même CRD, les laboratoires peuvent être répartis sur différents sites (Lannion, Issy...).
Enfin, chaque laboratoire, comportant généralement entre 50 et 100 personnes, est divisé en **URD** (Unité de R&D). Chaque URD a en général une spécialisation très précise. Mon stage s'est déroulé dans l'URD **R2A**, spécialisée dans les réseaux ambiants et spontanés, dont les réseaux ad-hoc font typiquement partie. Enfin, tout comme les laboratoires, les URD d'un même laboratoire peuvent être répartis sur plusieurs sites. Par exemple, dans mon laboratoire, 3 des 6 URD sont à Lannion, et les 3 autres sont à Issy-les-Moulineaux.

Ainsi, mon stage s'est déroulé :

- dans le CRD **CORE**, les réseaux ad-hoc représentant un coeur de réseau en eux-mêmes.
- Dans le laboratoire **M2I**, concernant tout ce qui touche aux réseaux et au multimédia (hors services de voix et de visio)
- Dans l'URD **R2A**, concernant les réseaux spontanés dont font partie les réseaux ad-hoc

[1] Se rendre à l'annexe B pour savoir à quoi correspondent les sigles employés ici pour les CRD et les laboratoires

1.1.5 L'unité de recherche R2A

Comme nous l'avons vu, l'unité de recherche R2A traîte de tout ce qui touche aux réseaux ambiants et spontanés. L'équipe est composée d'une vingtaine de personnes, dont 8 thésards, plusieurs maîtres de conférences, plusieurs ingénieurs, et de 5 stagiaires pendant ce semestre ; le responsable est Yvon Gourhant, maître de conférences. Cette équipe dispose également de plusieurs chefs de projet, dont Sidi-Mohammed Senouci, mon responsable.

Sidi-Mohammed Senouci est ainsi responsable du PRP (Projet de Recherche Pluridisciplinaire) Spontex, projet développé par France Telecom traitant des réseaux spontanés. Le projet spontex fait intervenir différentes entités et personnes, dont des universitaires. C'est donc dans le cadre de ce projet que mon stage s'est déroulé.

Ce stage est également encadré par Khaled Boussetta et Nadjib Achir, maîtres de conférences de l'Université de Paris XIII, spécialisés dans le domaine des réseaux. Ceux-ci sont à l'origine de ce sujet de recherche et travaillent donc en collaboration avec nous dans le cadre de ce stage.

1.2 Le contexte économique et technique

1.2.1 Le contexte économique

Le marché du jeu vidéo est en pleine expansion depuis une quinzaine d'années, et ne s'est jamais aussi bien porté qu'à l'heure actuelle. En 2005, Le chiffre d'affaires a progressé de 5,3% par rapport à 2004, atteignant ainsi le chiffre de 19 milliards de dollars. Cette progression n'est pas prête de s'inverser, car les prévisionnistes voient le marché augmenter d'au moins 50% d'ici à 2008-2010.

Cette progression fulgurante s'accompagne d'un accès à l'Internet haut débit qui se généralise dans tous les foyers occidentaux, particulièrement en France, et fait que le jeu vidéo voit autant de nouvelles opportunités s'ouvrir à lui, via le jeu dit "On Line". A l'heure actuelle, les jeux vidéos multijoueurs pour PC permettent quasiment tous de jouer sur Internet.

En parallèle à cet essor, une autre technologie est apparue et s'est rapidement imposée comme un standard incontournable des réseaux locaux : la norme IEEE 802.11, mieux connue sous le nom de Wifi. Avec cette apparition, c'est une nouvelle manière de jouer aux jeux vidéos multijoueurs qui s'offre au grand public. Premier pas vers cette révolution annoncée : la sortie en 2005 des consoles Nintendo DS et Playstation PSP, des consoles portables proposant un mode multijoueurs sur réseaux wifi en mode ad-hoc.

1.2.2 Le jeu vidéo sur réseaux ad-hoc

Pour ces parties multijoueurs, les consoles portables utilisent actuellement un mode ad-hoc monosaut (n'impliquant donc pas de routage). Il semble tout à fait naturel d'utiliser ce type de réseau pour les consoles portables, qui par définition s'utilisent de façon spontanée entre amis, entre écoliers dans la cours de récréation ou dans les transports en commun. Le mode ad-hoc répond ainsi parfaitement à cette demande qui devrait exploser dans les années à venir. En effet, sans le moindre câble ni installation particulière, il est possible pour les joueurs de se

connecter directement l'un à l'autre en sans fil rapidement et spontanément.

En parallèle de ce constat, nous avons pu remarquer que très peu d'études sont parues sur ce sujet, qui pourtant semble être extrêmement prometteur.

1.2.3 Les objectifs du stage

Face aux promesses qu'offrent les réseaux ad-hoc, nous savons que ceux-ci en sont encore au stade expérimental, et que beaucoup reste à faire dans le domaine, notamment en terme de qualité de service. Ainsi à l'heure actuelle, les jeux en mode ad-hoc se réalisent généralement à deux joueurs près l'un de l'autre, mais guère plus. Ce que nous comptons apporter au cours de ce stage, c'est de permettre à un nombre relativement élevé de joueurs de se connecter à une telle partie, dans la cours de récréation ou autre, assez éloignés ou pas, cela en utilisant notamment des algorithmes de routages pour un mode ad-hoc dit "multi-sauts". Pour cela, il nous faudra apporter des solutions au niveau du réseau sans fil et de la qualité de service pour permettre une jouabilité optimale du jeu en réseau. Ce travail s'annonce être assez conséquent, et sera ainsi vraisemblablement poursuivi en thèse. Pour le stage, nous souhaitons au minimum comprendre les intéractions qui peuvent exister entre réseaux ad-hoc et jouabilité des jeux vidéos. En particulier, un des objectifs est de comprendre quels impacts ont les différents algorithmes de routages ad-hoc sur le jeu vidéo. Il sera également important de savoir quelles sont les limites actuelles des jeux vidéos sur réseaux ad-hoc, notamment en terme de nombre de noeuds et de mobilité. A partir de là, il serait bien de pouvoir émettre des pistes de solutions pour repousser les limites détectées, qui seraient éventuellement explorées en détail au cours d'une thèse.

Ce sujet aurait pu s'étendre à une problématique plus large qui aurait été : "Flux temps-réel sur réseaux ad-hoc". Mais nous pensons que le jeu vidéo est un flux à part, et que ses besoins sont différents que les autres flux multimédias classiques, puisqu'aucune information de son, de vidéo ou d'image transite sur le réseau. Ce sujet est particulier et nous comptons apporter une réponse spécifique qui pourrait être implémentée dans le futur dans les prochaines générations de consoles portables. Voyons maintenant en détail comment nous comptons réaliser nos prochaines études, et ce que nous comptons obtenir.

Nous en avons fini avec la présentation générale du stage et de l'entreprise, entrons maintenant dans le vif du sujet avec une première partie dite de bibliographie, qui nous permettra de mieux appréhender les domaines des jeux-vidéos et des réseaux ad-hoc.

Chapitre 2

Bibliographie

Ce stage étant à la fois un stage d'ingéniérie et de recherche, une grande partie bibliographie a été effectuée au début du stage. Cette aspect état de l'art nous a été indispensable pour comprendre tous les mécanismes mis en jeu dans les deux aspects de ce stage, à la fois au niveau des jeux vidéos et au niveau des réseaux ad-hoc. Avant d'aborder le travail réalisé en pratique, nous allons donc explorer en détail ces domaines techniques.

2.1 Les jeux vidéos multijoueurs et le réseau

2.1.1 Introduction

Dans cette partie, nous allons évoquer les jeux vidéos et leurs intéractions avec les réseaux en général. En effet, un assez grand nombre d'études sont parues sur le sujet, car le jeu multijoueurs n'est pas un sujet nouveau et ses besoins en temps-réel impliquent certaines contraintes au niveau du réseau. Dans un premier temps nous évoquerons les différents types de jeux vidéos disponibles sur le marché avant d'entrer dans le vif du sujet.

2.1.2 Les différents types de jeu

Avant de voir les intéractions entre réseau et jeu vidéo, il est important de connaître quelques principes sur les jeux vidéos. Il faut tout d'abord savoir qu'à l'heure actuelle, tous les jeux vidéos multijoueurs fonctionnent sur un mode Client / Serveur. Ceci principalement car la centralisation permet un contrôle total des joueurs qui jouent, de l'état du réseau, et permet aussi de garantir une cohérence du jeu. [12] [13] [15]
Ensuite, toutes les études réalisées sur les jeux vidéos montrent que le critère réseau le plus important pour la jouabilité des jeux est le temps de latence. Il est crucial d'avoir un temps de latence faible pour assurer la qualité d'une partie en temps-réel, nous le verrons en détail dans les chapitres qui suivent. Enfin, il est important de faire une classification des jeux vidéos, car tous les jeux n'ont pas du tout les mêmes besoins en terme de qualité de réseau.

Voici donc une liste des principaux types de jeux vidéos [26] :

Jeux d'action

- **Beat Them All** : le joueur se bat contre tous. *Exemple* : le jeu Double Dragon
- **Combat** : combat un contre un. *Exemple* : Street Fighter
- **Inflitration**. *Exemple* : Splinter Cell
- **Jeux de plateformes**. *Exemples* : les célèbres Super Mario et Sonic

Jeux d'aventures

- **Classique** : le héros vit une aventure et résoud des énigmes. *Exemple* : le jeu Les Chevaliers de Baphomet
- **Jeux de rôle** (RPG : Role Playing Game). *Exemples* : Final Fantazy, Diablo, Baldur's gate
- **Survival Horror**. Le joueur est plongé dans un univers très sombre et effrayant. *Exemples* : Resident Evil, Silent Hill

Jeux de tir

- **Les FPS** (First Person Shooter). Le but est simplement de tuer les autres. Les jeux phares sont Counter Strike, Quake, Doom,Half Life, Unreal Tournament
- **Shoot them up** : le joueur est en général dans un vaisseau, et doit tirer et anéantir les autres vaisseaux. *Exemple* : Space Invaders

Jeux de réflexion

- **Classiques** : Jeux de cartes, échecs, casses-têtes, ou encore le célèbre Tetris
- **Gestion** : il s'agit de construire et de gérer sa ville, sa maison ou sa colonie. *Exemples* : Sim City, Les Sims ou encore Caesar III
- **Stratégie** : aussi appelés **RTS** (Real Time Strategy). Les joueurs doit généralement planifier en construisant sa base, son armée, le tout en y mêlant l'aspect économique. Le but est d'anéantir la base des autres avec son armée. *Exemples* : Warcraft III, Age of Empires

Jeux de sport

- **Classiques** : Jeux de sport comme le foot, le basket ou encore le hockey sur glace. *Exemples* : Pro Evolution Soccer, Tony Hawks pro skater
- **Motorisés** : Jeux de voitures. *Exemples* : Gran Turismo, Mario Kart

Divers

- **Les MMORPGs** (jeux massivement multijoueurs sur Internet). Ces jeux sont des mondes virtuels dans lesquels des centaines de joueurs peuvent interagir entre eux, combattre, commercer, s'aider, se parler. Il s'agit de communauté où chaque individu à un rôle à jouer (le magicien, le guerrier, l'elfe...). *Exemples* : World of Warcraft, Lineage II
- **Les jeux de simulation** (d'avion par exemple, comme Flight Simulator)
- **Les jeux musicaux**. Il s'agit de suivre le rythme et la musique (Donkey Konga)
- **Les inclassables**, comme le célèbre Pacman ou Pong

De tous ces types de jeu, on comprend qu'un jeu d'échec ou de puzzle n'a pas les mêmes besoins en temps réel qu'un jeu de tir. De plus, il y a des types de jeux qui n'ont pas de mode multijoueurs, c'est le cas par exemple des jeux de plateforme comme Super Mario, ou encore les jeux d'aventures où il faut résoudre des énigmes.
Parmis les jeux multijoueurs, différentes études ont été menées pour connaitre les intéractions entre ces jeux et les réseaux qu'ils utilisent. Les jeux multijoueurs les plus populaires, et ayant ainsi fait l'objet de bon nombre d'études, sont les suivants :

- Les **FPS** (First Person Shooter)
- Les **RTS** (Real Time Strategy)

- Les **MMORPG** (Massive Multiplayer Online Role Playng Game)
- Les **jeux de sport**, et principalement les **jeux de voitures**.

Ces quatre types de jeux sont ceux les plus largement joués sur Internet, et de manière générale en multijoueurs. Par la suite, nous nous référerons donc principalement à ces types de jeu pour nos études. Entrons désormais dans le vif du sujet en évoquant l'impact du réseau sur la qualité du jeu vidéo.

2.1.3 Les effets des caractéristiques réseau sur le jeu vidéo

Comme nous avons pu l'évoquer, le réseau et ses caractéristiques n'ont pas le même impact suivant le type de jeu instancié. Voici donc comment les dégradations du réseau agissent sur la jouabilité selon le type de jeu vidéo utilisé.

Les jeux de stratégie

Dans les RTS, il y a différentes phases de jeux : les déplacements de joueurs / la découverte de la carte, le combat et la construction de sa base [1]. Un réseau dégradé n'a pas le même impact sur ces différentes phases de jeu : c'est le déplacement qui patira le plus des mauvaises conditions, en impliquant un déplacement moins rapide du joueur. Pour la phase de construction, les bâtiments vont également se construire moins vite, mais dans un temps négligeable par rapport au temps total de la construction (de l'ordre d'une seconde pour un bâtiment qui en met 90 pour se construire). Enfin pour le combat, un réseau dégradé ne changera généralement pas l'issue du combat : le joueur avec l'armée la plus puissante gagnera. Dans ces jeux, les coups portés à l'ennemi sont généralement des sorts, donc il n'est pas nécessaire d'avoir une grande précision pour le toucher.

De ces constats, le jeu de stratégie n'a pas besoin d'un réseau de très grande qualité, et malgré son nom (stratégie en temps réel), ce type de jeu n'a pas un besoin réellement temps réel. Des temps de latence de l'ordre de la seconde sembleraient être encore convenables pour une jouabilité correcte. Le jeu de stratégie a donc des besoins en temps réel similaires à du browsing sur Internet. En y réfléchissant ce résultat est rassurant et logique, car il signifie que la stratégie à long terme et l'intelligence de jeu prime sur la rapidité à réagir à un évènement précis pour gagner. Ce qui est précisément la vocation d'un jeu de stratégie.

Les MMORPGs

Les MMORPGs (Massive Multiplayer Online Role Playing Games) sont des jeux de rôles à grande échelle. Les principales phases de jeu sont les déplacements et les combats [3]. La particularité qu'ont ces jeux est que près d'un millier de joueurs peuvent être connectés en même temps sur un seul serveur, jouant les uns avec les autres et les uns contre les autres. De plus, au moment des pics d'affluence, les joueurs lancent des "rushs", pour conquérir un château par exemple ; à ce moment là des combats à 200 contre 200 joueurs peuvent éclater. On comprend alors que le réseau est mis à rude épreuve.
La chance qu'ont les responsables des serveurs est que pour ces jeux, là encore comme pour les jeux de stratégie, le temps de latence n'est pas crucial pour avoir une bonne jouabilité. En effet, les combats ne demandent généralement pas une grande précision car on touche souvent l'adversaire avec des sorts ; la problématique est identique aux RTS, car là encore la stratégie à long terme prime sur la réaction à un évènement isolé.

Les Jeux de course

Lorsque nous évoquions les résultats sur les jeux de stratégie, nous indiquions que c'est la phase de déplacement qui souffrait le plus des dégradations du réseau, mais que dans ces jeux le déplacement n'était pas important. Or dans un jeu de course, là au contraire le déplacement est primordial, le but étant d'être le plus rapide à franchir la ligne d'arrivée. Les jeux de course ont donc des besoin réellement temps réel, avec un temps de latence faible [6].

Les FPS

Les FPS (First Person Shooter) sont certainement les jeux les plus joués sur Internet, un succès notamment du au celebrissime Counter Strike. Le principe est simple : vous avez des armes, et vous devez tuer les autres. Ce "jeu de massacre" peut se dérouler seul ou en équipe, auquel cas les équipes rivales doivent s'entretuer. Tout comme les jeux de course, les besoins en temps réel sont très importants, car il s'agit de tuer les ennemis avec des armes, et il faut être précis. Prenons un exemple. Un joueur 1 indique sa position à un joueur 2, lui disant "Je suis au point de coordonnées (20,20). Simplement dans des conditions de latence élevée, le temps que le message n'arrive à 2... Le joueur 1 sera au point (30,30), et donc si le joueur 2 décide de le tuer, il tirera au point (20,20), où ne sera plus le joueur 1. Pour palier à ce problème, une technique a été développée, il s'agit du `"Dead Reckoning"` [11]. Celle-ci permet d'extrapoler la position des joueurs en pariant que celui-ci ne changera pas de trajectoire dans les centaines de milli-secondes qui suivent.

Malgré le Dead Reckoning, les jeux de type FPS ont donc de gros besoins en temps de latence [5], c'est ce que nous pourrons vérifier ultérieurement avec le jeu `Quake III Arena`.

Après ce survol des différents types de jeux multijoueurs les plus en vogue, voyons maintenant synthétiquement les différents paramètres réseau, et l'impact que chacun a sur la qualité et la jouabilité du jeu vidéo.

Conclusions

Comme nous venons de le voir, l'impact de la qualité du réseau sur le jeu dépend du type de jeu auquel on joue. Cependant, les différentes caractéristiques du réseau n'influent également pas de la même manière sur la jouabilité. Comme nous avons déjà pu l'évoquer, le paramètre prédominant quel que soit le jeu est le temps de latence : un mauvais temps de latence (supérieur à 100ms pour les FPS et supérieur à 500 à 1000 ms pour les RTS) implique nécessairement une dégradation de la jouabilité, même si la limite d'acceptabilité varie selon le jeu. [1] [2] [3] [5] [6]

En ce qui concerne les autres paramètres tels que la gigue et le taux de perte, ils interviennent moins dans la dégradation du jeu. [2] [5] [6] Des gigues assez importantes de l'ordre de 150ms ainsi que de taux de perte de 5% voire même 10% ne semblent pas affecter la jouabilité du jeu.

Enfin, évoquons deux paramètres qui ne sont pas des paramètres réseaux : la résolution et le nombre d'images par seconde. La résolution peut impacter le réseau en augmentant le nombre d'informations à transmettre, et le nombre d'images par seconde (FPS) peut quant à lui chuter sur un réseau de mauvaise qualité. Ces deux paramètres sont importants pour la jouabilité, particulièrement le FPS. En effet, le FPS peut faire chuter les scores et rendre le jeu haché, réduisant ainsi la jouabilité. La résolution quant à elle a un effet sur la perception visuelle du

jeu, mais pas sur la jouabilité en tant que telle (les joueurs obtiennent des scores analogues en petite ou en grande résolution) [4].

De tout cela, on déduit que les paramètres importants à prendre en compte sont la latence et le FPS, et dans une moindre mesure le taux de perte et la gigue. Maintenant que nous avons pu voir quels étaient les impacts du réseau sur le jeu vidéo, voyons maintenant comment le jeu vidéo impacte le réseau, et ainsi quel type de trafic il génère.

2.1.4 Le trafic réseau généré par les jeux vidéos

Dans la problématique inverse, nous sommes en droit de nous demander : "Comment un jeu vidéo va-t-il se comporter sur un réseau local ? Quel conséquence y'a-t-il pour un réseau d'héberger une partie multijoueurs ?" Il faut tout d'abord savoir que les jeux envoient généralement des paquets de petite taille à intervalle de temps constant, le tout à débit relativement faible [5]. Il est donc possible de jouer aux jeux multijoueurs en ayant une connexion Internet à bas débit, à partir du moment où ces connexions n'apportent pas trop de latence. La petite taille des paquets fait que si un de ces paquets est perdu, cela n'a pas beaucoup d'importance sur le jeu, d'où la faible importance du critère "taux de perte".

De plus, comme nous l'avons déjà évoqué, les jeux vidéos fonctionnent sur un mode Client / Serveur. Cela signifie que tout le trafic est concentré sur ces serveurs, et que par conséquent il est nécessaire d'avoir une grande bande passante et un temps de réponse faible vers ceux-ci. Cela pose d'ailleurs problème pour les jeux de type MMORPG, car ces jeux nécessitent des centaines de joueurs sur un seul serveur, impliquant un coût énorme d'investissement et d'entretien de ceux-ci. Généralement ces très gros serveurs permettent de répondre dans un délai de 200ms, ce qui est acceptable pour un jeu MMORPG.

Le trafic qui s'écoule entre client et serveur n'est pas plus symétrique que l'architecture qu'ils utilisent [7]. En effet, le trafic descendant est généralement bien plus important que le trafic montant, de même pour la bande passante ; bande passante qui, de plus, varie en fonction du nombre de joueurs connectés.

Tous ces résultats nous permettent d'avoir un ordre d'idée sur les résultats à attendre lors de nos futures experimentations, et nous verrons que pour le jeu Quake III Arena toutes ces affirmations seront confirmées, ceci avec l'aide d'un simple sniffeur de réseau.

Pour terminer cette partie sur les intéractions entre réseau et jeu vidéo, voyons le cas particulier du jeu vidéo sur réseau ad-hoc.

2.1.5 Le jeu vidéo sur réseaux ad-hoc

A l'heure actuelle, il y a extrêmement peu d'études réalisées sur le domaine des jeux vidéos sur réseaux ad-hoc. Les études disponibles à l'heure actuelle sont basées sur du ad-hoc en monosaut, il n'y a pas de notion de routage. Parmis ces études, la plupart s'intéressent d'ailleurs aux consoles Nintendo DS et Sony PSP, offrant un mode multijoueurs sur réseau ad-hoc monosaut.

Le jeu vidéo sur consoles portables

En introduction, nous évoquions l'avênement de ces nouvelles consoles portables Nintendo DS et Sony PSP. L'intérêt de ces consoles pour notre étude est qu'elles utilisent la norme

IEEE802.11b, de plus en mode ad-hoc pour le multijoueur en local.
Les jeux sur ces consoles portables ont des caractéristiques réseaux assez différentes des jeux actuellement disponibles sur consoles ou PC, puisqu'ils doivent s'adapter à la sans filité ainsi qu'au plus faibles capacités des consoles. Les questions qui peuvent se poser sont donc :

- Qu'est-ce que la notion de turbulence dans les jeux en réseau sur consoles portables ?
- Ces turbulences sont-elles équivalentes d'une console à l'autre ?
- Ces turbulences sont-elles équivalentes d'un jeu à l'autre ?
- Ces turbulences réseaux sur consoles portables sont-elles équivalentes à celles d'un d'un jeu sur PC ?
- Enfin, est-ce que ce trafic de jeu interfère réellement sur les autres applications "classiques" du WLAN qui utilisent le même canal ?

Tout d'abord, la notion de "turbulence" est tout simplement le trafic réseau induit par les jeux vidéos [9]. Ce trafic est assez caractéristique, car comme nous l'avons vu les jeux vidéos envoient fréquemment des informations, et se sont généralement de petits paquets (pour les jeux PC classiquement on aura un envoi de paquet de 100 octets toutes les 100ms). Ces turbulences se découpent en grandes phases que sont **le paramétrage du jeu**, **la synchronisation**, **la partie** et **les transitions**.

Ensuite, les caractéristiques réseaux de la Nintendo DS et de la sony PSP sont assez différentes : on ne peut donc pas déduire un modèle global pour toutes les consoles. En effet, la Nintendo DS n'utilise que 1 à 2 Mbps de débit maximum (pour le 802.11b, rappelons que le débit théorique max est de 11Mbps), alors que la PSP peut aller jusqu'à utiliser toute la bande passante. Enfin, les jeux de PSP utilisent souvent le broadcast (même lorsque le jeu ne se joue qu'à 2 joueurs), alors que les jeux de DS utilisent un adressage direct.

De plus, le trafic n'est pas le même entre chaque jeu. Les jeux de DS ont des comportements assez similaires, alors que ceux de PSP génèrent des trafics très différents les uns des autres.

Ces turbulences induites par les consoles portables sont similaires à celles des jeux sur PC, mais sur console portable on envoie les informations plus fréquemment (de l'ordre d'un paquet toutes les 10ms), impliquant un débit requis supérieur aux jeux PC.

Enfin, les turbulences des jeux sur consoles portables induites sur le canal de transmission Wifi dépendent de la phase de jeu dans laquelle nous sommes. La phase de synchronisation peut faire considérablement chuter la capacité du canal (dans l'étude, en lancant le jeu Super Mario, la communication entre deux PC qui utilisaient le canal est passée d'un débit de 6Mbps à un débit de 1Mbps). En revanche, les autres phases semblent ne pas avoir d'impact réel sur le canal de transmission.

A ces 5 points on peut ajouter que la Nintendo DS est bien plus robuste aux coupures réseaux que la PSP, car elle est beaucoup plus capable de fournir aux joueurs la possibilité de continuer leur partie, cela même si les conditions d'émission sont difficiles. Enfin, le fait que les jeux de DS n'utilisent jamais le broadcast (à l'inverse de PSP), permet de ne pas encombrer le réseau inutilement et de "polluer" le reste du trafic qui s'écoule sur le WLAN.

Pour nos études à venir, nous devrons mener des études assez similaires, mais en utilisant non plus du ad-hoc monosaut comme ces consoles, mais du multisaut. Lors de ces études, nous devrons également avoir à définir la qualité et la jouabilité du jeu. Le principal problème est : "Comment définir des critères objectifs et scientifiques pour définir la qualité d'un jeu qui comprend une bonne part de subjectivité ?". C'est à cette question que nous allons tenter de répondre dans la partie suivante.

Le jeu vidéo sur réseaux ad-hoc multisauts

Autant le sujet des jeux vidéos sur Internet ont été maintes fois abordé tout comme des articles sont sortis concernant le jeu vidéo sur réseaux ad-hoc monosaut, autant le jeu vidéo sur réseau ad-hoc multisauts en est vraiment au stade de recherche. Il y a ainsi très peu de billets traitant du sujet. Néanmoins, le peu d'études qui ont été réalisées indiquent qu'au delà de 3 sauts, le réseau ad-hoc est incapable de satisfaire les besoins en temps-réel des jeux vidéos [25]. Bon nombre d'améliorations ont été apportées pour rendre cela possible, en introduisant notamment des techniques relevant de la qualité de service. La QOS sur réseaux ad-hoc est également au stade expérimental, et elle est à l'étude au sein du consortium IEEE802.11e.

Pour conclure ce chapitre traitant des jeux vidéos et de leurs intéractions avec le réseau, voyons désormais comment il est possible d'évaluer la jouabilité d'un jeu vidéo, chose qui nous sera utile pour nos expérimentations.

2.1.6 Les différentes méthodes employées pour évaluer la jouabilité d'un jeu vidéo

Une des principales difficultés de notre projet est que l'on doit étudier l'impact du réseau sur la qualité du jeu. Le problème est le suivant : comment faire pour quantifier cette notion de qualité perçue ? La notion subjective de qualité implique qu'il est difficile de mettre en place une démarche très rigoureuse pour les études à mener. Il est cependant possible de limiter ce paramètre par des techniques astucieuses. Voyons tout d'abord les types d'évaluations réalisés dans les différentes études.

Les différents types d'évaluation

a) Le score

Lorsque la jouabilité baisse, cela se ressent sur les performances du joueur, et donc sur son score. Une technique utilisée est donc de faire jouer différents joueurs sur plusieurs parties, et de noter leurs scores. Petit à petit, on dégrade le réseau, et on note les scores obtenus.

- **Avantage** : On prend en compte le critère le plus proche de la notion de jouabilité : le score. Ce critère est important à prendre en compte, puisque le but des joueurs est bien avant tout d'avoir le score le plus élevé.

- **Inconvénient** : Ce critère est très difficilement mesurable objectivement et rigoureusement. D'une partie à l'autre, les joueurs peuvent avoir des scores très différents, sans pour que cela soit du à de quelconques problèmes de réseau.

b) Le questionnaire sur Internet

Dans certaines études, on a demandé à des joueurs sur des forums d'estimer quels étaient leurs besoins en terme de qualité de réseau, notamment en ce qui concerne la bande passante et la latence, pour avoir une bonne jouabilité.

- **Avantage** : Possibilité d'avoir un grand nombre de réponses, et prise en compte de l'avis des premier concernés : les joueurs.

- **Inconvénient** : Une fois de plus, un critère très subjectif, ne permettant pas de se rendre compte rigoureusement de la qualité du jeu.

c) La qualité perçue par le joueur lors de la partie

Une autre méthode consiste à faire jouer des joueurs en dégradant le réseau, et à leur demander de dire à chaque fois ce qu'ils pensent de la jouabilité de la partie qu'ils effectuent via une note.

- **Avantage** : La jouabilité c'est avant tout la jouabilité perçue par le joueur lui-même, donc ici on capte cette jouabilité par l'intermédiaire de notes.

- **Inconvénient** : Même si cette méthode semble plus fiable et adoptant une démarche plus digne d'un scientifique, elle introduit toujours beaucoup de subjectivité et pas de critère terre à terre permettant de définir clairement la qualité du jeu.

d) Le calcul du FPS

La jouabilité c'est aussi le nombre d'images par seconde affichées à l'écran ; plus il y en a, théoriquement plus la qualité s'accroît.

- **Avantage** : Le nombre de FPS est un critère mesurable et totalement objectif, ayant une proche relation avec la jouabilité.

- **Inconvénient** : Un nombre faible d'images par seconde traduit nécessairement une mauvaise qualité visuelle du jeu. Mais inversement, un nombre élevé d'images par seconde n'implique pas forcément un jeu de qualité. Prenons un exemple. Dans une partie, un joueur subit de graves problèmes réseau ; son personnage se déplace de façon très saccadée. Pendant ce temps, en haut à gauche de l'écran, une écriture dit en clignotant : "attention, vous êtes presque déconnectés". Le personnage s'affichera peut-être à la vitesse de 3 images par seconde, mais l'écriture, elle, s'affichera à la vitesse de 90 images par seconde. De ce fait, le nombre de FPS mesuré sera... De 90 images par seconde, alors que le jeu sera hâché.

  De plus, le FPS traduit plus la qualité de la vidéo plutôt que la jouabilité, de façon analogue à la résolution. Ce n'est donc pas vraiment le meilleur critère à prendre en compte, même s'il peut être intéressant.

e) Le découpage du jeu en phases

Comme nous avons pu le voir, les perturbations du réseau ont un impact différent suivant la phase de jeu (déplacement, combat, ou encore construction de la base). Il est ainsi possible de mesurer les perturbations sur chacun des secteurs de jeu.

- **Avantage** : Méthode précise, qui prend en compte la vraie nature des perturbations, à savoir qu'elle n'ont pas le même effet sur tous les secteurs de jeu.

- **Inconvénient** : Il n'est pas toujours facile de faire des mesures des dégradations sur chacune des différentes phases. Pour le déplacement cela est envisageable simplement, par contre pour le tir il faudrait prendre en compte le score, et on se retrouve avec les mêmes inconvénients qu'indiqués précédemment à ce sujet.

De toutes ces techniques évoquées, certaines semblent être réellement peu scientifiques, car elles remettent en cause un point important : la reproductibilité des expériences.

Analyse et proposition d'amélioration

Le caractère reproductible des expériences

Le premier gros repproche que l'on pourrait faire à ces expériences est que celles-ci sont dans la majorité des cas non reproductibles et c'est un énorme défaut, puisque le caractère reproductible est la base de toute parution d'articles scientifiques. Ces expériences ne sont pas reproductibles tout simplement parce que l'on utilise des joueurs humains pour les expériences, et que d'une partie à l'autre les joueurs ne vont pas jouer de la même manière. Cela est d'autant plus vrai que lorsque le réseau se dégrade, l'intelligence humaine fait que les joueurs vont s'adapter à ces conditions difficiles et jouer différemment, ce qui fait que les résultats sont totalement faussés. Une solution pour remedier à ce problème serait d'utiliser des joueurs contrôlés par l'ordinateur (des bots), et de plus que ces bots réalisent toujours les mêmes actions. Cela n'est pas toujours réalisable dans la pratique.

Proposition

Notre proposition consiste à s'imicer complètement dans le jeu vidéo en lui-même, à travers son code source. Cela nous est rendu possible par le fait que le jeu Quake III Arena possède un code source libre, et ainsi aisément modifiable. Ainsi, nous irons chercher directement dans le code source les informations dont nous avons besoin, comme par exemple le gel des images ou encore les distorsions des actions des joueurs ; tout cela sera développé en détail dans la section dédiée au jeu Quake 3.

Cette partie traîtant des intéractions entre le jeu vidéo et le réseau est terminée. Poursuivons maintenant cet état de l'art en nous penchant sur le second aspect de notre sujet : les réseaux ad-hoc.

2.2 Les réseaux Wifi en mode ad'hoc

2.2.1 Introduction

Nous avons évoqué à de maintes reprises le terme "réseaux ad-hoc" jusqu'à présent sans réellement expliciter le terme. Un réseau ad-hoc est un réseau spontané sans fil dans lequel aucune structure prédéfinie n'existe. Il n'y a donc pas de points d'accès, et chacun communique directement avec ses pairs sur une liaison Wifi. Les expériences effectuées au chapitre 2.3 concernent un jeu sur réseau ad-hoc dit "monosaut", c'est à dire que l'on essaie d'entrer en contact directement avec le destinataire, et si on y arrive pas on jette les paquets. Dans un réseau ad-hoc dit "multisauts", si un noeud A n'arrive pas à joindre directement son destinataire B, il enverra le paquet à une station tierce et le paquet ira de noeud en noeud jusqu'à atteindre sa cible. Pour cela, il faut donc introduire des protocoles de routage dans cette structure qui n'est pas définie, ce qui n'est pas sans poser de problèmes bien évidemment.

Les algorithmes de routage ad'hoc Pour résoudre ce problème de routage, le groupe MANET a été fondé par l'IETF en 1995. En 10 ans, le groupe MANET n'a pas standardisé un protocole unique car très vite deux approches se sont présentées avec des avantages et des inconvénients qui n'ont pas permis de les départager.

Des débats du groupe MANET sont nées deux catégories de protocoles. Il y a les algorithmes proactifs et les algorithmes réactifs.

2.2.2 Les algorithmes de routage ad-hoc proactifs

Les algorithmes proactifs construisent les tables de routage avant que les demandes de trames n'apparaissent sur le réseau. A tout moment, un noeud connaît la topologie du réseau. Ici, sera présenté les protocoles OLSR, TBPRF et DSDV.

OLSR : Optimized Link State Routing Protocol

Ce protocole est une adaptation du protocole d'état de lien. Il réduit la taille des messages de contrôle et minimise l'inondation du trafic de contrôle. Il se base sur le nombre de sauts pour fonctionner, ie le nombre de noeuds à traverser pour atteindre un autre noeud.

Ce protocole échange régulièrement les tables de routage sur le réseau.

Pour ce faire, il utilise les relais multipoints (MPR). Chaque noeud du réseau élit ses MPR parmi ses voisins à un saut avec un lien symétrique. Ceci permet à un noeud d'accéder à tous ses voisins à deux sauts.

Les noeuds MPR annoncent régulièrement leur rôle de MPR à leur voisinage. Les MPR permettent la mise en place des routes sur tout le réseau, car ce sont les seuls à pouvoir retransmettre les paquets sur le réseau. L'avantage de choisir les MPR de cette façon permet d'éviter la retransmission des paquets sur les liens unidirectionnels.

Le fonctionnement L'idée de ce protocole est de **minimiser la diffusion** dans le réseau. Pour cela, chaque noeud doit **élire le minimum de multipoints réseaux**, qui sont les seuls à pouvoir retransmettre dans le réseau. La suite présente donc cette élection.

Tout noeud émet des paquets Hello présenté ci après, indiquant la liste de ces voisins. Ainsi, si un noeud récupère tous les paquets Hello de ses voisins, il peut connaître tous ses voisins à deux sauts.

A partir de cette topologie, le but est qu'un noeud puisse accéder à tous ses voisins à deux sauts. Pour illustrer ceci, la figure 2.1 montre une situation. Nous allons nous intéresser au cas du noeud du milieu. Celui-ci est à deux sauts de tous les noeuds, il doit donc pouvoir tous les atteindre par l'intermédiaire des MPR.

Les MPR possibles sont les points d'accès A, B, C et D, puisqu'ils sont reliés directement au noeud central. Si on élit C et D, on se rend compte que les noeuds liés seulement à A et B ne seront pas accessibles. On comprend donc qu'il faut élire A et B. Si seuls A et B peuvent retransmettre les paquets, il n'y a pas de points d'accès inaccessibles directement ou par le biais de la retransmission de A ou de B.

Le but est donc atteint, tous les noeuds sont accessibles par ses deux MPR. Il est assez facile de résoudre visuellement ce problème dans de nombreux cas, mais ce problème est NP-complet. Pour le résoudre, les noeuds utilisent une heuristique.

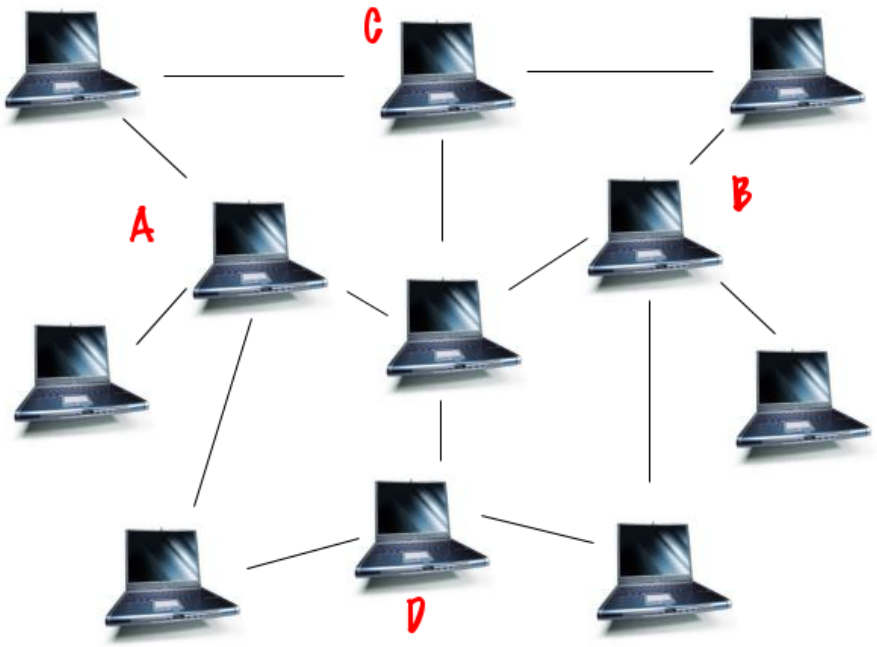


FIG. 2.1 – L'élection des multipoints relais

Une fois l'élection faite, chaque noeud informe son choix à son ou ses MPR. Les MPR savent donc qu'ils sont les seuls à retransmettre les messages. Chaque noeud ne connaît qu'une topologie locale du réseau. Il connait la liste de tous les noeuds accessibles sur le réseau et stocke son MPR associé vers lequel il doit envoyer le paquet pour que celui-ci arrive à destination.

TBRPF : Topology Broadcast Based on Reverse-Path Forwarding

Le protocole TBRPF a pour but, comme le protocole OLSR, de réduire l'utilisation de la bande passante. A la différence d'OLSR, dans le protocole TBRPF, tous les mobiles vont connaître la **topologie complète du réseau**. Ceci coûte plus cher en bande passante, mais l'intérêt, c'est la possibilité de traiter des **chemins multiples** ou de faire de la **qualité de service**.

Chaque mobile possède son propre arbre de plus court chemin représentant la topologie du réseau. Seules les modifications de la table de routage sont envoyées et non la table entière comme dans OLSR [32].

Ce protocole utilise l'algorithme de Dijkstra pour déterminer le chemin le plus court. L'intérêt ici, est que l'on peut choisir la métrique que l'on veut pour le choix des routes (qualité du lien, débit,...). Il est découpé en 2 modules indépendants : le module de découverte des voisins et le module de routage.[36]

Pour maintenir les arbres de plus court chemin de chaque noeud valide, un noeud, détectant une modification de topologie, envoie un message de mise à jour aux autres noeuds. Pour éviter le bouclage de ce genre d'informations dans le réseau, le mécanisme illustré par la figure 2.2 est mis en place.

On considère que A veut émettre une modification sur le réseau. Son arbre de plus court chemin propre est représenté par les flèches du schéma. Quand A émet une modification sur le réseau, seul les noeuds internes de l'arbre retransmettent cette modification. Pour ce faire, chaque noeud, à la réception du message de A calcule l'arbre associé à A et décide s'il est une

feuille ou un noeud de l'arbre. Dans le cas ou c'est un noeud et seulement dans ce cas, il émet le paquet. Dans notre exemple, seuls les noeuds C et G retransmettent aux noeuds E, F et H.

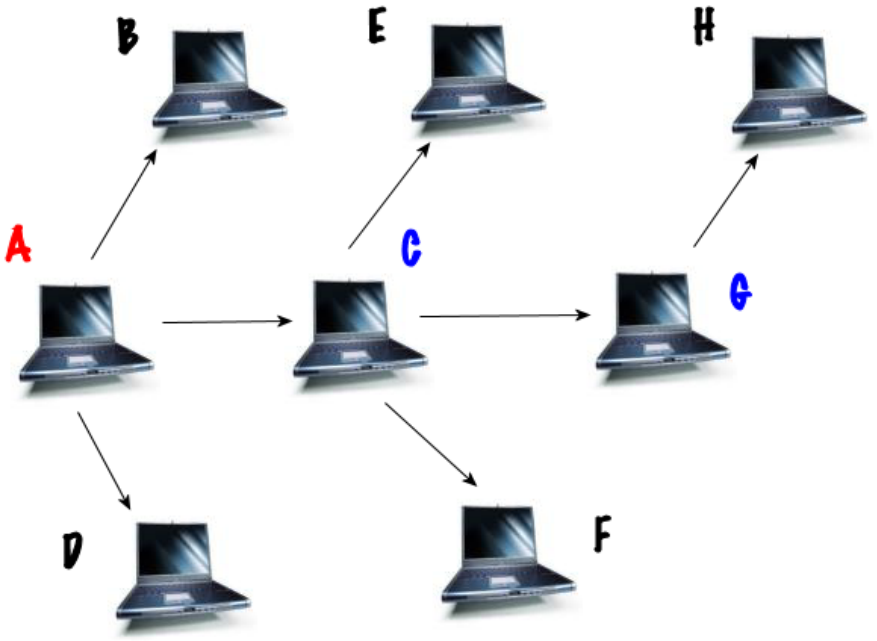


FIG. 2.2 – Les arbres dans TBRPF

DSDV : Destination-Sequenced Distance Vector

Ce protocole est inspiré du protocole RIP (Routing Information Protocol) sur les réseaux filaires. Il repose sur un vecteur de distance. Un enregistrement de table de routage contient donc les éléments suivants :

- L'adresse IP de la destination.
- Le nombre de sauts entre la destination et la source.
- Le noeud voisin sur lequel il faut envoyer le paquet pour accéder à la destination.

Voici les défauts de DSDV :

- Trop de signalisation
- le protocole n'est pas adapté à la mobilité des noeuds, ie les routes ne sont pas trouvées rapidement.

Les auteurs ont décidé de ne pas continuer à implémenter ce protocole. Ils ont préféré travailler sur le protocole AODV, qui est un protocole réactif et qui va être présenté dans la suite [42].

Conclusion

Par cette étude, on se rend compte que seuls deux algorithmes se sont imposés parmi les trois proposés ici. Ce sont les algorithmes OLSR et TBRPF. Nous allons maintenant envisager un autre type d'algorithme de routage.

2.2.3 Les algorithmes de routage ad-hoc réactifs

Pendant que les algorithmes proactifs présentés ci-avant, se développaient, une autre approche faisait son apparition : les algorithmes réactifs. Ceux-ci construisent leurs tables de

routage lorsqu'un noeud le demande. Les noeuds ne connaissent pas la topologie du réseau à tout moment, il définissent le chemin à utiliser dans le cas où un noeud veut émettre.

Les principes communs aux protocoles réactifs

Ces protocoles se composent de deux modules, un module de découvertes des routes et un module de maintenance des routes. Les paragraphes suivants présentent ces deux mécanismes.

Le mécanisme Route Discovery Le but de ce module est de trouver les nouvelles routes du réseau.

La suite présente un cas d'utilisation qui montre bien comment le protocole fonctionne. Un noeud veut émettre vers un destinataire inconnu. Il envoie une requête RREQ en broadcast sur le réseau.

Chaque noeud intermédiaire qui reçoit ce paquet, concatène son adresse et renvoie ce paquet à ses voisins.

Quand le destinataire reçoit le paquet, il envoie un paquet RREP avec la route.

Dans cette configuration, les noeuds intermédiaires enregistrent la route et répondent si une machine désire accéder à la même destination ou une destination qui se trouve sur le chemin.

Quand l'émetteur reçoit le paquet RREP, il émet un paquet de type SrcR contenant les données à échanger.

Le mécanisme Route Maintenance La connectivité dans les réseaux ad'hoc n'étant pas assurée, après l'envoi de chaque paquet, on s'assure que le chemin pour atteindre sa destination, est toujours valide. Pour cela, on effectue les trois tests décrit ci-après.

Premier test, on demande à la couche liaison de données si les liens nécessaires sont encore actifs.

Si ce n'est pas le cas, DSR écoute sur le réseau pour savoir si le noeud reçoit des paquets du noeud suivant dans la route. Si le noeud trouve des paquets de ce noeud, cela signifie que le lien est valide.

Si le deux tests échouent, alors DSR réémet le paquet en demandant un acquittement (AckReq). Le noeud suivant devra répondre par un acquittement. En cas d'échec, le noeud concerné met à jour sa table de routes et envoie un paquet de type Route Error (RErr) à la source. Celle-ci pourra choisir une nouvelle route ou lancer la procédure de Route Discovery.

AODV : Ad hoc On Demand Distance Vector

AODV est un protocole qui est récent et qui évolue encore. Ainsi sont venues se greffer des améliorations petit à petit. La suite va d'abord présenter le principe de ce protocole et ensuite, sera envisagé l'amélioration de ce protocole.

Il faut préciser que le protocole AODV est décrit par la RFC 3561 disponible à l'adresse : http ://www.ietf.org/rfc/rfc3561.txt.
Le statut de cette RFC est expérimental.

Les fonctions de base AODV est un protocole de routage réactif **unicast** et **multicast**. Il évite les boucles dans les réseaux et est **auto-démarrant**.

Pour le multicast, ce protocole stocke des arbres pour les routes des différents membres de chaque groupe multicast [39].

Le protocole AODV implémente les types de messages suivants :

- Route Request (RREQ)
- Route Reply (RREP)

- Route Error (RERR)
- Route Reply Acknoledgement (RREP-ACK)

Les 3 premiers paquets ont été présentés ci-avant, mais le paquet RREP-ACK n'a pas été présenté. Il sert à régler le problème des liens asymétriques ou unidirectionnels que nous évoquerons plus loin.

Ces messages sont délivrés sur le port UDP, port numéro 654. Les retransmissions ne sont pas spécifiées, sauf dans le cas de retransmissions globales. AODV permet d'ajouter des extensions aux messages. Par exemple, on peut spécifier, pour un noeud le temps entre 2 messages Hello. Il y a aussi des extensions pour la qualité de service et la découverte de services.

Une table de routage dans AODV est constitué des champs suivants :

- **adresse IP destination**
- **Numéro de séquence courant** : permet de savoir si on tient compte d'un message qui arrive ; on ne tient compte des messages qui possèdent un numéro supérieur à celui-ci
- **Le noeud suivant pour la transmission d'un paquet à cette destination**
- **Temps durant lequel cette entrée de la table est valide** : ce temps est initialisé au début et est mis à jour à chaque mise à jour de l'entrée concernée.

Il n'existe qu'une seule route par destination dans AODV.

Dès que le temps de validité d'une route est dépassé, la route est supprimée de la table. Le noeud considère que le lien est brisé. Dans certains cas, c'est une perte, si le lien est toujours actif. Mais, dans d'autres cas, ça permet de ne pas essayer de transmettre des messages à un noeud qui a bougé et n'est plus atteignable. L'efficacité de ce mécanisme dépend de la mobilité des noeuds.

Au niveau de l'implémentation des protocoles à la demande, il se pose souvent des problèmes. En effet, la plupart des applications quittent quand elle n'arrive pas à joindre un lien. Il faut donc réécrire le code de la pile IP et de toutes les applications pour pouvoir mettre en attente ces applications.

DSR : Dynamic Source Routing

Ce protocole de routage utilise des connexions **unicast**. Un noeud peut stocker **différentes routes pour la même destination**.

De plus, DSR utilise le **source routing**. Dans les faits, quand un noeud cherche à envoyer un paquet, il commence par envoyer un paquet RREQ pour connaître le chemin à emprunter. Il inscrit dans l'entête du paquet son adresse en début d'une liste. Ensuite, chaque noeuds qui voit passer le paquet, ajoute son adresse à cette liste des noeuds déjà traversés. La destination connaît donc le chemin pour aller à la source. Après ce mécanisme, tout paquet circulant entre la destination et la source contiendra en entête le chemin complet pour acheminer le paquet.

Cette méthode surcharge considérablement les entêtes de tous les paquets, mais résout le problèmes des **liens asymétriques**. En effet, le paquet RREQ peut emprunter une route et le paquet RREP une autre route, vu que les deux sont toujours envoyés en diffusion. Mais dans le paquet RREP, il y aura le chemin qu'a emprunté le paquet RREQ pour atteindre sa destination. Le noeud source pourra donc utiliser ce chemin pour envoyer son paquet même si le paquet RREP n'est pas revenu par le même chemin.

Pour illustrer ceci, voici la situation de la figure 2.3.

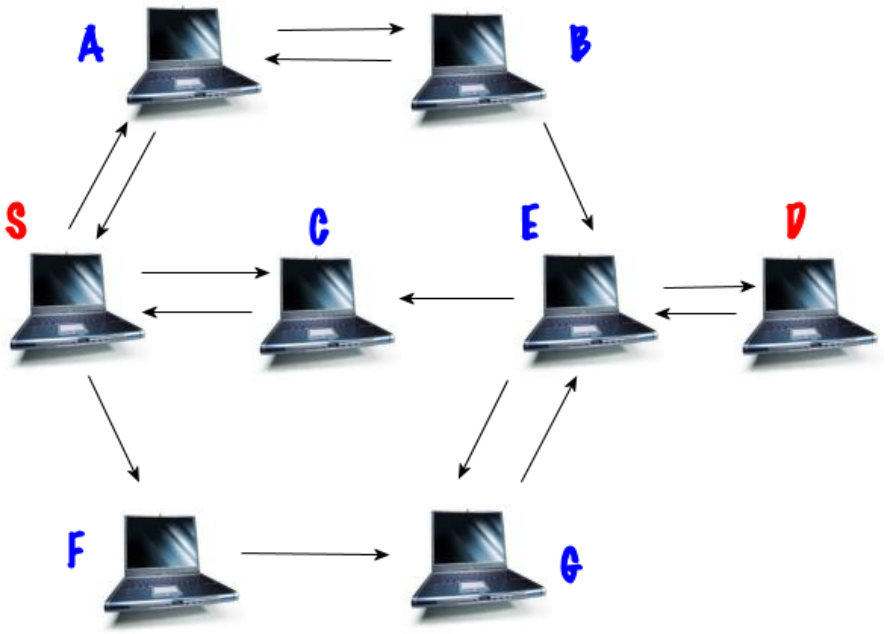


FIG. 2.3 – Le fonctionnement de DSR

Le noeud source S veut envoyer un paquet au noeud destination D. Les flèches entre les noeuds signifient que les noeuds peuvent communiquer dans le sens indiqué par la flèche. On remarque ici qu'il n'y a pas de chemin symétrique entre S et D, ie un chemin par lequel S et D pourraient envoyer leurs données dans les deux sens. Grâce au mécanisme de DSR, la communication est possible. Tout d'abord, S envoie un paquet RREQ en diffusion. Le paquet peut arriver à D par le chemin S -> A -> B -> E -> D (1) ou S -> F -> G -> E -> D (2). Considérons que le paquet arrive par le premier de ces chemins (1). Alors D encapsule dans l'entête d'un paquet RREP, le chemin (1). Celui-ci envoie en diffusion ce paquet RREP à S. Celui-ci ne peut atteindre S que par le chemin D -> E -> C -> S. Le noeud S enregistre le chemin (1) pour envoyer des paquets à D et la communication peut s'établir. Si D voulait envoyer à S, le même mécanisme fonctionnerait. Cela prouve que DSR peut fonctionner sur des liens asymétriques.

De plus, DSR permet de stocker plusieurs routes pour la même destination. La source pourra donc décider d'envoyer un paquet sur deux par exemple sur chaque chemin. Cela résoud donc le problème de **l'équilibrage des charges** du réseau [32].

Conclusion

Vis à vis des deux protocoles présentés ici, on peut indiquer qu'originellement AODV était était moins performant que DSR. Cependant, tous les ajouts qui lui ont successivement été apportés lui permettent d'être dans les protocoles de routage les plus complets à ce jour.

Les 2 parties qui viennent d'être présentées proposent deux approches différentes au problème du routage dans les réseaux ad'hoc. Une troisième solution consiste en une solution hybride, mélangeant les deux techniques évoquées ci-avant.

2.2.4 Un algorithme hybride : ZRP (Zone Routing Protocol)

A travers les deux sections précédentes, nous avons présenté deux approches qui avaient chacunes leurs avantages et inconvénients, l'idée du protocole qui va être présenté ici est de mélanger les deux approches.

Pour mélanger les deux approches, le protocole ZRP utilise la notion de zone. Celui qui implante le routage dans un réseau avec ZRP définit un rayon r pour délimiter des zones. Chaque noeud créé une zone autour de lui qui correspond à tous les noeuds qui sont à r sauts de lui. Les différentes zones se recoupent donc.

A l'intérieur des zones, on utilise du routage proactif et à l'extérieur du routage réactif. Le protocole proactif de ZRP est appelé IARP (*IntrAzone Routing Protocol*). Il créé donc des tables de routage pour chaque noeud de cette zone. Dans le cas où on veut router un paquet qui ne se trouve pas dans cette zone, on lance un protocole réactif, appelé IERP (*IntErzone Routing Protocol*). Ce protocole s'appuie sur les mobiles de bordure de zone. Au lieu de diffuser un paquet de recherche dans tout le réseau comme dans les protocoles réactifs, le noeud source demande aux noeuds bordure de sa zone, si ceux-ci n'ont pas la destination cherchée dans leur table de routage. Si ce n'est pas le cas, les noeuds bordures demandent à leurs noeuds bordures respectifs et ainsi de suite jusqu'à trouver la destination. Pour éviter la surcharge du réseau, chaque noeud bordure traite une recherche que s'il ne vient pas juste de la faire [32].

Conclusion

Les protocoles de routage des réseaux ad'hoc ont encore des progrès à faire, mais on comprend bien à travers cette étude que ces réseaux pourraient être intéressant dans bon nombre de domaines, et particulièrement dans celui des jeux vidéos qui semblent être parfaitement adapté pour cela. Après avoir évoqué les principaux algorithmes de routage ad-hoc disponibles, il est également important pour nous d'étudier les mécanismes de qualité de service existants, l'objectif du stage étant d'apporter des améliorations réseau à la jouabilité du jeu sur réseaux ad-hoc multisauts.

Nous en avons désormais terminé avec cet état de l'art. Nous avons ainsi pu tout d'abord comprendre comment les jeux-vidéos interagissaient avec le réseau de façon génerale, puis nous avons étudié cela sur un jeu en particulier : Quake III Arena. Ensuite, nous nous sommes penchés sur l'étude des réseaux ad-hoc à travers l'étude des différents algorithmes de base. Maintenant, penchons nous sur un aspect plus pratique à travers l'étude d'un jeu en particulier : Quake III Arena.

2.3 quake 3 Arena et les réseaux

2.3.1 Introduction

Maintenant que nous en avons fini avec la recherche bibliographique, étudions le cas d'un jeu en particulier : Quake III Arena. Ce jeu est un FPS (First Person Shooter), qui est sorti le 10 décembre 1999. Le choix d'étudier ce jeu plutôt qu'un autre s'appuie sur divers raisons. Tout d'abord, nous avons vu précédemment que les FPS sont les jeux les plus joués sur Internet, et ce sont également ceux qui sont les plus exigents en terme de qualité de réseau. Pour cela, il était intéressant de se pencher sur ce type de jeu. Ensuite, Quake III Arena est une sorte de référence dans le monde des jeux vidéo, de par sa qualité pour un jeu de cette époque, mais

aussi en ce qui concerne la recherche. En effet, il a très souvent servi de "benchmark", tout comme peut l'être par exemple l'image "Lena" en imagerie. Cela est peut-être du au fait que ce jeu est codé de façon très académique et typique, et qu'il est très facile d'avoir accès à tous les paramètres et à toutes les variables du jeu, via sa ligne de commande.

Dans la section qui suit, nous allons donc voir si les intéractions entre réseau et jeux vidéos évoquées au chapitre précédent se confirment dans la pratique. Ces expériences pourront alors nous permettre de réellement comprendre le fonctionnement réseau d'un jeu vidéo, et ainsi parfaire notre connaissance avant d'introduire du routage ad-hoc. A terme, nous pourrons alors atteindre notre objectif initial, à savoir permettre une qualité accrue et un nombre conséquent de joueurs sur un réseau wifi en mode ad-hoc multisauts.

2.3.2 Caractérisation du trafic réseau généré par le jeu Quake3

Pour comprendre comment se comporte le jeu Quake III Arena au niveau réseau, nous avons préparé plusieurs expériences. Les premiers tests se sont déroulés avec une version de démonstration, en attendant d'avoir acquis les jeux originaux accompagnés de leur licence. La version de démonstration permet de jouer en multijoueur, et nous avons rapidement réussi à jouer une partie en LAN, en mode ad'hoc ; pour savoir comment le réseau ad-hoc a été monté, voir l'annexe A.3.1. Cette partie a eu lieu sous windows, car la version linux du jeu ne fonctionne pas sur les ordinateurs mis à disposition. Ce jeu fonctionne en mode Client / Serveur, donc un des deux ordinateurs que nous avions a joué le rôle du serveur, et l'autre celui du client. Dans ce type de jeu, celui qui heberge le jeu peut aussi être client et jouer.

Pour comprendre comment se déroulaient les échanges d'information dans cette partie, nous avons mis en place sur le poste serveur Ethereal, un logiciel permettant de capturer et d'analyser le trafic.

L'objectif de notre projet consiste à étudier le trafic réseau engendré par le jeu vidéo, puis une fois celui-ci connu proposer une implémentation optimale sur réseaux ad'hoc. Le but de ces premières expériences est donc de faire fonctionner un jeu sur réseaux ad'hoc, et d'étudier ce qu'il se passe. Voici donc le compte-rendu de notre première expérience réalisée, et des premières conclusions.

Matériel utilisé

Cette première expérience s'est déroulée sur deux ordinateurs portables identiques Dell D610. Un des ordinateurs a pris le rôle de serveur, et l'autre de client. L'ordinateur du serveur utilisait une carte wifi PCMCIA Netgear WAG511, alors que le client utilisait sa carte wifi intégrée, une Intel PRO/Wireless 2200BG.

Sur ces deux ordinateurs a été installé le jeu Quake III Arena dans sa version de démonstration, sous système Windows XP service pack 2. De plus, sur le serveur a été installé le logiciel Ethereal version 0.10.14, logiciel qui avait en charge d'enregistrer le trafic réseau généré par la partie. Enfin, ces ordinateurs sont liés en réseau via un mode ad'hoc point à point, il a donc fallu au préalable configurer le réseau pour arriver à cette structure. Les deux ordinateurs étaient séparés de quelques centimètres, placés sur le même bureau.

Méthodologie

Pour voir l'impact des différentes phases de jeu sur le réseau, nous avons mis en place une méthode simple, qui est de réaliser différentes actions à des instants donnés, tout cela chronométré. Voici donc la méthode qui a été employée pour ce test :

1. **au temps t = 0 s** : le joueur responsable du serveur clique sur "multijoueur"
2. **au temps t = 10 s** : le serveur est lancé, il est prêt à héberger la partie et à accueillir d'autres joueurs
3. **t = 15 s** : le joueur client clique sur "multijoueurs" ; le serveur disponible ne tarde pas à s'afficher comme étant disponible
4. **t = 20 s** : le client sélectionne le serveur en cliquant dessus, il sera bientôt connecté à la partie (temps d'attente de 2 secondes environ)
5. **t = 45 s** : depuis près de 25 secondes les deux joueurs sont connectés et jouent l'un contre l'autre. A ce temps de 45 secondes, le joueur du poste client tue le joueur du poste serveur.
6. **t = 75 s (1mn15)** : le serveur redémarre l'arène : les joueurs sont redispersés, récupèrent toute leur vie et perdent les armes qu'ils avaient accumulées. La partie est redémarrée à 0.
7. **t = 120 s (2mn)** : Les joueurs ayant chacun un lance roquettes en main, ils se tuent mutuellement au même moment.
8. t = 145 s (2mn25) : Le joueur client se déconnecte de l'arène, et se retrouve ainsi à l'écran d'accueil du jeu. Le joueur serveur est désormais le seul en piste.
9. **t = 165 s (2mn45)** : Le joueur client clique sur "multijoueurs" à partir de l'écran d'accueil. Les serveurs disponibles s'affichent alors.
10. **t = 180 s (3mn)** : Le joueur client resélectionne le serveur, et la partie redémarre.
11. **t = 210 s (3mn30)** : Le joueur serveur tue le joueur client.
12. **t = 230 s (3mn50)** : Le joueur serveur quitte l'arène : la partie est interrompue.
13. **t = 240 s (4mn)** : Le joueur serveur quitte définitivement le jeu ; nous arrêtons alors la capture.

Les résultats

Ethereal a permis d'enregistrer tout le trafic, mais l'intégralité des données ne nous intéresse pas (différentes requêtes de maintenance du réseau wifi par exemple). Le logiciel permet de filtrer les paquets désirés. Ainsi, nous avons pu obtenir les courbes suivantes, qui correspondent au nombre de paquets transférés seconde par seconde, ainsi que le nombre d'octets transférés par seconde :

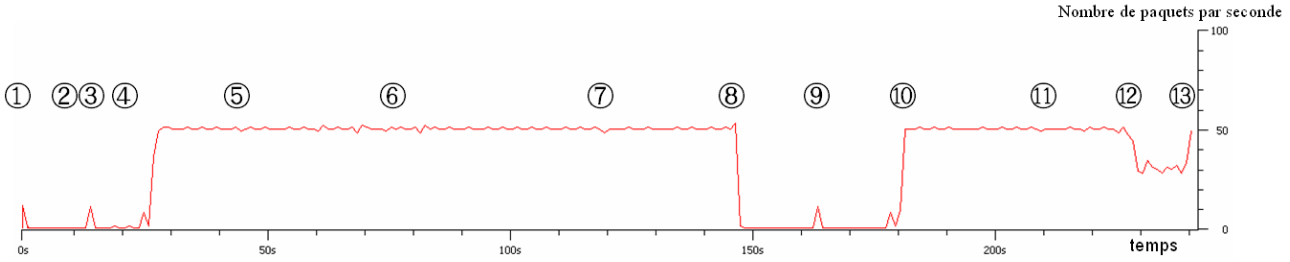


FIG. 2.4 – Le nombre de paquets transmis par seconde durant l'expérience

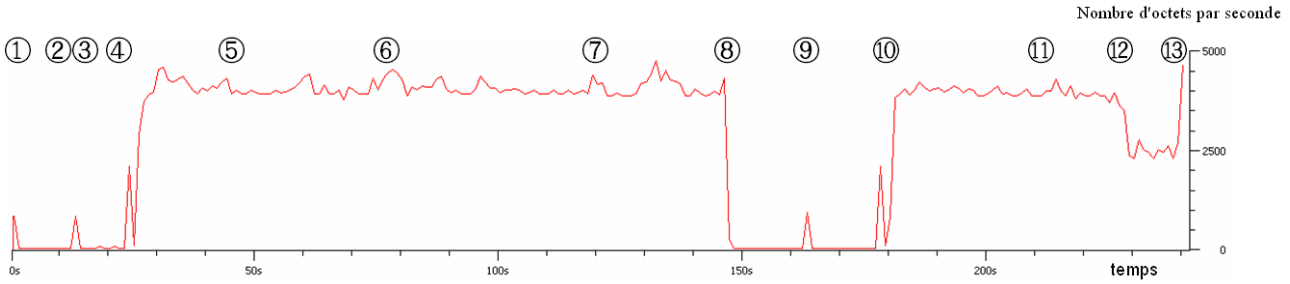


FIG. 2.5 – Le nombre d'octets transmis par seconde durant l'expérience

Sur ces figures, les chiffres entourés correspondent aux étapes que nous avons suivi lors de cette experimentation. Nous allons donc revenir point par point sur chacune des étapes pré-citées, et en deduire l'impact de ces phases sur le réseau.

L'étape 1) Cette première étape est le point de départ de l'expérimentation. C'est le moment où un des joueurs clique sue "multijoueurs". Nous avons constaté lors de précédentes expériences que le jeu n'émet strictement aucun paquet sur le réseau tant que l'on n'a pas cliqué sur ce mode. Lorsqu'un des joueurs a donc choisi ce mode multi-joueurs, le jeu va rechercher s'il y a des serveurs disponibles. Ceci explique pourquoi il y a un pic d'envoi de messages au temps 0, la station émet des requêtes en broadcast pour savoir si un serveur existe déjà. Nous avons remarqué qu'à chaque fois, ces broadcasts de recherche de serveur étaient constitués de 8 paquets.

L'étape 2) Elle a lieu au bout de 10 secondes : le serveur est lancé. Comme on peut le constater sur cette figure, cette phase n'a pas d'impact sur le réseau, contrairement à ce que l'on aurait pu penser. Il n'y aura plus d'émission de paquet tant qu'un client ne se sera pas manifesté pour joindre la partie.

L'étape 3) Il s'agit de la même étape que la 1, sauf que cette fois ci c'est le second joueur qui clique sur "multi-joueurs". Le fait de cliquer sur ce mode génère automatiquement une recherche de serveur, et donc... L'envoi en broadcast de 8 paquets. Nous obtenons donc là le second pic d'envoi de messages.

L'étape 4) Un serveur a répondu au client : il en existe bien un de disponible. Le client clique donc sur celui-ci pour joindre la partie. Les stations s'échangent des informations générales ayant pour but d'intégrer le client dans la partie ; ces informations impliquent un pic d'envoi juste avant le début du jeu. En comparant les deux figures, on constate que ces paquets de synchronisation sont peu nombreux, mais ils sont beaucoup plus gros que les autres.

L'étape 5) A partir de l'étape 4, le jeu est lancé, et on voit qu'en mode "jeu", le nombre d'envoi de paquets par seconde est incroyablement stable : il y a constamment un envoi de 50 paquets par seconde, à un ou deux près. La taille de paquet est également stable, comme on peut l'aperçevoir sur les deux courbes. On constate que les paquets ont alors approximativement une taille de 100 octets.
A l'étape 5, il y a eu un meurtre : le joueur client a tué le joueur serveur. On constate ici que cela n'a pas de conséquence sur le réseau, pas plus que les différents mouvements qui ont été réalisés.

L'étape 6) Ici, la partie est redémarrée. Une fois de plus, on constate que le nombre d'envoi de paquet est stable, même si ces paquets sont légèrement plus volumineux.

L'étape 7) A cette étape, il y a un double meurtre. Encore une fois, peu d'impact sur le réseau, étape assez similaire à l'étape 6 (comme à celle-ci les joueurs redémarrent la partie, sauf que les scores ne retombent pas à 0).

L'étape 8) Le client se déconnecte. On constate que le trafic chute très brutalement pour atteindre 0 octets transmis par seconde. Cela signifie que le serveur n'envoie des messages que lorsqu'au moins un client est connecté ; sinon, il reste silencieux.

L'étape 9) Etape similaire à la 1) et la 3), le client cherche à se reconnecter, donc il envoie un broadcast pour connaitre les serveurs qui sont disponibles. Evidemment, l'ancien serveur est toujours là, donc il répond et s'affiche à l'écran.

L'étape 10) Le client se reconnecte, étape identique à la 4 ; une fois de plus, la synchronisation génère beaucoup de trafic d'un coup, avant que le jeu ne reparte à un débit moyen de 50 paquets par seconde.

L'étape 11) Le joueur du serveur tue le joueur du client. Une fois de plus, pas d'impact notable sur le réseau.

L'étape 12) Le serveur se déconnecte. Pendant ce temps, le client est toujours connecté à la partie. Sur l'écran est alors indiqué qu'il est déconnecté, et le jeu cherche à retrouver la connexion. On constate nettement que l'envoie de paquet est exactement deux fois moins important à cet instant, et en regardant de près les messages qui sont envoyés, on constate qu'il n'y a plus que le client qui émet, à la même cadence qu'avant. En revanche, le serveur n'émet bien évidemment plus.
Cette étape est assez intéressante, car on aurait pu penser que les choses se passeraient comme lors de l'étape 8. On constate donc que le trafic entre client et serveur est asymétrique.

L'étape 13) Le joueur du serveur quitte totalement le jeu, et la simulation s'arrête. On remarque que le nombre de paquets envoyés augmente. En fait, il s'agit de réponses ICMP au client indiquant que le serveur n'est plus disponible sur le port du jeu. Il y a autant de ces requêtes que d'envoi de paquets de la part du client, qui n'arrête toujours pas ses émissions. Ceci explique donc pourquoi le nombre de paquets envoyés augmente.
Au bout d'environ une minute, si le client n'a toujours pas réussi à récupérer la connexion, le jeu quitte automatiquement la partie et renvoie le joueur à l'écran d'accueil.
Une dernière chose à noter : lorsque l'on fait quitter l'arène au serveur puis quitter le jeu, le client continue ses envois de paquets, espérant retrouver la connexion. En revanche, si on quitte directement le jeu sur le serveur, la partie du client est directement coupée et on se retrouve sur l'écran d'accueil.

Résultats généraux En ce qui concerne le type de trafic généré, on remarque que les paquets sont exclusivement des paquets UDP, qu'ils ont une taille quasiment fixe en moyenne de 80 octets, et que l'envoi est très cadencé et très stable dans le temps. Cela conforte l'idée que nous nous étions fait, à savoir que le temps de latence voire la gigue étaient importants ; ici,

ces paramètres sont très faibles, et de fait la qualité que nous avons perçu est parfaite.

Débit En terme de débit, le déroulement de ce jeu a induit une moyenne de 2978 octets transmis par seconde, correspondant à un débit de 0,024 Mbit/s (24 kbit/s). Cela est bien évidemment très loin de saturer le réseau, qui rappelons-le peut aller en théorie jusqu'à 11Mbit/s sur réseau 802.11b. Cela confirme donc le fait qu'il n'est pas nécessaire d'avoir une grosse connexion réseau pour pouvoir jouer correctement, ici une ligne RTC pourrait presque être suffisante.

Asymétrie de l'échange Ensuite, lors de cette experimentation, on a constaté que 61,6% des paquets allaient dans le sens Client -> Serveur, et le reste (38,4%) dans le sens Serveur -> Client. Le caractère asymétrique des échanges entre client et serveur mis en évidence dans la partie bibliographique puis lors des étapes 8 et 12 de notre expérimentation est donc une fois de plus confirmé ici. Cette proportion risque de chuter encore en introduisant d'autres clients, chose qui sera vraisemblablement faite ultérieurement.

Nombre de paquets et taux de perte Lors de cet échange, le client a envoyé 3411 paquets dont 19 broadcasts ; 3397 paquets étaient donc destinés en unicast au client. Sur ces 3397 paquets, tous ont été reçus par le client.
Le client a quant à lui envoyé 5509 paquets, dont 5490 étaient destinés au serveur en unicast. Sur ces 5490 paquets envoyés, tous ont été reçus. Le résultat est donc simple : il y a eu 0% de taux de perte sur cette communication. Lorsque le serveur s'est déconnecté, le PC captait toujours ce qu'envoyait le client, simplement le serveur ne répondait pas aux requêtes puisqu'il ne jouait plus.

Conclusion

Cette expérimentation nous a permis plusieurs choses. Tout d'abord, de vérifier en pratique ce que nous avions vu dans le premier chapitre, à savoir que les jeux vidéos envoyaient de petits paquets de façon très régulière, le tout à petit débit. De plus, il nous a permis de mieux comprendre les mécanismes mis en jeu lors de la connexion et de la déconnexion des clients et des serveurs, mais aussi de voir quels étaient les impacts des meurtres, déplacements et remise à zero de la partie sur le réseau. Nous avons ainsi pu comprendre que le jeu générait un trafic réseau facilement modélisable et interprétable dans le cas d'une liaison parfaite.

2.3.3 Le lagomètre et la qualité du jeu vidéo

Au cours du chapitre traîtant des jeux vidéos, nous avons pu constater qu'une des difficultés majeure qui apparaissait lorsque l'on testait les jeux était de pouvoir quantifier objectivement leur qualité et leur jouabilité. A travers nos recherches et nos tests sur le jeu Quake III Arena, nous avons pu constater que des informations objectives et cruciales étaient affichées sur l'écran du joueur, via ce que l'on appelle le **lagomètre**.

Le lagomètre, comme son nom l'indique, est censé détecter les "lags", c'est à dire les moments où la jouabilité du jeu est amoindrie. Ce lagomètre permet donc de se faire une idée assez précise de la jouabilité du jeu, en se basant nous allons le voir sur les variables que nous avons

précédemment citées. Pour mettre les choses dans leur contexte, voici une capture d'écran du jeu avec le lagomètre affiché :

FIG. 2.6 – Capture d'écran du jeu et de son lagomètre

L'avance et le retard des images affichées Le lagomètre est le petit carré dessiné en bas à droite de l'image. On peut aperçevoir à mi-hauteur un "trait" bleu accompagné de zones jaunes. Ces traits correspondent simplement au retard ou à l'avance qu'a l'image par rapport au temps où elle aurait dû être affichée. Lorsque le trait est bleu, cela signifie que l'image a été traitée en avance, et lorsque le trait est jaune, l'image a été traîtée en retard.

La hauteur des traits, qu'ils soient bleus ou jaunes, est propostionnelle à l'avance (ou au retard) de l'image affichée. Lorsque tout va bien comme sur la capture d'écran ci-dessus, on voit très peu les traits jaunes et bleus : les informations sont ainsi traitées à temps, ni trop tôt ni trop tard. Dans des conditions parfaites, on ne devrait voir ni trait bleu ni trait jaune, mais en pratique cela n'arrive jamais. En effet, de par la nature complexe des échanges client / serveur du jeu, les images ne sont jamais affichées exactement au moment où elles le devraient.

Comme on peut l'apercevoir à la figure 2.7, lorsqu'il y a de grosses dégradations, on voit plus de bleu et surtout beaucoup plus de jaune, signe que les images arrivent soit trop en avance, soit beaucoup trop en retard.

Le ping et la perte Ensuite, tout en bas de ce lagomètre on peut aperçevoir de petits points verts : il s'agit du temps de ping du jeu. La hauteur de chaque trait correspond à la durée du

ping. Sur la figure 2.6, le ping est très faible (1ms) voire quasi nul, c'est pour cela que sur cette capture d'écran on ne distingue que très difficilement ces traits verts. Lorsqu'il y a perte de paquet (ping > 900ms), les traits ne sont plus verts mais deviennent rouges.

Pour mieux se rendre compte du fonctionnement du lagomètre, voici plusieurs captures d'écran à différents moments et avec différentes dégradations réseau :

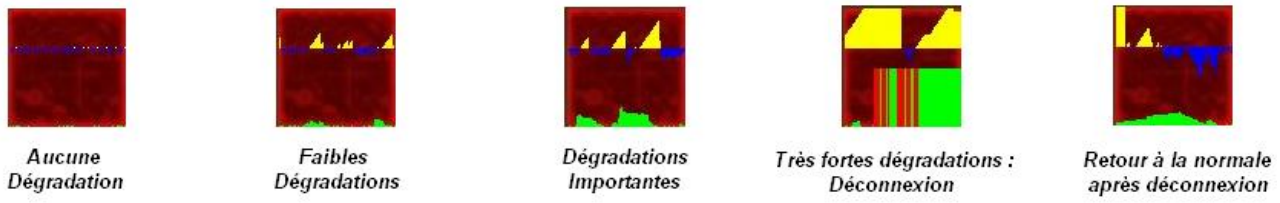


FIG. 2.7 – L'affichage du lagomètre en fonction des dégradations

Ici dans les cas les plus défavorables, la courbe du haut est quasiment entièrement jaune (toutes les images sont affichées en retard et le retard s'accumule), et en bas le ping est très important (courbe verte haute), certaines fois il y a même de la perte de paquet symbolisée par les traits rouges. Dans ces cas là l'image est quasi gelée et il n'est pour ainsi dire plus possible de jouer.

Tirer des informations de qualité du lagomètre et des mécanismes du jeu Notre principal objectif ici est bien sûr de pouvoir récupérer des métriques de qualité pertinentes au coeur du jeu pour pouvoir juger de sa jouabilité. Le lagomètre nous permet d'en récupérer certains, mais notre analyse approfondie du code source de Quake nous a permis d'en trouver d'autres. Voici un petit résumé des métriques que l'on peut mesurer en vue d'évaluer la qualité du jeu vidéo :

- `Le retard et l'avance des images` : Cet élément est vraisemblablement le plus important, il est récupérable au niveau du lagomètre lui-même
- `Le ping et le taux de perte` : Comme nous l'avons vu, le lagomètre et les variables qu'il utilise permettent d'obtenir le ping du jeu mais aussi la perte de paquet
- `Le gel des images` : A chaque instant nous avons l'identifiant de l'image qui est affichée chez le client : nous sommes donc en mesure de savoir si l'image est gelée ou pas
- `Le nombre de déconnexions` : Il est possible de savoir combien de fois la fonction relative à la déconnexion a été appelée, et donc combien de fois le jeu a estimé qu'il y avait déconnexion du client

Ces quatre critères sont selon nous très représentatifs de la jouabilité du jeu, et ils ont le très grand avantage d'être des critères objectifs et quantifiables. Par la suite lors de nos expérimentations, nous aurons donc à utiliser ces critères.

2.3.4 Conclusion

Nous en avons désormais terminé avec cet aspect bibliographique qui nous a permis d'avoir un aperçu sur les domaines du jeu vidéo et des réseaux ad-hoc. Maintenant, nous allons évoquer la conception que nous avons émise en vue de la phase finale de réalisation.

Chapitre 3

Caractérisation et modélisation des études à réaliser

Comme nous l'avons évoqué en introduction, notre objectif majeur est de connaître les limites que peuvent offrir les réseaux ad-hoc pour permettre une bonne jouabilité des jeux vidéos. Ainsi, nous devons nous orienter dans deux directions :

- Que peuvent offrir les réseaux ad-hoc aux jeux vidéos en terme de qualité de service ?
- Quels sont les besoins réels du jeu vidéo en terme de qualité de réseau ?

Sur ce premier point, nous procèderons par simulation, et verrons ce que peuvent offrir les réseaux ad-hoc en terme de délai et de taux de perte notamment. Notre but sera de voir combien de joueurs peuvent participer à la partie sans dégrader le réseau de façon trop importante, quel est l'impact de l'algorithme de routage sur les délais et les pertes, ou encore quel est l'impact de la mobilité des noeuds et en particulier du serveur.

Sur le second point, nous aurons donc à dégrader le réseau progressivement, et voir l'impact sur le jeu vidéo. Pour mesurer cet impact, nous utiliserons bien évidemment les critères de jouabilité évoqués au chapitre 2.3.3 sur le lagomètre. Nous serons donc en mesure de définir les limites en terme de délai et de perte notamment à ne pas dépasser pour pouvoir jouer convenablement.

Dans ce chapitre, nous allons modéliser ce que nous comptons réaliser sur ces parties "capacités des réseaux ad-hoc" et "besoins des jeux vidéos", ainsi que les résultats finaux attendus.

3.1 Trouver les limites offertes par les réseaux ad-hoc

La première étape de notre travail consiste donc à étudier le comportement des réseaux ad-hoc dans des conditions analogues à celles de joueurs de jeux vidéos. Lors de ces tests, nous tâcherons donc de simuler le plus fidèlement possible le trafic réseau du jeu vidéo, mais aussi les faibles capacités en énergie des consoles portables. Voici comment vont se dérouler les simulations, à travers un schéma :

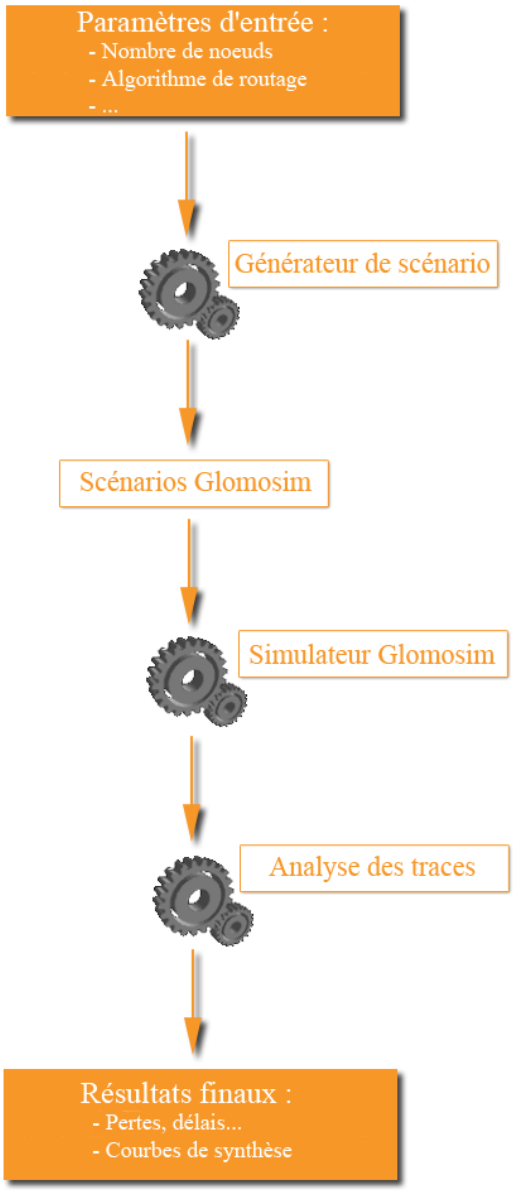


FIG. 3.1 – Les différentes étapes de la simulation réseau

Les paramètres d'entrée que nous ferons varier seront le nombre de noeuds, leur disposition sur la carte, leur mobilité ainsi que la mobilité du serveur. Nous ferons également varier les algorithmes de routage utilisés. Ensuite, une fois ces paramètres retenus, il nous faudra réaliser tous les scripts : c'est la seconde étape. Nous envisageons de refaire l'expérience une vingtaine de fois pour chaque point avec un générateur de nombre aléatoire à chaque fois différent, pour avoir un bon intervalle de confiance. Ainsi, si par exemple on utilise 3 algorithmes de routage différents, 15 nombre de noeuds différents... On se retrouve aisément à 900 scripts à réaliser ! C'est pourquoi il faudra créer un programme qui génèrera automatiquement tous les scripts en fonction de paramètres qu'on lui indiquera.

Une fois tous les scripts créés, il faudra les lancer. Là encore, il sera nécessaire de réaliser des programmes permettant de lancer de façon automatique tous ces scripts, puis de les analyser. Enfin, à l'issue de l'analyse, on pourra tracer des courbes et avoir des résultats quant à ce que peuvent offrir les réseaux ad-hoc à un trafic de jeu vidéo. On pourra par exemple savoir que dans telle configuration de jeu, nous aurons tant de perte de paquet et de délai.

La seconde étape sera bien sûr de savoir si ces caractéristiques de perte et de délai sont adéquates à ce que requiert un jeu, nous passons au second aspect de l'étude.

3.2 Les jeux vidéos et leurs besoins en qualité de réseau

Pour cette partie de l'expérience, nous aurons à étudier l'impact direct des perturbations réseau sur le jeu. Pour cela, nous utiliserons un émulateur de réseau, en l'occurence DummyNet, qui introduira des dégradations sur la liaison. Voici l'architecture de la plateforme de test :

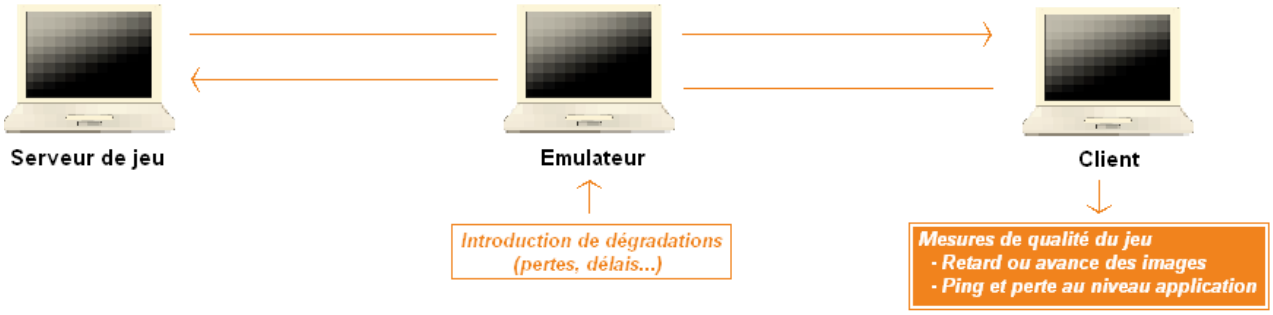


FIG. 3.2 – La mesure de qualité du jeu en fonction des dégradations réseau

Les paramètres d'entrée seront des dégradations réseaux croissantes, comme le délai, la perte, ou encore la gigue. Ces paramètres d'entrée seront logiquement ceux que nous aurons récupéré en sortie de nos simulations, nous permettant ainsi d'introduire des dégradations analogues à celles d'un vrai réseau ad-hoc.

La méthodologie à mettre en place sera simple : il y aura 2 PC sous Windows qui joueront à Quake III Arena, et entre ces deux PC il y en aura un troisième, qui aura en charge de dégrader le réseau. Différentes dégradations seront alors appliquées, et deux joueurs (un côté client et un côté serveur) évolueront et combatteront pendant 2 minutes, pendant que le jeu stockera toutes les informations nécessaires dans des fichiers de log. A partir de ces fichiers, il nous sera possible de conclure quant à la jouabilité du jeu en fonction des dégradations introduites.

En combinant ces deux aspects de simulation et d'émulation, nous pourrons donc conclure quant aux questions que nous nous posons ; pour conclure cette partie de modélisation, voici un diagramme offrant une vue d'ensemble du dispositif.

3.3 Synthèse

Pour synthétiser les actions à mener, il suffit simplement de lier les deux étapes précédemment citées en mettant en entrée de la seconde les paramètres de sortie de la première. Voici ce que l'on obtient :

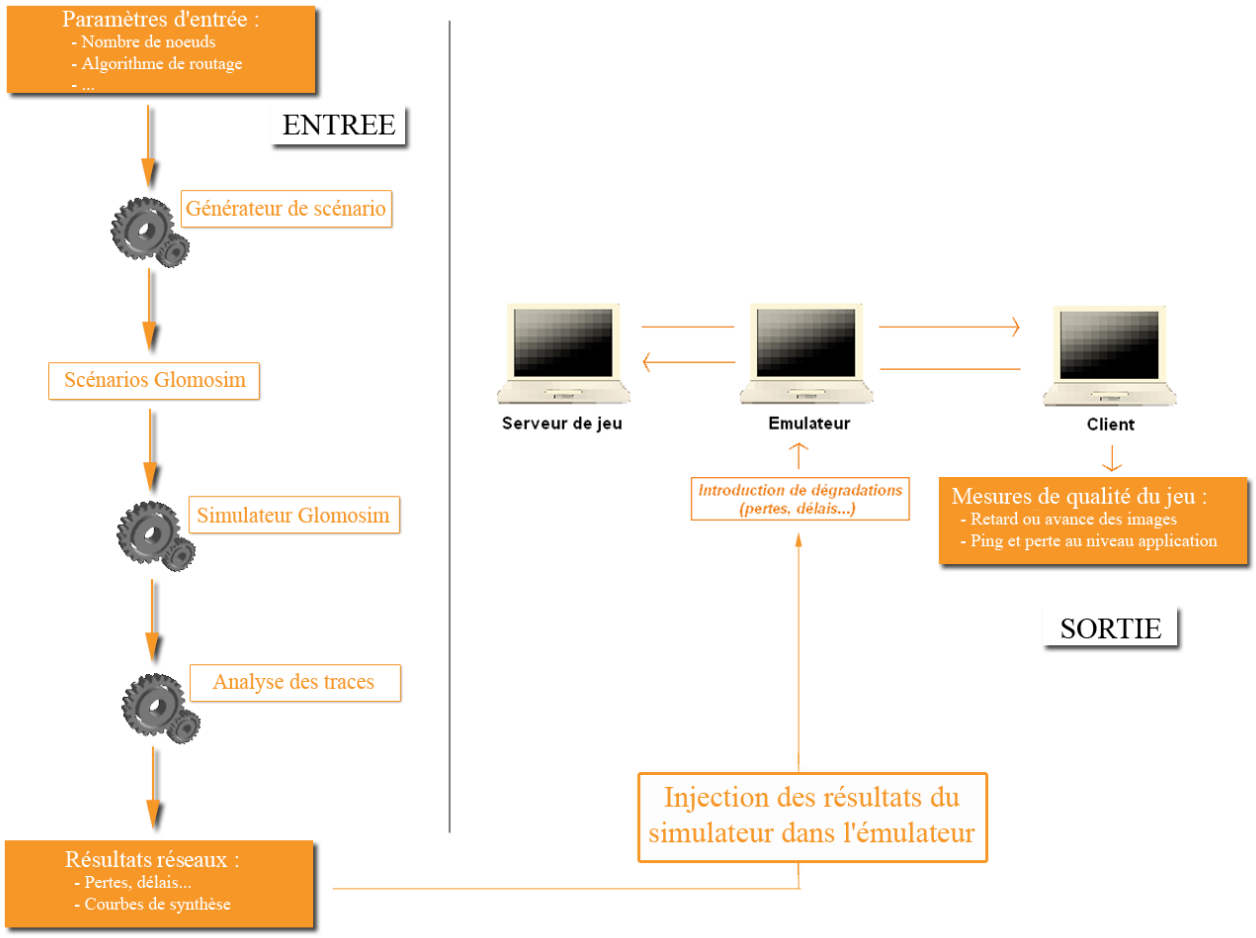


FIG. 3.3 – Une vue globale des étapes à réaliser

A travers ce processus, on indiquera en entrée les paramètres réseaux voulus : nombre de noeuds, mobilité, ou encore algorithme de routage utilisé. En sortie, nous aurons la qualité et la jouabilité du jeu. Ainsi, nous pourrons répondre aux questions cruciales que nous nous posons, à savoir notamment :

- Jusqu'à combien de joueurs peut-on jouer aux jeux vidéos en ad-hoc ?
- Tous les algorithmes de routage permettent-ils de jouer convenablement aux jeux vidéos ?
- Le fait que les joueurs se déplacent pose-t-il un problème au jeu vidéo ?

Cette partie conception et modélisation des études à mener est maintenant terminée, nous allons maintenant évoquer la dernière partie du stage, à savoir la partie réalisation.

Chapitre 4

Phase de réalisation : Etude de Performances

Maintenant que nous avons évoqué la conception du projet, voyons maintenant comment l'étude a été menée de façon pratique. Evoquons tout d'abord La plateforme réseau mise en place à travers l'émulateur.

4.1 Le développement de la plateforme réseau et de la partie émulation

Cette plateforme réseau est la réalisation de la figure 3.2. L'objectif ici est de relier 2 PC qui jouent au jeu en interposant entre eux un perturbateur de réseau : DummyNet. Voyons donc à présent comment fonctionne Dummynet avant de voir les scripts et expériences que nous avons réalisés.

4.1.1 Le fonctionnement de Dummynet

DummyNet est un outil disponible sur système FreeBSD. C'est un outil système nécessitant une compilation du noyau freeBSD.
Dummynet fonctionne de pair avec le firewall système ipfw, un équivalent de iptables. Avec le firewall, on peut filtrer un type de trafic particulier, et avec Dummynet on peut lui appliquer la dégradation que l'on veut. Voici un exemple de commande :

```
Ipfw add prob 0.1428 pipe 1 ip from 192.168.10.1 to any
Ipfw pipe 1 config bw 250Kbit/s delay 50ms plr 0.2
```

Sur la première ligne, on créé notre premier pipe. Celui-ci sera mis en place pour tout le trafic qui partira du PC ayant pour IP 192.168.10.1 et à destination de n'importe quelle machine. Prob indique une probabilité, ici on aura une chance sur sept que ce pipe soit utilisé ; en pratique, cela permet de mettre des actions aléatoires sur les paquets pour une même liaison.
Ensuite sur la seconde ligne, on indique les caractéristiques que l'on veut appliquer à ce pipe. Ici, on permet une bande passante de 250Kbit/s, et on introduit 50ms de délai et 20% de perte. Pour introduire des délais variable et ainsi de la gigue, il suffit de jouer avec `prob` en appliquant plusieurs pipes à une même liaison avec des probabilités diverses.

Avec Dummynet, il est donc possible d'introduire de la perte de paquet, du délai, mais aussi de limiter la bande passante, et d'introduire de la gigue. On peut donc parfaitement contrôler les dégradations réseau que l'on introduit, et pouvoir dire quel est l'impact sur le jeu vidéo.

4.1.2 L'architecture mise en place

La plateforme qui a été montée est assez simple, elle suit le schéma 3.2. Le PC du milieu utilisant Dummynet a simplement joué le rôle de routeur. Ainsi, sur ce PC ont été installé deux cartes Ethernet, et un routage statique a été mis en place, en mettant la première interface en lien avec le serveur, formant ainsi un premier VLAN, et la seconde carte en lien avec le client, formant un second VLAN.

4.1.3 Les scripts développés

Rappelons notre objectif sur cette phase : il s'agit de savoir quel est l'impact des caractéristiques du réseau sur le jeu. Ainsi, le but est de faire varier les dégradations en entrée, et de voir en sortie quel est la qualité du jeu obtenue.

Pour cela, voici le type de scripts que nous avons mis en place :

```
ipfw -q flush

ipfw pipe 1 config bw 250Kbit/s delay 6.25ms plr 0.2
ipfw pipe 2 config bw 250Kbit/s delay 12.5ms plr 0.2
ipfw pipe 3 config bw 250Kbit/s delay 18.75ms plr 0.2
ipfw pipe 4 config bw 250Kbit/s delay 25ms plr 0.2
ipfw pipe 5 config bw 250Kbit/s delay 31.25ms plr 0.2
ipfw pipe 6 config bw 250Kbit/s delay 37.5ms plr 0.2
ipfw pipe 7 config bw 250Kbit/s delay 43.75ms plr 0.2

ipfw add prob 0.1428 pipe 1 ip from any to any
ipfw add prob 0.1616 pipe 2 ip from any to any
ipfw add prob 0.2 pipe 3 ip from any to any
ipfw add prob 0.25 pipe 4 ip from any to any
ipfw add prob 0.33 pipe 5 ip from any to any
ipfw add prob 0.5 pipe 6 ip from any to any
ipfw add pipe 7 ip from any to any
```

La première ligne permet de supprimer tous les pipes éventuellement créés précédemment. Sur cet exemple, on a un délai moyen introduit de 100ms. En effet, **any to any** introduit des dégradations sur chaque liaison, et il y en a 4 ici : du serveur vers dummynet, de dummynet vers le client, du client vers dummynet, et de dummynet vers le serveur. L'écart type des délais est ici de 50ms, et on donne à chacun des 7 pipes la même probabilité d'être utilisé.

Si l'on avait voulu avoir un délai de 200ms et un écart type de 100ms, il suffisait de multiplier tous les délais par 2. En pratique, c'est ce que nous avons fait, et nous avons ainsi pu tracer des courbes de type :

- Qualité du jeu en fonction du délai introduit
- Qualité du jeu en fonction de la gigue

– Qualité du jeu en fonction de la bande passante
– Qualité du jeu en fonction du taux de perte

Nous évoquerons ces résultats dans le chapitre qui y est consacré, voyons maintenant quels sont les critères de qualité que nous avons relevé.

4.1.4 Quake 3 et les critères de qualité du jeu

Comme nous l'avons vu dans la partie bibliographique consacrée aux jeux vidéos, il est souvent difficile de juger objectivement la qualité du jeu vidéo. Notre choix a été de récupérer des critères objectifs au sein du code source du jeu Quake III Arena. L'intérêt de ce jeu est multiple. Tout d'abord, son code source est libre, donc adaptable. Ensuite, c'est un jeu assez répandu et joué dans le monde du jeu vidéo. Enfin, c'est un FPS (First Person Shooter) qui sont, comme nous l'avons vu, les jeux les plus sensibles aux dégradations du réseau. Donc, si Quake III Arena fonctionne bien au niveau du réseau, on peut penser que tous les jeux fonctionneront également bien.

Pour nos expérimentations, nous avons remarqué que les paramètres les plus importants à prendre en compte étaient l'avance (ou le retard) des images, ainsi que le ping et le taux de perte au niveau de l'application. En effet, le premier paramètre mesure la jouabilité via la distorsion et le gel des images, pendant que l'autre mesure le retard. Ce dernier élement pris seul n'influence pas du tout la fluidité du jeu, on reçoit les informations en retard mais de façon cadencée et bien ordonnée. En revanche, il est primordial d'avoir un ping faible pour avoir l'information le plus tôt possible, et ainsi être plus sûr de toucher sa cible.
En mesurant le retard et l'avance des images, on mesure la fluidité du jeu ou en d'autre terme le "lag", qui est une des composantes selon nous de la jouabilité. Pour qu'un jeu en réseau ait une jouabilité optimale, on doit à la fois avoir un temps de réponse (ping) très faible, et une fluidité parfaite (pas de gel des images, pas de distorsion, pas de "lag", toutes les images arrivent de façon cadencée et ordonnée). Nous avions également d'autres paramètres mesurant notamment uniquement le gel des images et aussi le nombre de deconnexions, mais ces paramètres peuvent être englobés dans celui qui mesure l'avance et le retard des images.

Dans la section résultats, nous présenterons ainsi les paramètres de qualité relevés dans le jeu en fonction de la dégradation réseau introduite par Dummynet. Maintenant, évoquons la seconde partie de notre travail : l'aspect simulation sur GloMoSim.

4.2 L'étude de performance sur simulateur GloMoSim

Le simulateur GlomoSim va nous permettre de traiter le second aspect du travail, à savoir : que peuvent fournir les réseaux ad-hoc aux jeux vidéos en terme de performance de réseau ? Nous avons vu au chapitre précédent ce que nous voulions obtenir à partir de ce simulateur, voyons donc maintenant comment il fonctionne en pratique, ainsi que les scripts que nous avons réalisé.

4.2.1 Le fonctionnement de GloMoSim

GlomoSim est un outil libre disponible sous linux et sous windows. Il permet de simuler différents architectures, en particulier celle des réseaux ad-hoc. GlomoSim est basé sur une librairie nommée "Parsec", qui fait qu'il est possible de paralléliser les traitements. GlomoSim dispose à priori de peu de modèles pour les réseaux ad-hoc, dont certains sont bogués. Nous

avons néanmoins à notre disposition deux des algorithmes majeurs : OLSR et AODV. Voici un exemple de script GlomoSim :

exemple.in

```
1  SIMULATION-TIME                          120S
2  SEED                                     0
3  TERRAIN-DIMENSIONS                       (300, 300)
4  NUMBER-OF-NODES                          4
5
6  NODE-PLACEMENT                           FILE
7  NODE-PLACEMENT-FILE                      C:\experiences\noeuds\4noeuds.input
8
9  MOBILITY                                 NONE
10
11 PROPAGATION-LIMIT                        -111.0
12 PROPAGATION-PATHLOSS                     TWO-RAY
13 NOISE-FIGURE                             10.0
14 TEMPARATURE                              290.0
15 RADIO-TYPE                               RADIO-ACCNOISE
16 RADIO-FREQUENCY                          2.4e9
17 RADIO-BANDWIDTH                          11000000
18 RADIO-RX-TYPE                            SNR-BOUNDED
19 RADIO-RX-SNR-THRESHOLD                   10.0
20
21 RADIO-TX-POWER                           0.62965
22
23 RADIO-ANTENNA-GAIN                       0.0
24 RADIO-RX-SENSITIVITY                     -91.0
25 RADIO-RX-THRESHOLD                       -81.0
26 MAC-PROTOCOL                             802.11
27 NETWORK-PROTOCOL                         IP
28 NETWORK-OUTPUT-QUEUE-SIZE-PER-PRIORITY   100
29
30 ROUTING-PROTOCOL                         AODV
31
32 APP-CONFIG-FILE                          C:\experiences\application\4noeuds.conf
33
34 APPLICATION-STATISTICS                   YES
35 TCP-STATISTICS                           NO
36 UDP-STATISTICS                           NO
37 ROUTING-STATISTICS                       NO
38 NETWORK-LAYER-STATISTICS                 NO
39 MAC-LAYER-STATISTICS                     NO
40 RADIO-LAYER-STATISTICS                   NO
41 CHANNEL-LAYER-STATISTICS                 NO
42 MOBILITY-STATISTICS                      NO
43
44 GUI-OPTION                               NO
45 GUI-RADIO                                NO
46 GUI-ROUTING                              NO
```

FIG. 4.1 – Un exemple de script GloMoSim

Dans les scripts GloMoSim, on doit indiquer tous les paramètres désirés pour les simulations à effectuer. A la ligne 1 de la figure 4.1, on indique tout d'abord combien de temps est censé durer la simulation ; ici, 2 minutes. A la ligne 2, il s'agit du générateur de nombre pseudo-aléatoire. Ligne 3, on définit les dimensions du terrain, et ligne 4 on indique le nombre de noeuds désirés. Ligne 6 et 7 on définit la position des noeuds, position indiquée dans un autre fichier nommé 4noeuds.input. A la ligne 9, on indique la mobilité des noeuds, ici il n'y en a pas.

Les lignes 11 à 28 correspondent aux propriétés radio de la liaison. Notons dans ces lignes deux paramètres importants : la ligne 19 indique la bande passante disponible, et la ligne 21 correspond à la portée radio ; ici, 0,63 correspond à une portée de 120 mètres.

A la ligne 30, on indique le protocole de routage à utiliser : ici, il s'agit d'AODV. Ligne 32 on donne le fichier de configuration que l'on utilisera pour les transferts niveau applicatif. Enfin, de la ligne 34 à 46 il s'agit des traces que l'on souhaite récupérer, notamment si l'on veut des informations au niveau MAC, IP, TCP ou application.

Le fichier de configuration concernant la position des noeuds fonctionne de façon assez simple. Voici un exemple de ligne pouvant y être incorporé :

```
3   0 (100, 100, 0)
```

Ici, le noeud 3 sera simplement positionné au point de coordonnées x = 100, y = 100, et z = 0. Le 0 correspond à la mobilité.

En ce qui concerne le second fichier relatif aux échanges applicatifs, voici un exemple de données qui peuvent être incorporées :

```
CBR 0 1 0 80 50MS 0S 0S
```

Ici, le transfert aura lieu entre le noeud 0 et le noeud 1, et il s'agira d'un transfert de type CBR. Le second 0 correspond au nombre de paquets que l'on souhaite envoyer ; en mettant 0, les paquets seront envoyés en continu jusqu'à la fin de l'échange. Ensuite, 80 correspond à la taille du paquet, 50MS signifie que les paquets seront envoyés toutes les 50MS. Les deux 0S indiquent le moment de début et de fin de l'échange. En mettant 0, cela signifie que les dates de début de la fin de l'échange seront les mêmes que celles de début et de fin de la simulation.

Le choix de tous ces paramètres n'est pas anodin, il s'inscrit bien sûr dans un choix de notre part de coller au plus près de la réalité. Le choix du type de trafic échangé, de la portée des noeuds ou encore de leur mobilité sera donc dicté par le contexte applicatif, qui sera le jeu vidéo entre amis.

Maintenant que nous avons évoqué le fonctionnement de GloMoSim et de ses scripts, voyons en pratique comment nous avons réalisé et exécuté nos propres scripts.

4.2.2 La méthodologie utilisée

Pour arriver à des résultats interprétables, il nous faut réaliser tout une série d'étapes qui sont retracées sur la figure 3.1. Nous allons donc ici expliquer chacune de ces étapes et ainsi la méthode que nous avons utilisé pour arriver à nos résultats.

Le générateur de script en langage C

Le fichier C qui a été créé permet de générer automatiquement tous les scripts voulus. Pour les expériences et premiers résultats que nous allons présenter à la prochaine section, nous avons dû générer pas moins de 600 scripts : 20 seed différents, 2 algorithmes et 15 nombre de noeuds différents (de 2 à 16 joueurs).

Avec ce programme, il est possible de ranger tous ces scripts dans différents dossiers, et de rendre l'analyse amplement simplifiée. Une fois tous ces scripts créés, il faut donc les lancer et les analyser, cela une fois de plus de façon automatisée.

Lancement et analyse des scripts

Pour analyser et lancer tous ces scripts, nous avons eu recours à des scripts shell et awk. Or, nos tests se sont déroulés dans un environnement Windows ; nous avons donc eu recours à MinGW, un outil analogue à Cygwin.

Donnons un exemple de ligne de code qui peut être incluse dans les fichiers scripts d'analyse :

```
cat glomo.stat | grep packets | awk -f tauxPerte.awk >> ~/resultat.stat
```

Par défaut, GloMoSim rend en sortie un fichier nommé glomo.stat. Dans ce fichier, toutes les lignes concernant la perte de paquet contient le mot "packets", on les récupère donc via un grep. Il ne nous reste plus qu'à utiliser un programme awk pour calculer des moyennes à partir de cela ; voici un exemple de programme awk utilisé :

tauxPerte.awk

```
BEGIN{

sumcountsentserveur = 0;
sumcountrcvserveur = 0;

sumcountsentclient = 0;
sumcountrcvclient = 0;

}
{
if ($10 == "received:")
{
        if ($2 == "0,")
        {
                sumcountrcvserveur+= $11;
        }
        else {
                sumcountrcvclient+= $11;
        }
}
else {

        if ($10 == "sent:"){
                if ($2 == "0,") {
                        sumcountsentserveur+= $11;
                }
                else {
                        sumcountsentclient+= $11;
                }

        }
}
}
END{

printf("TdP_S->C:\t%f\t", 100-((sumcountrcvclient*100)/sumcountsentserveur));
printf("TdP_C->S:\t%f\t\n", 100-((sumcountrcvserveur*100)/sumcountsentclient));

}
```

FIG. 4.2 – Le fichier awk utilisé pour le calcul du taux de perte

Dans les lignes du fichier glomo.stat concernant l'envoi et la réception de paquet, tout ce qui concerne l'envoi est indiqué par un "sent :", et ce qui concerne la réception par "received :". Donc avec le programme awk, on calcule le nombre total de paquets reçus, tous les paquets envoyés, et à partir de là il est aisé de calculer le taux de perte.

Comme on peut l'apercevoir sur la ligne de commande ci-dessus, le résultat sera envoyé vers le fichier resultat.stat.

Les résultats finaux

Au bout de cette chaîne, nous obtenons finalement un fichier textuel brut. L'application d'un dernier script nous permet de faire une moyenne parmis les 20 expériences différentes, et ainsi avoir un très bon intervalle de confiance ; rappelons en effet que nous lançons 20 simulations (avec 20 seed différents) par point.

Ces résultats peuvent désormais être interprétés, dans un tableur de type excel ou via l'utilisation de logiciels comme gnuplot.

Il est donc désormais possible d'interpréter les résultats obtenus en sortie du simulateur. Maintenant, il ne nous reste plus qu'à expliquer les expérimentations que nous avons menées et les résultats que nous avons obtenus.

4.3 Résultats obtenus

4.3.1 Dummynet et l'impact du réseau sur le jeu

A la section précédente, nous avions vu comment fonctionnait Dummynet. A partir de là, nous avons tout simplement fait varier les paramètres que sont le délai, la perte, la bande passante et la gigue. Voyons donc dès à présent les expériences que nous avons mises en place.

Les expérimentations

Pour chaque point des courbes que nous avons tracé, nous avons reproduit l'expérience trois fois et fait une moyenne. L'expérience consistait à jouer à Quake III Arena avec un client et un serveur, et Dummynet au milieu qui introduisait des dégradations. Il y avait un joueur côté serveur, mais qui n'intervenait pas (on ne le faisait pas jouer). Il y avait à chaque fois un même nombre de bots (joueurs contrôlés par l'ordinateur), à savoir 5. La carte utilisée était toujours la même. Enfin, chaque expérience durait deux minutes, pendant lesquelles le joueur côté client se défendait contre les bots. A partir de notre expérience sur le jeu, nous avons pu émettre un tableau permettant d'évaluer la qualité du jeu, et ainsi mieux appréhender les courbes à venir :

Note	Retard des images	Ping du jeu
0	> 60	> 250
1	50 - 60	200 - 250
2	40 - 50	150 - 200
3	30 - 40	100 - 150
4	20 - 30	50 - 100
5	< 20	< 50 ms

FIG. 4.3 – Les notes de qualité du jeu en fonction du délai et du retard des images

Ce que l'on entend par "retard des images" correspond en fait à l'avance et au retard des images. Nous avons alors mis ces distorsions en valeur absolue, ce qui nous donne un écart entre le temps auquel l'image a été affichée et le temps auquel elle aurait effectivement dû l'être. Par abus de langage, par la suite le "retard des images" correspondra donc à cette distorsion en valeur absolue et non uniquement au retard lui même.

Voici maintenant les résultats que nous avons obtenus sur cette première partie d'expérimentation.

a) Le délai

Pour cette expérience, nous avons augmenté progressivement le délai au niveau de Dummynet. Nous avons fixé la perte de paquet à 20% pour toutes les expériences, ainsi qu'une bande passante de 250Kbit/s. Enfin, l'écart type relatif des délais est resté constant, et égal à 0,667. En augmentant les délais on augmente donc aussi les écarts-types de façon proportionnelle.

Le fait d'introduire d'autres dégradations en plus de celle qui nous intéresse (ici le délai) n'est pas anodin ; en effet, n'introduire qu'une dégradation à la fois n'est pas du tout réaliste

et représentatif d'un réseau ad-hoc, et donnera donc des résultats biaisés. Voici les premiers résultats obtenus vis à vis du délai :

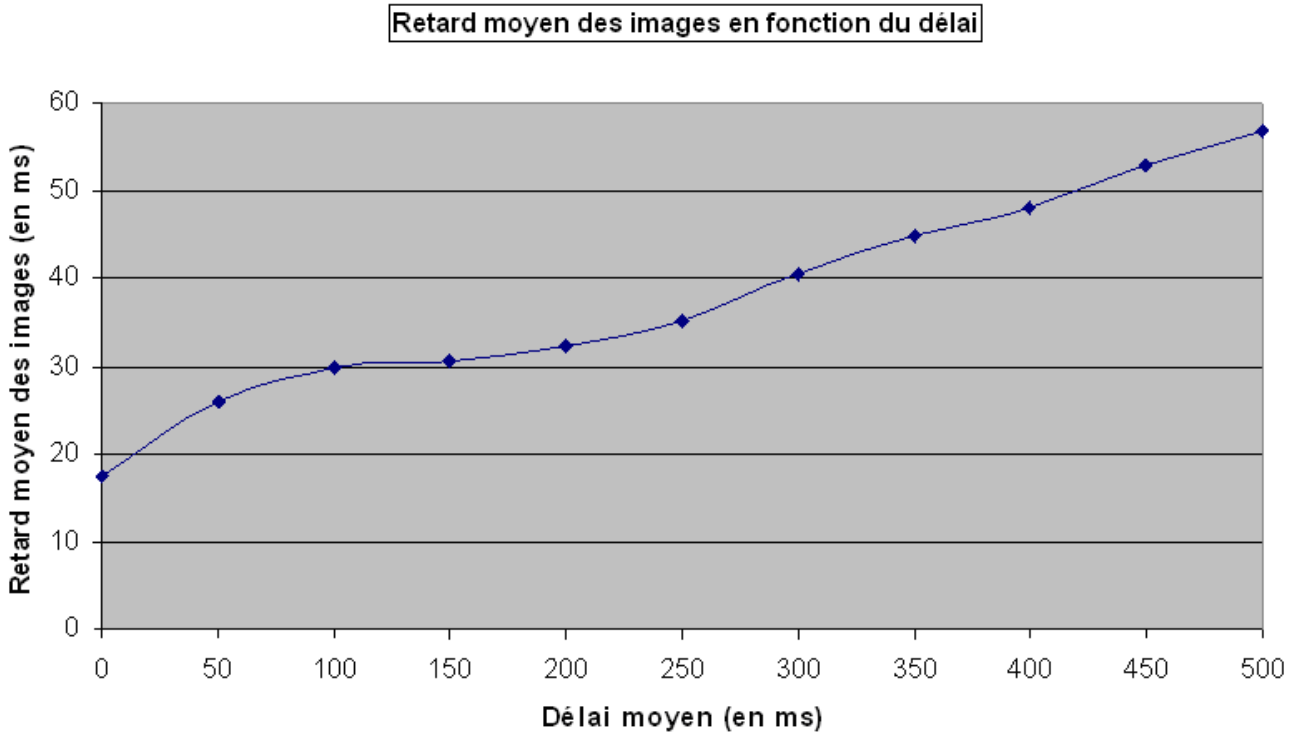


FIG. 4.4 – Le retard des images en fonction du délai introduit

Si on se réfère au tableau 4.3, la fluidité du jeu est encore correcte jusqu'à des délais de l'ordre de 200 à 250ms. Au dela, cela commence à poser bon nombre de problèmes au niveau de la fluidité et du lag.

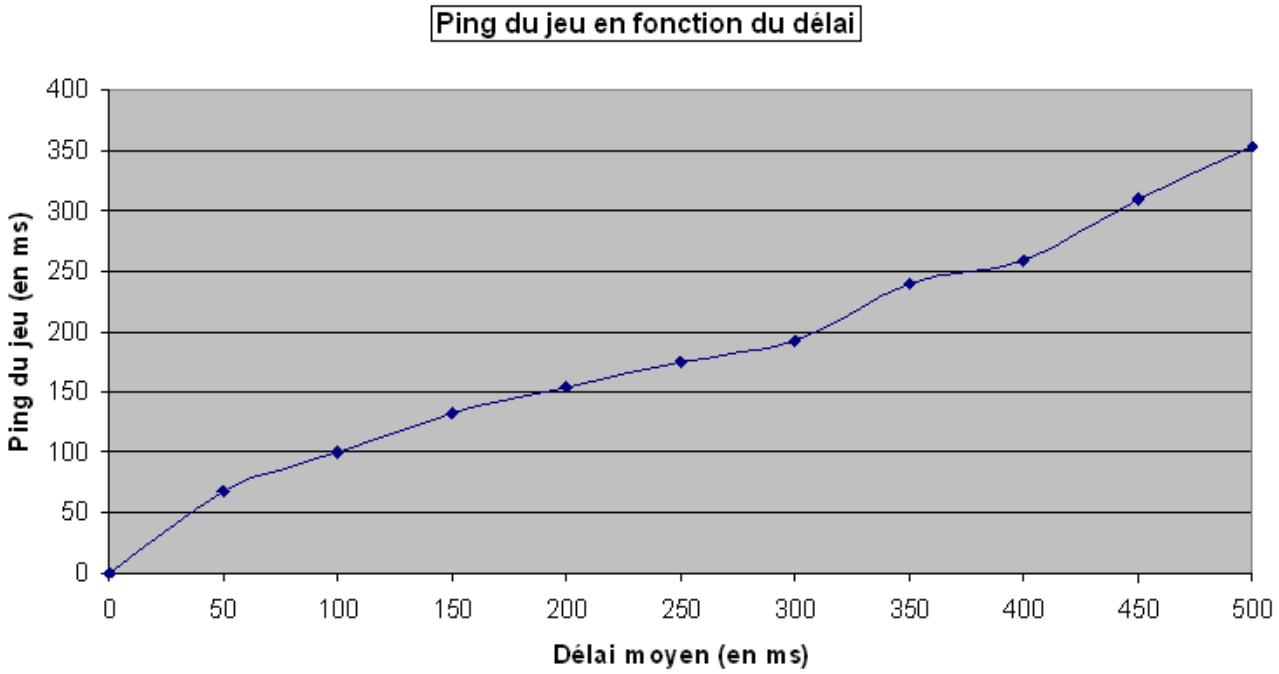


FIG. 4.5 – Le ping du jeu en fonction du délai introduit

Cette courbe pourrait paraître inutile, puisque l'on trace le ping du jeu en fonction du

ping au niveau IP. Cependant, il faut savoir que tout paquet ayant plus de 600ms de délai est considéré comme une perte pour le jeu Quake III. Le jeu fonctionne comme cela puisqu'il est préférable d'avoir une perte importante (nous allons le voir dans les sections suivantes) plutôt que du retard.
Ainsi, le ping du jeu est forcément plus faible que le ping niveau réseau, puisque le ping n'est bien sûr calculé que sur les paquets qui ont été considérés comme reçus.

En ce basant sur notre tableau de notation, on considère qu'un délai moyen au niveau réseau commence à être vraiment gênant à partir de 200 à 250ms. En cumulant ce paramètre et celui de fluidité évoqué ci-dessus, **on conclut donc que le réseau ne doit pas introduire des délais aller/retour supérieurs à 200 voire 250ms au maximum pour une jouabilité correcte.**

Voyons à présent ce qu'il en est de la gigue.

b) La gigue

Pour faire varier la gigue, nous avons simplement augmenté l'écart-type des délais. La moyenne des délais quant à elle a été fixée à 200ms, une zone tangente comme nous venons de le voir. Le taux de perte reste quant à lui fixé à 20%, et la bande passante limitée à 250Kbit/s. Voici les résultats :

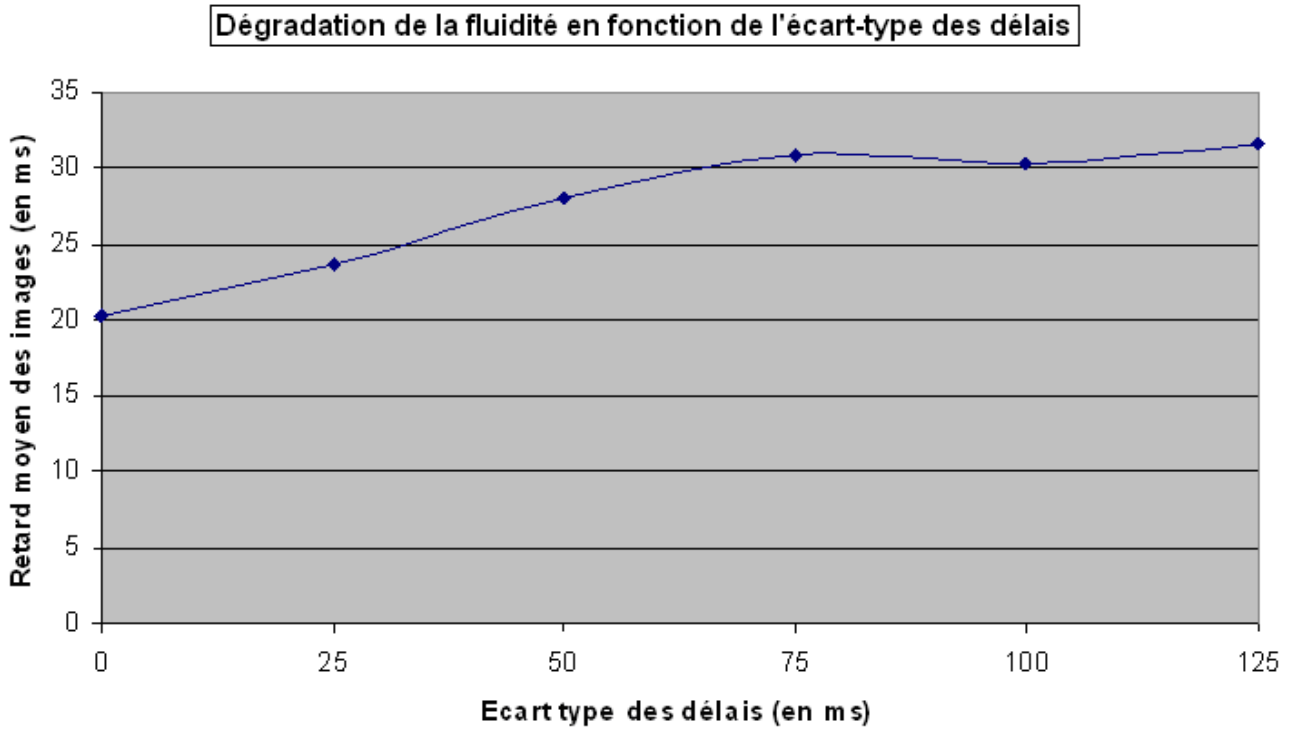


FIG. 4.6 – Le ping du jeu en fonction de l'écart-type des délais

Il n'y avait ici pas d'intérêt à faire une courbe en mesurant le ping, puisque le délai moyen était constant. En revanche, il était intéressant de voir la différence de fluidité dans le jeu en fonction d'un écart-type des délais variable. Comme on le constate sur cette courbe, la gigue introduite apporte des dégradations, mais celles-ci sont tout de même mineures par rapport à d'autres dégradations comme le délai par exemple. En effet, d'un écart-type de 0ms jusqu'à 125ms, on passe d'une fluidité parfaite à une fluidité moyenne et acceptable. Cela confirme donc

ce que nous avons vu dans la bibliographie, à savoir que **la gigue n'est pas le critère le plus critique pour une bonne jouabilité**.

c) La bande passante

Pour nos tests vis-à-vis de la bande passante, nous avons fixé le délai moyen à 150ms, et le taux de perte à 20%. Nous avons alors fait varier la bande passante disponible, voici les résultats.

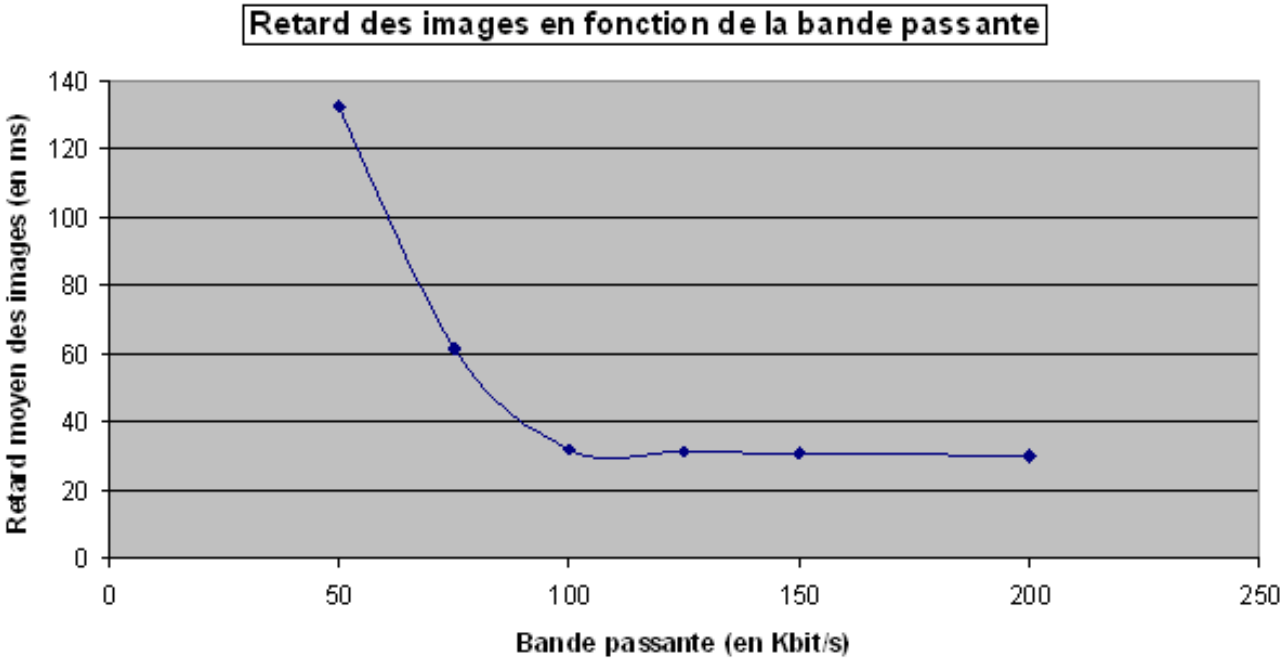


FIG. 4.7 – Le retard des images en fonction de la bande passante

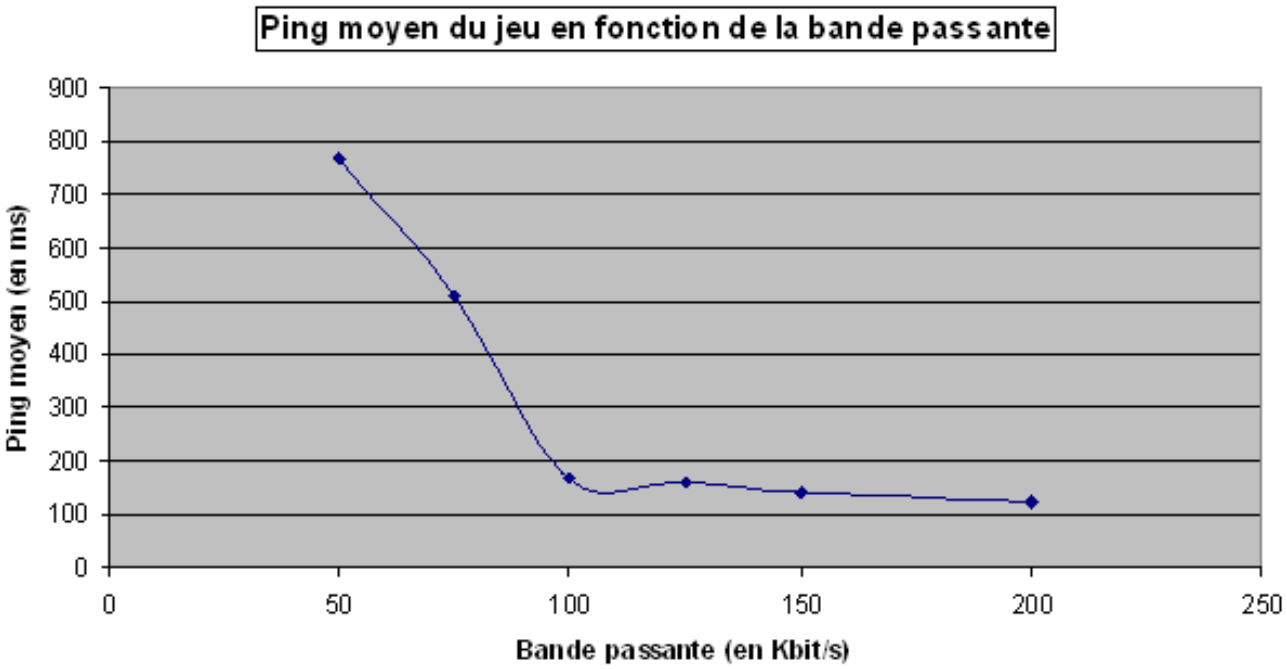


FIG. 4.8 – Le ping moyen du jeu en fonction de la bande passante

On constate ici une limite très nette entre 100Kbit/s et en dessous. A partir de 100, la jouabilité est stable et assez bonne, en revanche en dessous on a très vite une saturation et une très forte dégradation du service. Cela est bien évidemment dû aux caractéristiques réseau du jeu, qui envoie de façon très régulière toujours la même quantité d'information.

A noter toutefois que dans certaines autres configurations (taux de perte différent, délai plus faible ou moins faible...) on peut avoir une mauvaise jouabilité pour des bandes passantes de plus de 250Kbit/s.

Ainsi, le jeu nécessite un minimum de 100Kbit/s, mais une bande passante de 300Kbit/s et au dessus est parfaite en toutes circonstances.

Passons maintenant au dernier critère pris en compte : le taux de perte.

d) Le taux de perte

Pour ces expérimentations, nous avons introduit un délai de 150ms avec toujours cet écart-type relatif de 0,667, ainsi qu'une bande passante de 250Kbit/s. Nous avons simplement fait varier le taux de perte, et voici nos résultats :

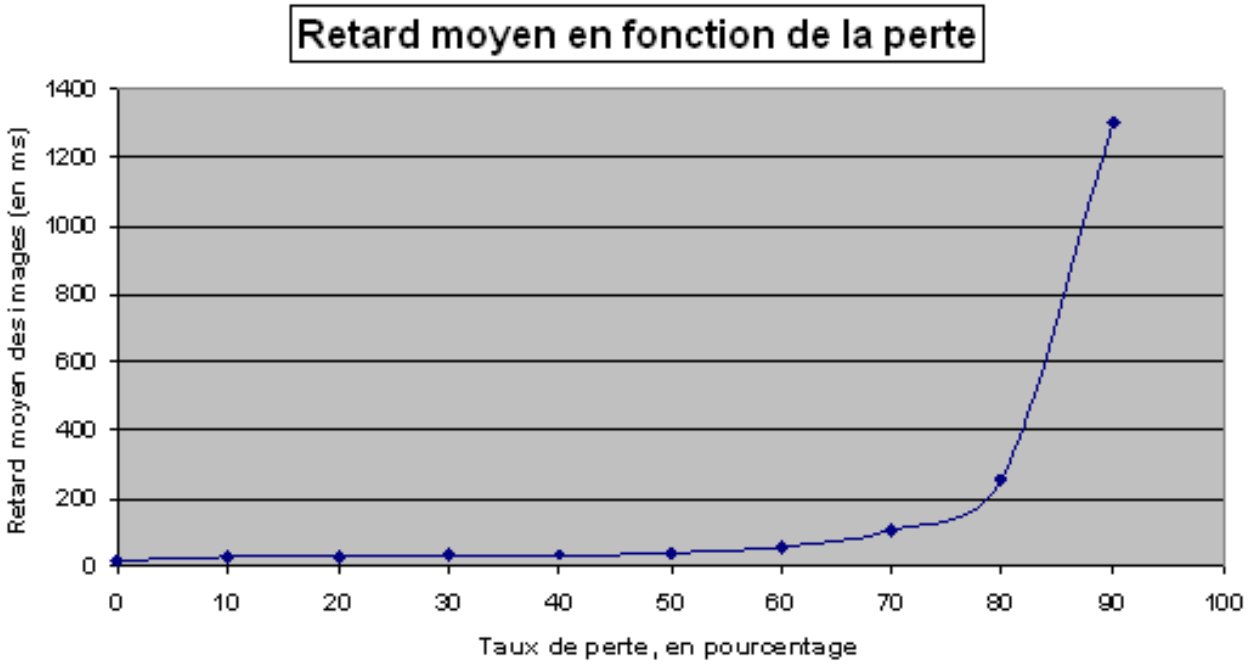


FIG. 4.9 – Le retard des images en fonction de la perte

Sur cette première courbe, on constate que le jeu commence à être réellement dégradé à partir de 60 à 70% de perte de paquet, ce qui est déjà très élevé. Cela confirme ce que nous avions vu, à savoir qu'il est préférable qu'un paquet soit perdu plutôt qu'il n'arrive en retard. Perceptuellement, les dégradations commencent à se faire sentir entre 30 et 40% de perte, ce qui là encore va dans le sens de ce que nous avions vu lors de nos recherches bibliographiques : la plupart des jeux permettent en effet des pertes de 35% sans dégradation visible.
A partir de 70% de perte, en revanche la courbe monte très rapidement jusqu'à atteindre des niveaux totalement injouables.
Voyons maintenant ce qu'il en est vis-à-vis du ping du jeu en fonction du taux de perte :

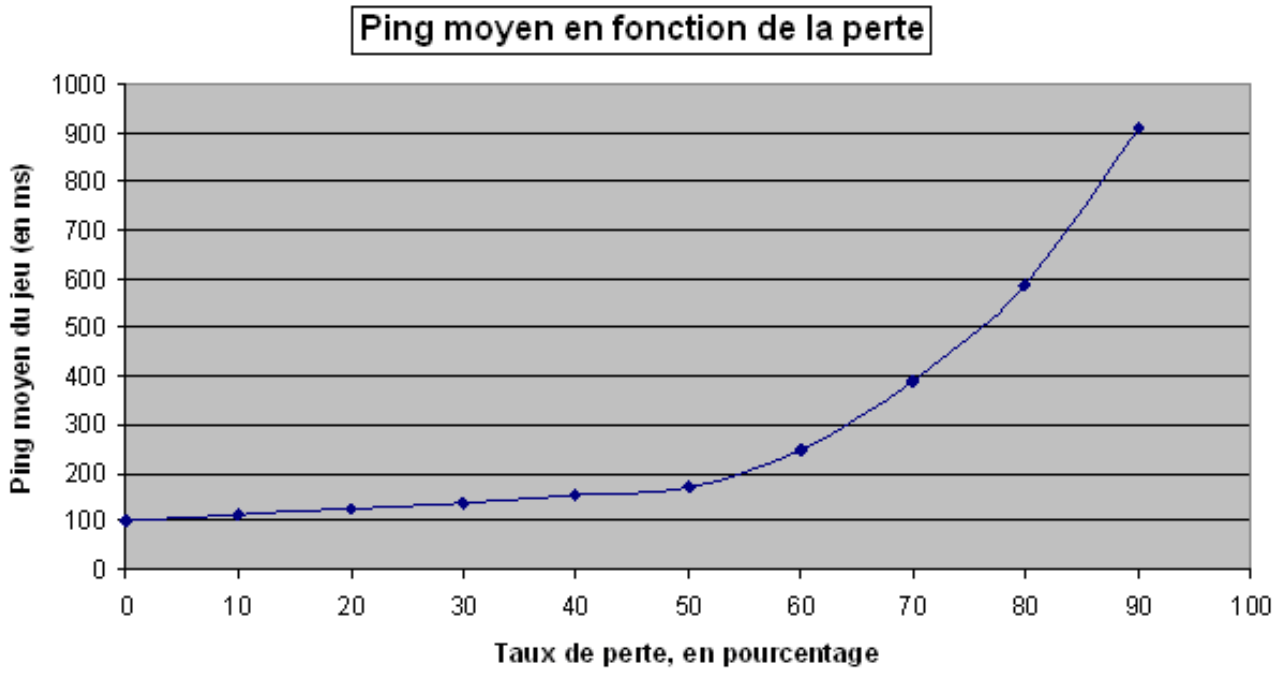


FIG. 4.10 – Le ping moyen du jeu en fonction de la perte

Cette nouvelle courbe affirme ce que nous avions vu sur la précédente, peut-être de façon légèrement moins marquée. A partir de 60% de perte, on a des délais aller/retour de plus de 250ms, ce qui est trop pour une jouabilité correcte.

En conclusion, Quake III Arena permet des pertes de paquet pouvant aller jusqu'à 60%, mais pas plus.

e) Synthèse

Maintenant que nous avons vu les différents résultats suivant le critère, résumons et synthétisons-les. Le jeu vidéo nécessite :

- **Un délai allant jusqu'à 250 voire 300ms**
- **Une bande passante supérieure à 100Kbit/s, idéalement supérieure à 300Kbit/s**
- **Une gigue faible mais ce critère n'est pas primordial**
- **Une perte de paquet inférieure à 60%**

Maintenant, notre but est de savoir si les réseaux ad-hoc permettront de respecter ces contraintes là ou pas. Nous entrons dans la seconde partie de l'étude, basée sur de la simulation.

4.3.2 GloMoSim et le réseau ad-hoc

Maintenant que nous connaissons les besoins des jeux-vidéos en terme de qualité de réseau, nous allons voir ce que les réseaux ad-hoc peuvent fournir à ceux-ci.

La première expérience réalisée

Pour cette expérimentation, nous avons réalisé toutes les étapes décrites au chapitre 4.2. Les paramètres que nous avons pris en entrée sont les suivants :

- Terrain de taille 300m x 300m
- Les noeuds ont des portées de 120m (ce qui est faible, mais correspond aux capacités actuelles des consoles)
- Le noeud serveur envoie vers chacun de ces client du trafic CBR, avec une taille de paquet de 80 octets, cela toutes les 50ms. Cela correspond totalement à ce que nous avions relevé au chapitre 2.3.2
- Les noeuds client envoient au serveur des paquets de taille 80 octets toutes les 20ms. Comme ci-dessus, cela correspond à ce que nous avons mesuré dans la réalité du jeu Quake III
- Sur cette expérience, les noeuds sont immobiles
- Enfin, voici le placement que nous avons utilisé :

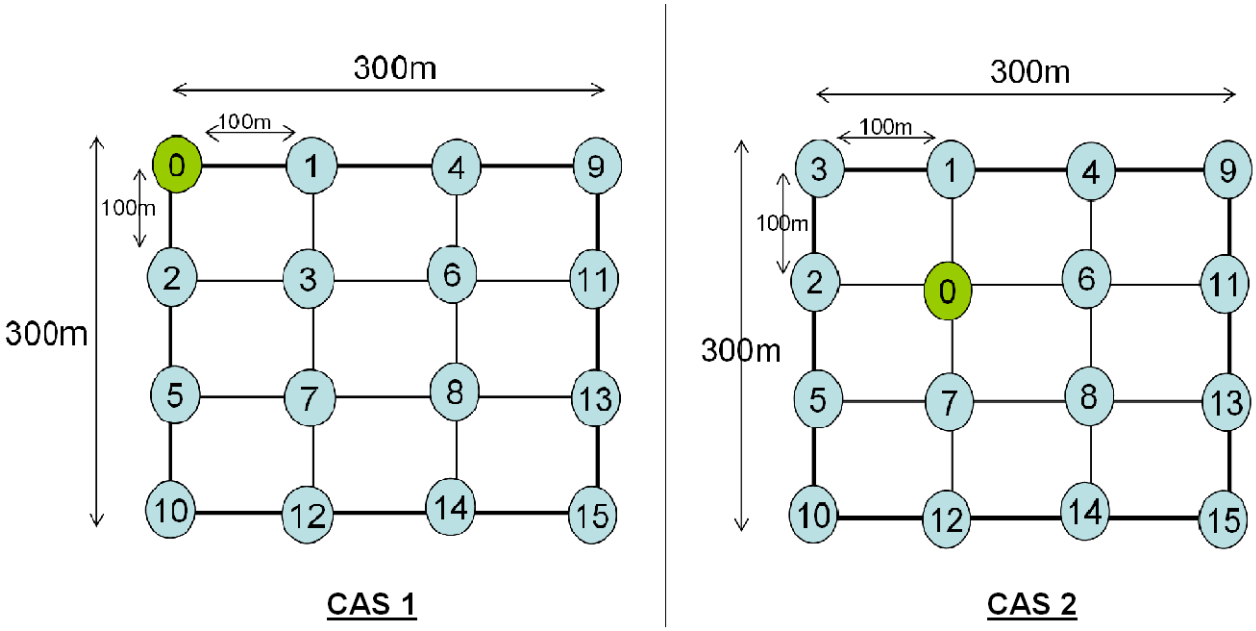


FIG. 4.11 – La configuration spaciale des noeuds

Le noeud 0 en vert constitue le noeud serveur. Lors de futures expérimentations, nous feront varier l'emplacement de celui-ci.
Lorsque nous n'utilisions que 3 noeuds, ceux-ci étaient donc aux emplacements du noeud 0, 1 et 2. En ajoutant un 4ème noeud, celui-ci a été placé à l'emplacement 3... Et ainsi de suite jusqu'au 16ème noeud. Nous avons réalisé deux séries d'expériences, une où le serveur est à une extrémité de la carte, et l'autre où il est quasiment dans le centre de la carte.

Nous savons donc maintenant ce que nous avons passé en paramètre du simulateur, voyons maintenant les résultats.

Les résultats

Voici les deux principaux graphiques que nous avons pu tracer à la suite de cette expérimentation :

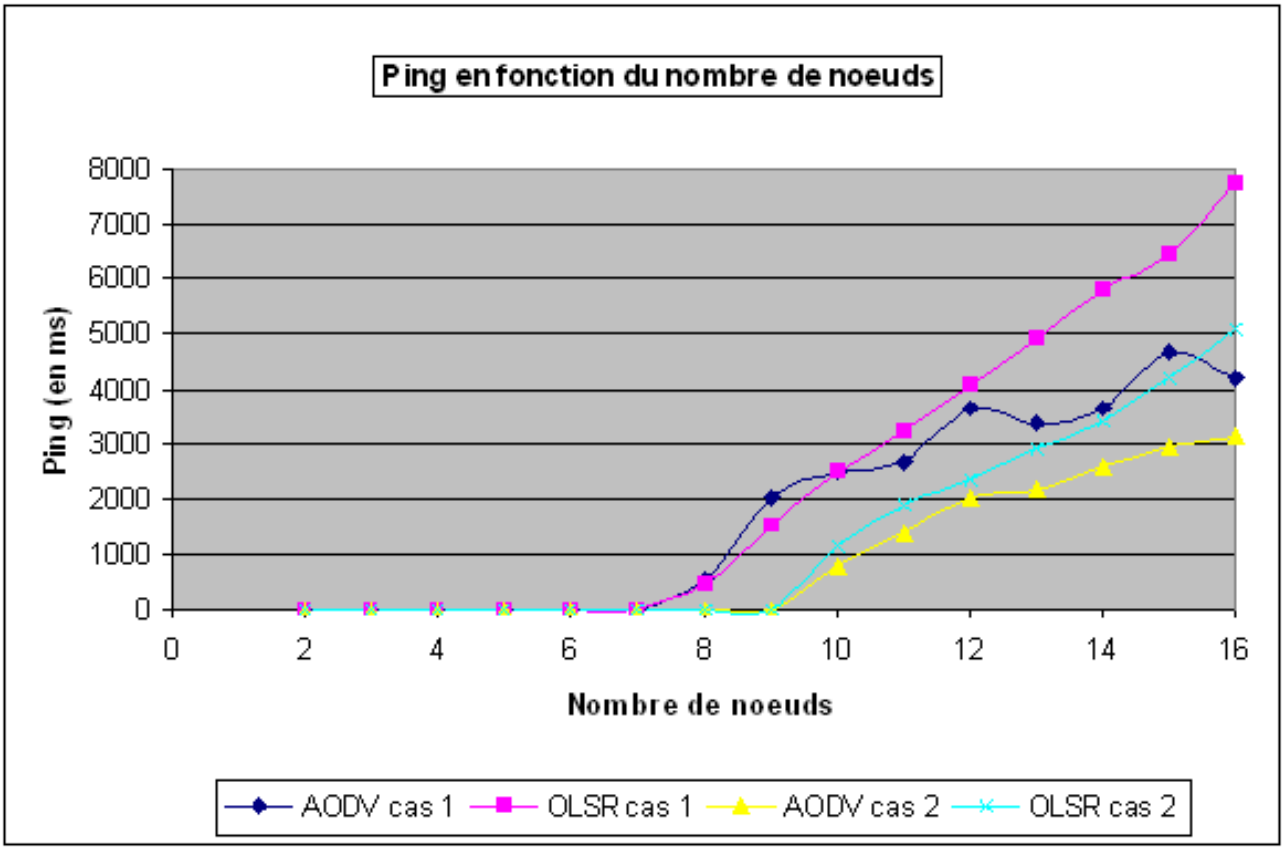


FIG. 4.12 – Le délai aller/retour moyen entre client et serveur en fonction du nombre de noeuds

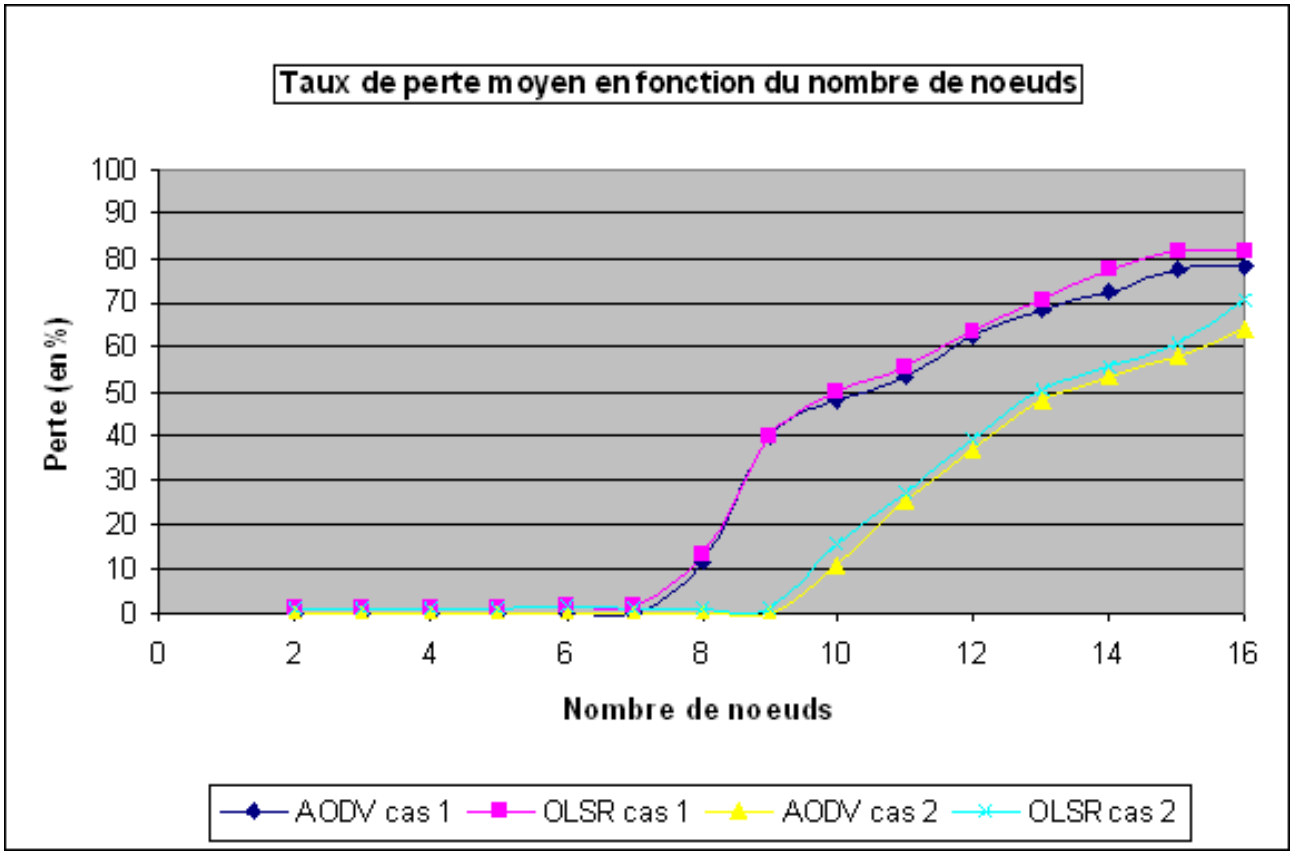


FIG. 4.13 – Le taux de perte moyen entre client et serveur en fonction du nombre de noeuds

La première chose que l'on constate à la vue de ses résultats, c'est que l'emplacement du serveur est stratégique pour le bon fonctionnement du jeu en réseau. En effet, les courbes bleues et violettes sont assez nettement au dessus des courbes jaunes et cyans.

Vis à vis de la courbe 4.12, on constate que l'arrivée du 8ème noeud bouleverse complètement le réseau et implique une très grande augmentation des délais dans le premier cas. En regardant la figure 4.11, on remarque dans cette première configuration qu'un huitième noeud implique désormais des communications à trois sauts, alors qu'elles étaient de 3 au maximum avec un nombre de noeuds inférieur. On peut donc supposer que les réseaux ad-hoc multisauts ne permettent pas de jouer au jeux-vidéos lorsqu'il y a 3 sauts ou plus à faire.

En se référant aux résultats que nous avons obtenu au chapitre précédent, la qualité de jeu sera correcte si le délai aller/retour est inférieur à 250ms, et le taux de perte inférieur à 60%. Pour les courbes bleues et violettes, on remplit la première condition jusqu'à 7noeuds, et la seconde jusqu'à 10 voire 11 noeuds. Cela confirme encore une fois que le plus gros soucis des jeux vidéos est bien d'avoir un temps de latence réduit, et c'est ce qui est le plus difficile à fournir pour les réseaux ad-hoc. Enfin, on remarque que les deux algorithmes de routage utilisés que sont OLSR et AODV, pourtant issus de familles d'algorithmes différents, ont des performances assez proches.

En ce qui concerne l'expérience 2 où le serveur était placé de façon plus centrale sur la carte, les résultats sont légèrement meilleurs. En effet, vis-à-vis du délai on peut désormais jouer jusqu'à 9 joueurs au lieu de 7, et vis-à-vis de la perte 13 à 14 joueurs au lieu de 10 à 11. Enfin, on constate une fois de plus que les dégradations surviennent très rapidement dès lors que l'on a 3 sauts ou plus à faire.

La seconde expérience réalisée

Pour cette nouvelle expérience, notre but était de voir l'impact de la mobilité du serveur sur la fluidité du jeu. Nous sommes donc parti dans une configuration identique à celle du cas 2 de la figure 4.11, et nous avons fait se déplacer le serveur de manière aléatoire sur la carte à une vitesse de 5 mètres par seconde (18kms/h), soit une vitesse équivalente à un homme qui court à vitesse modérée. Tous les autres paramètres sont identiques à la première expérience (temps de simulation, taille du terrain etc), mis à part que l'expérience a été reconduite 10 fois par point au lieu de 20.

Pour avoir un point de comparaison, nous avons donc incorporé sur les graphiques qui suivent les résultats que nous avions obtenu précédemment, à savoir les courbes de couleur cyan et jaune.

Voici les résultats que nous avons obtenu :

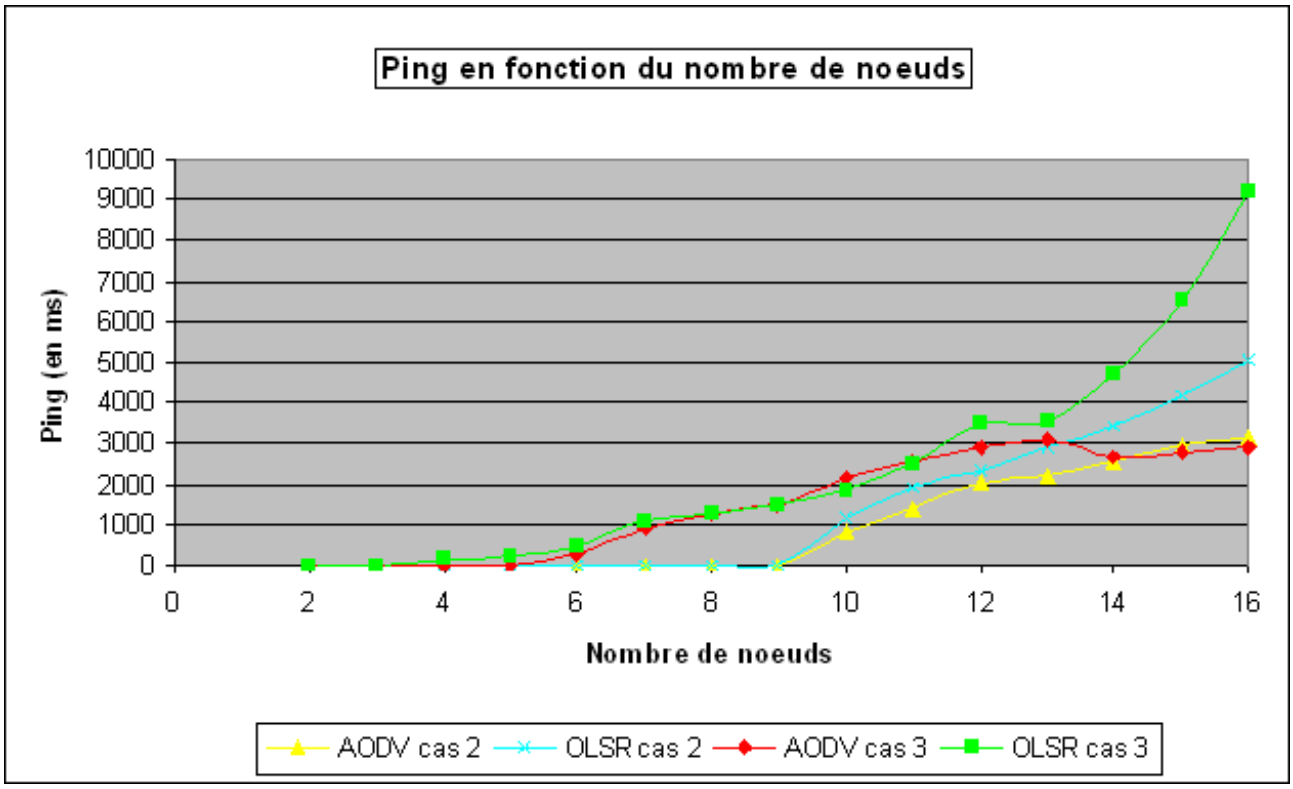


FIG. 4.14 – Seconde expérience : le délai aller/retour moyen entre client et serveur en fonction du nombre de noeuds

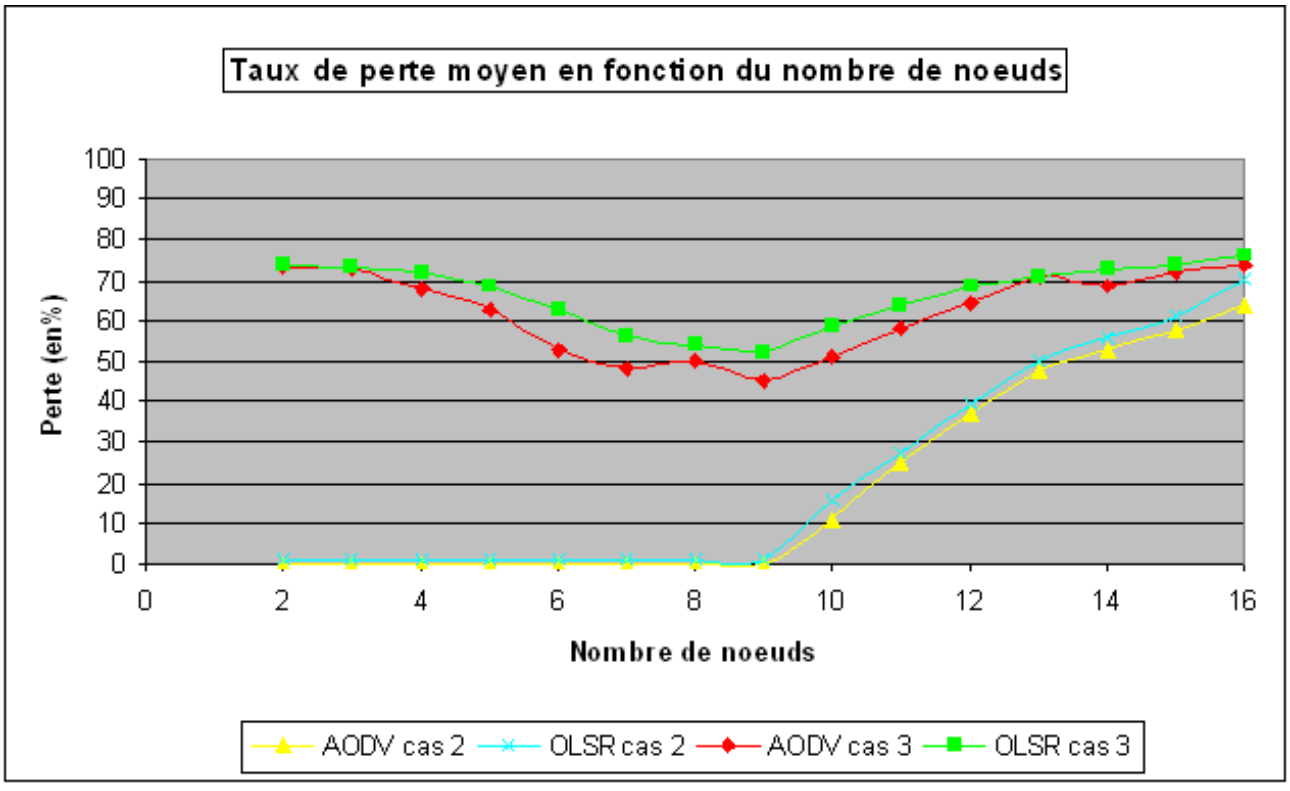


FIG. 4.15 – Seconde expérience : le taux de perte moyen entre client et serveur en fonction du nombre de noeuds

Concernant la courbe du délai, lors de cette expérience on constate que les algorithmes ne donnent pas les mêmes résultats. En effet, lorsque le nombre de noeud devient important, le

protocole proactif OLSR est bien moins bon (ou plutôt bien plus mauvais) que l'algorithme réactif AODV. En ce qui concerne la perte en revanche les deux algorithmes donnent des résultats proches.

Avec cette mobilité, vis-à-vis du délai il est possible de jouer jusqu'à 5 voire 6 joueurs au maximum, et vis-à-vis de la perte il n'est pour ainsi dire pas possible de jouer. A ce propos, on constate d'ailleurs que la perte est très importante pour un nombre de noeuds faible, qu'elle diminue pour un nombre de noeuds moyen, et qu'elle augmente à nouveau pour un grand nombre de noeuds.

D'ici la fin du stage, nous essaierons probablement de voir comment se comporte le réseau en fonction de la vitesse de déplacement du serveur. Ici, nous avons pris une limite maximale de déplacement pour un piéton pour voir si le déplacement aurait dans ce cas une incidence, manifestement oui, donc peut-être d'autres expériences seront menées à ce propos.

En conclusion de toutes ces expériences réalisées :

- **L'emplacement du serveur doit être choisi de façon intelligente**
- **Concernant le critère de délai et sans mobilité, on ne pourra jouer ici qu'avec un maximum de 7 joueurs, voire 9 dans le meilleur des cas**
- **Concernant le taux de perte et sans mobilité, un nombre maximal de 10 à 11 joueurs peut être permis, jusqu'à 13 ou 14 au meilleur des cas**
- **Le choix de l'algorithme de routage ne semble généralement pas avoir une grande importance sur les résultats**
- **La mobilité du serveur, même modérée, a un fort impact sur le réseau et par voie de conséquence sur le jeu**

4.3.3 Synthèse

La figure 3.3 et la liaison simulation/émulation

Originellement, notre but était de suivre totalement le schéma de la figure 3.3. Ainsi, nous comptions mettre les résultats de simulation GloMoSim (vis-à-vis de la perte et du délai notamment) en entrée de Dummynet, et voir comment réagissait le jeu aux dégradations d'un réseau ad-hoc simulé. Cependant, compte-tenu des résultats obtenus, notamment au niveau des figures 4.12 et 4.13, cela n'a pas été réellement possible, ou tout du moins n'aurait pas eu grand intérêt. En effet, jusqu'à 7 noeuds on constate aucune dégradation réseau, ce qui nous laisse supposer que la jouabilité sera parfaite. A l'inverse, il en est de même à partir de 9 noeuds où on se doute qu'il sera strictement impossible de jouer. Les courbes issues de ce mélange d'émulation et de simulation nous aurait donc fourni des courbes en forme d'escalier à une marche, autrement dit n'auraient pas eu un grand intérêt.

Cependant, nous avons tenu à réaliser cette expérience, pour les noeuds qui sont à la limite de la jouabilité, ce qui est le cas pour 8 noeuds de la figure 4.12. Nous avons donc incorporé les résultats obtenus pour les courbes violettes et bleues dans l'émulateur, et ainsi observé l'impact direct du réseau sur le jeu. En suivant le tableau de notation 4.3, nous avons dans cette configuration **une note de qualité égale à 3/10** pour l'algorithme **OLSR** (ping de 235ms et retard des images de 49ms), et de **1/10** pour l'algorithme **AODV** (ping de 298ms et retard des images de 58ms).

Conclusion sur les expériences menées

Grâce à toutes ces expériences, nous savons donc tout d'abord ce que nécessite le jeu Quake III Arena, et plus généralement le jeu vidéo. Cette généralisation est permise car Quake III Arena est un FPS, et il fait donc partie du type de jeu le plus affecté par un réseau de mauvaise qualité. En conséquence, si ce jeu supporte les conditions du réseau, alors normalement tous les autres jeux le pourront également.

En parallèle, nous savons dorénavant que les réseaux ad-hoc, au delà d'un certain nombre de noeuds, ont encore du mal à fournir une qualité suffisante aux besoins des jeux vidéos.

Les perspectives possibles

Pour améliorer la qualité du jeu vidéo sur les réseaux ad-hoc, plusieurs pistes sont possibles. Tout d'abord, **modifier les algorithmes de routage existants**. En effet, certains de ces algorithmes sont paramétrables, et sont le plus souvent configurés pour avoir un minimum de perte dans le réseau, quitte à faire augmenter le délai. Or, nous avons vu que notre principal souci était le délai.

Une autre possibilité est tout simplement de **créer un nouvel algorithme de routage en mode ad-hoc**, qui serait plus adapté au jeu vidéo ; en pratique, cela ne sera pas réalisable en un mois.

Il est également possible d'**apporter des améliorations en donnant des recommandations aux joueurs**, notamment en terme de placement du serveur. En effet, il est sûrement préférable que celui-ci soit au centre de la carte plutôt que dans un coin comme ce fût le cas pour nos expériences.

Enfin, **augmenter la portée d'émission des noeuds** permettrait de limiter le nombre de sauts, et ainsi réduire de façon considérable les dégradations liées au routage. A l'heure actuelle, nous sommes en train de trouver des solutions permettant de jouer à quelques joueurs de plus que ce chiffre de 7, mais cela fera l'objet du dernier mois de stage et n'est donc pas encore terminé et exposable dans ce document.

Les difficultés techniques que nous avons rencontrées

Au cours de ce stage, nous avons été heurtés par bon nombre de difficultés techniques que nous n'avons pas toujours su résoudre. C'est là toute la différence avec un stage d'ingénieur dit "classique", en recherche on essaie et ça ne fonctionne malheureusement pas toujours. Voici les points qui ont posés problème :

- **Le problème de la reproductibilité des expériences**. C'est un problème récurrent dans la recherche et que nous avons soulevé dans la partie bibliographie, que nous avons en partie résolu en essayant d'être le plus objectif possible
- **Le problème de l'émulation**. Pour une reproductibilité parfaite, l'émulation (c'est à dire l'utilisation d'un système réel sur les parties applicatives et une simulation des couches basses) aurait été optimale. Malheureusement cela est expérimental, et nous avons eu du mal à trouver la méthode pour réaliser des expériences ayant du sens et étant objectives
- **Les bots côté client**. Cet aspect a été une relative déception pour nous, puisque nous espérions pouvoir implémenter des joueurs virtuels sur chaque client de façon à pouvoir relever le score. Or, le jeu Quake III Arena ne le permet pas. Ce point additionné au

précédent a fait que nous avons renoncé d'utiliser un grand nombre de clients au profit d'un seul

- **Les limitations de la norme Wifi en elle-même** La norme Wifi n'est par nature pas réellement adaptée à la qualité de service. En effet, des mécanismes tels que celui de RTS/CTS de la couche MAC permet l'équité entre les sources... Ce que nous cherchons précisément à éviter en QoS en privilégiant des flux par rapport à d'autres. La couche MAC du Wifi n'est donc à l'heure actuelle pas idéale pour le jeu-vidéo, et nous ne pouvons malheureusement pas intervenir sur ce point, puisqu'il s'agit d'une norme
- **La faiblesse des algorithmes de routage actuels**. Depuis plusieurs années, des dizaines d'algorithmes de routage ad-hoc ont été proposés ; au final, il ne reste que très peu d'élus : OLSR, AODV, DSR, TBRPF, voire ZRP. Ces algorithmes sont complexes et à l'heure actuelle il est assez difficile de trouver une algorithme meilleur que ceux-ci

Ainsi, il est très difficile d'apporter des solutions réellement novatrices en seulement 6 mois pour permettre une meilleure jouabilité sur ces technologies. En effet, nous sommes à la fois bloqués par la norme Wifi qui est imposée, et à la fois par le fait que les algorithmes de routage actuels bien qu'optimum satisfont difficilement les contraintes réseaux des jeux-vidéos et du temps réel. Nous avons donc conclu de tous ces points qu'une thèse de 3 ans sur ce sujet serait un minimum pour apporter de nouvelles solutions, un stage de 6 mois ne pouvait être qu'un prémice.

Nous en avons désormais fini avec cette partie réalisation, nous allons désormais nous intéresser à la forme du projet plutôt qu'au fond, à travers la gestion de projet qui a été mise en place ainsi qu'un bilan sur les compétences acquises.

Chapitre 5

Gestion de projet

5.1 Planning prévisionnel

Voici le planning prévisionnel (ou diagramme de Gantt) que nous avons émis pour la gestion de ce projet :

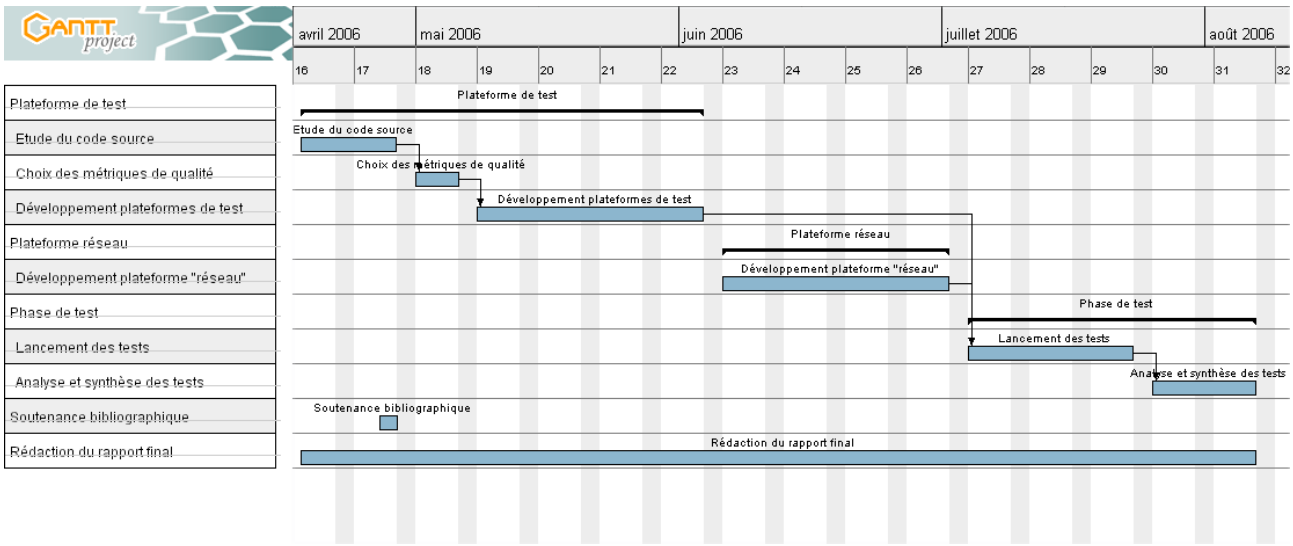


FIG. 5.1 – Le planning prévisionnel

Ce diagramme a été émis au moment de la modélisation des actions à mener, et donc juste avant la partie réalisation du projet. Initialement, on constate trois grandes phases : le développement des plateformes de test, le développement de la plateforme réseau, et la phase de test. Nous avions cependant un doute concernant la phase de développement sur l'aspect réseau, car nous ne savions pas encore si nous opterions pour de l'émulation ou pour l'utilisation d'un système réel.

5.2 Planning effectif

Voici maintenant le planning que nous avons réellement tenu :

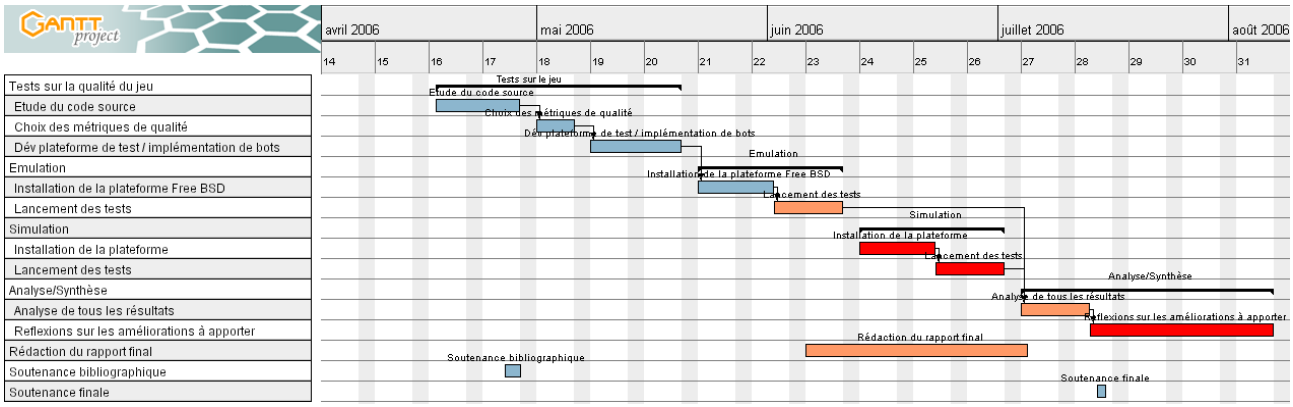


FIG. 5.2 – Le planning tenu

Les phases en oranges ou rouges sont celles qui sont différentes par rapport à notre prévision. Celles qui sont orangées correspondent aux étapes que nous avions prévues mais que se sont retrouvées modifiées, et les étapes en rouge sont celles qui ont été ajoutées et que nous n'avions pas prévues.

Tout d'abord, nous avions initialement prévu de lancer tous les tests à l'approche de la fin du stage, mais il en a été autrement. En effet, nous avons préféré lancer les tests relatifs aux jeux vidéos d'une part et aux réseaux ad-hoc d'autre part directement après avoir monté chacune des deux expérimentations.

La différence majeure qui existe entre le diagramme prévisionnel et le diagramme effectif est la partie simulation. En effet, nous ne l'avions pas prévue, mais elle s'est imposée à nous suite à plusieurs imprévus. A l'origine, nous comptions utiliser une plateforme d'émulation qui prendrait directement en entrée les paramètres d'un réseau ad-hoc, mais ce type de plateforme n'existe pas encore réellement et se trouve encore être au stade expérimental. Ensuite, nous voulions faire participer un grand nombre de joueurs via l'utilisation de joueurs virtuels (bots) sur chacun des clients. Ainsi, nous aurions pu prendre en compte des critères comme le score obtenu. Simplement, sur Quake III Arena il n'est pas possible d'implémenter des bots côté client, il aurait fallu un effort de développement très important et ce n'était pas tout à fait notre propos ici.

Nous avons donc décidé de la chose suivante : nous utiliserons un client et un serveur, et nous regarderons les dégradations sur un seul client. Ces dégradations seront introduites par Dummynet. Au lieu que cet outil de dégradation prenne directement en entrée les paramètres d'un réseau ad-hoc (nombre de noeuds etc), nous avons utilisé un simulateur qui prenait en entrée ces paramètres, nous donnait en sortie des taux de perte, des délais... Que nous pouvions ensuite injecter dans l'émulateur. Ainsi les deux grosses parties de notre étude ont été l'aspect émulation d'une part et l'aspect simulation d'autre part.

Une autre différence majeure par rapport à notre planning initial est que nous avons eu de bons résultats plus rapidement que prévu, ce qui nous permet pour le dernier mois de stage de chercher des pistes d'amélioration pour la jouabilité des jeux sur ces réseaux ad-hoc. Ces pistes ont été évoquées à la fin du chapitre consacré à la réalisation, pour l'instant nous n'avons que des pistes. Ce travail fera donc l'objet de ce dernier mois de stage.

Enfin, la rédaction du rapport de stage a été plus courte que prévue, il nous a d'abord fallu savoir complètement vers quel type d'expérimentations nous allions nous orienter.

Globalement, la gestion de projet a été plutôt réussie, puisque nous avons de bons résultats nous permettant d'apporter éventuellement des solutions sur cette problématique.

Pour conclure ce rapport, passons désormais au bilan du stage.

Chapitre 6

Bilan

Ce stage de six mois se terminant, il est bon de faire une analyse rétrospective retraçant les points forts, mais aussi les points à améliorer.

6.1 Les contraintes scolaires

Ce stage de six mois avait double vocation : valider à la fois le diplôme d'ingénieur en informatique, mais aussi le master recherche ALD (Architectures Logicielles Distribuées). Ainsi, en travaillant chez France Telecom R&D, je remplis parfaitement ces deux contraintes en travaillant sur des aspects ingénierie dans une grande entreprise, et d'autres purement de recherche. Ce présent rapport a été présenté sous un angle ingénierie, mais il sera également aisé de le présenter pour septembre sous un aspect recherche, car ce projet lie vraiment les deux mondes. De ce point de vue, le contrat est rempli.

De plus, avant le stage, des consignes ont été données du côté de l'école. Les voici :

- **Le stage devra être d'un niveau technique approprié (ni trop faible ni trop ambitieux)**. De ce point de vue, ce stage s'est déroulé dans le bon ton, puisque ce sujet est tout à fait expérimental donc loin d'être trivial, mais il reflète des aspects d'ingénierie accessibles pour quelqu'un qui sort d'une spécialité réseaux.
- **Le stage devra traîter des technologies intéressantes et valorisantes pour votre cursus**. Les réseaux ad-hoc font partie de la norme Wifi, qui connait le succès que l'on sait à l'heure actuelle. De plus, ces technologies sont réellement innovantes et amenées à être développées sur bon nombre de consoles portables, comme c'est d'ailleurs déjà le cas sur la Nintendo DS ou la Sony PSP.
- **Le stage ne devra pas être composé de petits sous-projets peu liés entre eux**. Mon projet a été le même de bout en bout, j'ai du traiter de la problématique "Jeux-vidéos sur réseaux ad-hoc", de la bibliographie à la réalisation en passant par la conception.
- **Assurer qu'un travail personnel est clairement identifiable**. Tout ce qui est indiqué dans ce rapport est le fruit de mon travail personnel, que ce soit sur les aspects de bibliographie, de modélisation ou de réalisation ; même si bien sûr, chacune de ces étapes a été au préalable validée par mes tuteurs.
- **Si possible vous amener à dialoguer, interroger et solliciter des compétences de personnes occupant des fonctions diverses dans l'entreprise**. De ce point de vue, le laboratoire où je travaille est très compétent sur cette problématique des réseaux ad-hoc. J'ai ainsi pu participer et débattre de ces problèmes en réunion d'équipe, mais aussi me faire épauler par différents thésards et chefs de projet sur différentes thématiques, notamment sur l'aspect simulation avec GloMoSim. De plus, j'ai aussi beaucoup échangé

avec Khaled Boussetta et Nadjib Achir, maîtres de conférences à l'Université de Paris XIII, où je me suis rendu pendant 2 semaines pour dialoguer, apprendre et aussi mettre en place un plan d'étude complet.

Au niveau des conditions posées par l'école, je pense donc que celles-ci ont été parfaitement remplies.

6.2 Les attentes de l'entreprise

L'objectif du stage du côté de l'entreprise était de pouvoir connaître les limites du jeu vidéo sur réseaux ad-hoc, et de savoir en quelle mesure celles-ci pouvaient être repoussées. Cette problématique s'inscrit dans le cadre d'un projet plus vaste que France Telecom vient de lancer, purement dédié aux jeux vidéos sur réseaux ad-hoc ; des thèses seront sûrement proposées dans ce cadre là. En six mois de stage, il n'est bien sûr pas possible d'améliorer considérablement l'existant, en proposant par exemple de nouveaux algorithmes. Néanmoins, un de nos objectifs principaux était de savoir quelle était la limite théorique de jouabilité des jeux vidéos sur ces réseaux, et nous avons déjà des résultats probants. De ce point de vue, je pense que le contrat est en partie rempli.
En s'appuyant sur ces travaux, un thésard pourra donc savoir quels sont les problèmes, et à partir des analyses de ces problèmes pouvoir apporter des solutions concrètes au niveau du réseau pour améliorer significativement les performances. Notre objectif pour cette fin de stage est d'émettre des pistes à explorer, et aussi donner des solutions permettant quelques améliorations sans pour autant révolutionner l'existant.

Ainsi, les objectifs fixés par l'entreprise sont je pense en grande partie remplis, ils ne le sont pas totalement car malheureusement bon nombre d'obstacles inattendus se sont présentés à nous (impossibilité d'utiliser une vraie plateforme d'émulation, limitations de la norme Wifi en matière de qualité de service, ou encore d'implémenter des bots côté client etc) ; mais la recherche c'est aussi ça, on ne peut pas tout trouver et tout réussir, et c'est une des choses que je retiendrai de ce stage.
Des recherches plus poussées dans ce domaine du côté de l'entreprise pourrait donc mener à de nouvelles possibilités pour les éditeurs de jeux vidéos, et je pense que ce stage a été un bon début dans ce domaine où très peu d'études ont été menées jusqu'à présent.

6.3 Mes attentes personnelles

D'un point de vue personnel, j'avais envie bien sûr de répondre aux attentes de l'école, à savoir marier les aspects ingéniérie et recherche. De plus, j'avais envie de réaliser un stage dans le domaine des réseaux, en particulier des réseaux ad-hoc, un domaine que je connaissais déjà bien et que j'avais envie d'approfondir. Le grand intérêt de ce stage a donc été qu'il m'a permis d'améliorer cette compétence Wifi / Réseaux ad-hoc, tout en les appliquant à un cas pratique : les jeux-vidéos. Or, c'est précisément sur ces domaines liés aux réseaux et au multimédia que j'ai envie d'intervenir dans le futur, que ce soit de la voix sur IP, de la télévision numérique ou autres. Ainsi, j'ai donc pu à la fois :

- Evoluer dans une entreprise pionnière en matière d'innovation technologique, leader européen des télécommunications
- Découvrir le monde de l'entreprise au sein d'un centre de recherche de renommée mondiale (rappelons que de ce centre de Lannion sont sortis entre autres le Minitel, le GSM ou encore l'ATM)

- Lier des technologies réseaux et multimédia
- Pouvoir travailler sur des aspects ingénierie et recherche, validant ainsi deux diplômes et me donnant une double compétence

D'un point de vue technique, mes attentes sont donc pleinement comblées.

En ce qui concerne l'aspect humain, le laboratoire où je travaille concentre bon nombre de gens renommés dans ces domaines techniques très pointus. Ainsi, dans les projets menés au sein de notre URD, peut-on être amené à travailler en collaboration avec des chercheurs de renommée mondiale, citons par exemple Guy Pujolle. C'est donc très enrichissant de pouvoir évoluer dans un tel environnement, autant du point de vue technique que humain.

Enfin, l'accueil qui est m'a été fait ici et plus généralement aux stagiaires est bon, et j'ai ainsi pu évoluer pendant six mois dans une structure agréable et sympatique. Je pense donc que ce stage a été très enrichissant à tous les points de vue, et qu'il sera un excellent prémice en vue de mon insertion professionnelle.

6.4 Si c'était à refaire : les points à améliorer

Si je devais recommencer ce stage dans les mêmes conditions, je pense que j'essaierais de modifier quelques points, pour mener le projet à bien. J'ai noté deux principaux points à améliorer.

Tout d'abord, mon inexperience en sortant de l'école a fait que j'ai pu être rétiscent sur quelques points à me lancer dans des projets complexes de grande envergure, comme la modification du code source du jeu Quake III. J'ai ainsi pu apprendre que la recherche était différente de l'ingénierie sur ce point : rien n'est simple, et il ne faut pas refuser la difficulté ; or, de ma formation d'ingénieur j'ai souvent tendance à vouloir rendre d'abord les choses plus simples avant de les traîter. Sur un stage de recherche, cela n'est pas possible : c'est difficile, mais il ne faut pas forcément vouloir éviter les problèmes.

Ensuite, l'école d'ingénieur m'a appris qu'il était très important de bien rendre compte de son travail de de soigner cet aspect "communication". Cependant, notamment sur le rapport bibliographique, j'ai certainement pris trop de temps pour la rédaction et la remise de compte-rendus, alors que cela n'était pas tout à fait utile. Avec du recul, je pense que ce rapport était trop long par rapport à ce que demandait l'école et le master, et à l'avenir j'essaierai d'équilibrer mon temps entre travail effectif et compte-rendu de ce qui a été fait.

Conclusion

Lors de ce rapport, nous avons pu évoquer tous les aspects du stage que j'ai pu effectuer pendant 6 mois, au sein du groupe Orange. Ainsi, nous avons tout d'abord compris comment fonctionnait l'entreprise France Telecom et en particulier ses centres de recherche R&D, avant d'évoquer tour à tour tous les aspects du stage : bibliographie, conception, réalisation, et gestion de projet. Enfin, nous avons émis un bilan retraçant les points forts du stage et les points à améliorer pour la suite.

France Telecom, au sein du site de Lannion, accueille chaque année entre 200 et 250 stagiaires, preuve d'une volonté d'attirer des jeunes tournés vers les technologies de demain. Dans ce cadre, moi comme beaucoup de stagiaires que j'ai pu cotoyer ici tirons un bilan très positif de cette première véritable expérience professionnelle en tant qu'ingénieur débutant.

En conclusion, ce fût donc un stage riche en enseignements avec son lot de difficultés, de réussites, mais surtout un stage qui m'a offert un vrai tremplin vers la vie active et le monde de l'entreprise.

Bibliographie

Papiers de recherche

[1] Mark Claypool, "The effect of latency on user performance in Real-Time Strategy games", Elsevier Computer Networks Special issue on Networking Issues in Entertainment Computing. Volume 49, Issue 1, Pages 52-70. September 2005

[2] Matthias Dick, Oliver Wellnitz, Lars Wolf, "Analysis of factors affecting players' performance and perception in multiplayer games". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[3] Tobias Fritsch, Hartmut Ritter, Jochen Schiller, "The Effect of Latency and Network Limitations on MMORPGs (A Field Study of Everquest2)". In Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[4] Mark Claypool, Kajal Claypool, Feissal Damaa, "The Effects of Frame Rate and Resolution on Users Playing First Person Shooter Games". In Proceedings of ACM/SPIE Multimedia Computing and Networking (MMCN), January 2006

[5] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, Mark Claypool, "The Effects of Loss and Latency on User Performance in Unreal Tournament 2003". In proceedings of ACM Network and System Support for Games Workshop (NetGames), September 2004

[6] Takahiro Yasui, Yutaka Ishibashi, Tomohito Ikedo, "Influences of Network Latency and Packet Loss on Consistency in Networked Racing Games". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[7] Jaecheol Kim, Jaeyoung Choi, Dukhyun Chang, Taekyoung Kwon, Yanghee Choi, Shinlim-dong, Kwanak-gu, Eungsu Yuk, "Traffic Characteristics of a Massively Multi-player Online Role Playing Game". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[8] Chris Chambers, Wu-Chang Feng, "Patch Scheduling for On-line Games". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[9] Mark Claypool, "On the 802.11 Turbulence of Nintendo DS and Sony PSP Hand-held Network Games". In roceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[10] Jon Gretarsson, Feng Li, Mingzhe Li, Ashish Samant, Huahui Wu, Mark Claypool and Robert Kinicki, "Performance Analysis of the Intertwined Effects betweenb Network Layers for 802.11g Transmissions". In proceedings of the 1st ACM Workshop on Wireless Multimedia Networking and Performance Modeling (WMuNeP), october 2005

[11] Sudhir Aggarwal, Hemant Banavar, Sarit Mukherjee, Sampath Rangarajan, "Fairness in Dead Reckoning based Distributed MultiPlayer Games". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[12] Patric Kabus, Wesley W. Terpstra, Mariano Cilia, Alejandro P. Buchmann, "Addressing cheating in distributed MMOGs". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[13] Shinya Yamamoto, Yoshihiro Murata, Keiichi Yasumoto, Minoru Ito, "A Distributed Event Delivery Method with Load Balancing for MMORPG". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[14] Yugo Kaneda, Hitomi Takahashi, Masato Saito, Hiroto Aida, Hideyuki Tokuda, "A challenge for reusing multiplayer online games without modifying binaries". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[15] Steven Gargolinski, Christopher St. Pierre, Mark Claypool, "Game server selection for multiple players". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[16] Sebastian Zander, David Kennedy, Grenville Armitage, "Dissecting server-discovery traffic patterns generated by multiplayer first person shooter games". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[17] Bart De Vleeschauwer, Bruno Van Den Bossche, Tom Verdickt, Filip De Turck, Bart Dhoedt, Piet Demeester, "Dynamic Microcell Assignment for Massively Multiplayer Online Gaming". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[18] Leo Petrak, Olaf Landsiedel, Klaus Wehrle, "Framework for evaluation of networked mobile games". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[19] Swee Ann Tan, William Lau, Allan Loh, "Networked game mobility model for first-person-shooter games". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[20] Li Tang, Jun Li, Jin Zhou, Zhizhi Zhou, Hao Wang, Kai Li, "FreeRank : implementing independent ranking service for multiplayer online games". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[21] Masaki Fujimoto, Yutaka Ishibashi, "Packetization interval of haptic media in networked virtual environments". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[22] Andreas Janecek, Helmut Hlavacs, "Programming Interactive Real-Time Games over WLAN for Pocket PCs with J2ME and .NET CF". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[23] Jeff Yan, Brian Randell, "A systematic classification of cheating in online games". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[24] Angie Chandler, Joe Finney, "On the effects of loose causal consistency in mobile multiplayer games". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

[25] Dirk Budke, K´aroly Farkas, Bernhard Plattner, Oliver Wellnitz, Lars Wolf, "Real-Time Multiplayer Game Support Using QoS Mechanisms in Mobile Ad Hoc Networks". In proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, 2005

articles Internet

[26] http ://fr.wikipedia.org/wiki/Type_de_jeu_vid%C3%A9o

[27] http ://fr.wikipedia.org/wiki/Jeu_vid%C3%A9o

[28] http ://www.pcinpact.com/actu/news/Le_marche_du_jeu_video _console_na_pas_fini_de_croi.htm

[29] http ://www.zdnet.fr/actualites/informatique/0,39040745,392271,00.htm

[30] http ://www.afjv.com/press0407/040702_jeux_video_etude.htm

Documentation sur les réseaux ad-hoc

Ouvrages

[31] Paul Mühlethaler, "802.11 et les réseaux sans fi", éditions Eyrolles, août 2002

[32] Kaldoun Al Agha, "Réseaux sans fils et mobiles", éditions Eyrolles, novembre 2004

[33] Al Agha Pujolle Vivier, "Réseaux mobiles & réseaux sans fil", éditions Eyrolles, septembre 2001

Articles parus sur Internet

[34] http ://fr.wikipedia.org/wiki/WPA

[35] http ://fr.wikipedia.org/wiki/TKIP

[36] http ://www.paris-sansfil.fr/TBRPF

[37] http ://www-lor.int-evry.fr/ anna/DSR_validationv3.pdf

[38] http ://www.faqs.org/rfcs/rfc3561.html, juillet 2003

[39] http ://reseaucitoyen.be/index.php ?AODV

[40] http ://fr.wikipedia.org/wiki/OLSR

[41] http ://www.cs.helsinki.fi/u/kraatika/Courses/IPsem04s/slides/comp.pdf

[42] http ://www-rp.lip6.fr/ blegrand/cours/DESS/CoursReseauxAdHoc.pdf

Documents de recherche au format PDF

[43] Elisabeth M. Belding Royer, Charles E. Perkins, "Evolution and future directions of the ad'hoc on-demand distance-vector routing protocol", 2003.

[44] Claude Chaudet , Isabelle Guérin Lassous, "Routage QoS et réseaux ad-hoc : de l'état de lien à l'état de noeud". Rapport de recherche INRIA n°4700, ISSN 0249-6399. Janvier 2003.

[45] Prasant Mohapatra, Jian Li, And Chao Gui, University of California. "QOS in mobile ad hoc networks IEEE Wireless communications", June 2003

Emulation, Simulation, Création de réseaux ad-hoc

[46] http ://nile.wpi.edu/NS/

[47] http ://www.linuxjournal.com/article/5929

[48] http ://www.linuxdevcenter.com/pub/a/linux/2000/06/22/LinuxAdmin.html

[49] http ://www.marseille-wireless.org/spip/article.php3 ?id_article=38

Annexe B

Les CRD et leurs différents laboratoires

RESA	**Réseaux d'Accès**
BWA	**B**roadband **W**ireless **A**ccess
FACE	**F**réquences, **A**ntennes, **C**EM, **E**nvironnement
GHOA	**G**ateway and **H**ome network **O**ptical **A**ccess
MNA	**M**obile **N**etwork **A**ccess
NET	**N**etwork **E**ngineering **T**ools
RMC	**R**éseaux d'accès **M**ultiservices sur support **C**uivre
SCG	**S**outien **C**ontrôle de **G**estion
SDE	**S**ervices de **D**onnées aux **E**ntreprises
SPO	**S**ervice **P**ilotage des **O**pérations

SIRP	**Services Intégrés, Résidentiels et Personnels**
CLI	**Cli**ents, **U**sages et **P**rospectives de services
SIA	**S**ervices **I**ntégrés **A**synchrones
SIC	**S**ervices **I**ntégrés et de **C**ontenus
SIS	**S**ervices **I**ntégrés **S**ynchrones

TECH	**Technologies**
EASY	**E**nrichissement des services par un **A**ccès **S**imple à l'**Y**nformation
IRIS	**I**mage **R**ichmedia, nouvelles **I**nteractions et hyperlangages
LEI	**L'E**conomie de l'**I**nnovation
ONE	**O**bjets **C**ommunicants, **N**ouvelles interfaces et **E**quipements terminaux
QVP	**Q**ualité et **V**aleur **P**erçue
SSTP	**S**peech and **S**ound **T**echnologies & **P**rocessing
SUSI	**S**ociologie des **U**sages et traitement **S**tatistique de l'**I**nformation

CORE	**Cœur de Réseau**
CPN	Core **P**acket networks for NGN & IMS
MCN	**M**etropolitan & optical Core **N**etworks
M2I	**M**ultime**d**ia networks for non-conversational fixed/mobile services : **I**mage, **I**nternet
M2V	**M**ultime**d**ia networks for conversational fixed/mobile services : **V**oice, **V**isio
NAS	**N**etwork **A**rchitecture center for **S**ervice deployment
NIS	**N**etwork **I**ntegration center for *S*ervie deployment

MAPS	**Middleware & Plate-Formes Avancées**
AMS	**A**rchitecture, Ingéniérie de plates-formes **M**iddleware et **S**ervices Intégrés
DPC	**D**istribution et **P**rotection des **C**ontenus
ECB	**E**lectronic **C**ommerce and **B**anking
EIS	**E**nablers for **I**ntegrated **S**ervices
LIS	**T**est et **I**ntégration de **S**ervices
MDG	**M**anageable **D**evices and **G**ateways
MMC	**M**ultimedia **C**ommunications
NSS	**N**etwork and **S**ervices **S**ecurity

BIZZ	**Services aux Entreprises**
CIL	**C**entre d'**I**ntégration **L**ogicielle (Euralba)
DIAM	**D**ata Services **I**ntegration, **A**dministration and **M**2M
MUSE	**M**arketing, **U**sages et nouveaux **S**ervices à l'**E**ntreprise
NESS	**N**etwork **E**nhanced **S**ervices **S**olutions
PMX	**P**rofessional **M**ultimodal e**X**change services

Annexe C

Fiches de suivi

Cette partie a pour but de présenter toutes les fiches de suivi qui ont pour but d'être validées par les tuteurs. Elles permettent d'avoir une vue globale du travail fourni semaine par semaine.

semaine 6 (du lundi 6 au vendredi 10 février 2006)

Ce qui a été fait

- Installation du poste de travail (y compris linux)
- Téléchargement des codes sources de Quake III Arena
- Lecture, analyse et synthèse complète de 8 documents de recherche sur le sujet. La synthèse est rédigée au propre sur traitement de texte en utilistant généralement une page par article.

Compte rendu de la réunion du 7 février 2005

- Présents : Sidi-Mohammed, Khaled, Najib et Pierre
- Cette réunion avait pour but de mettre en place le projet, définir les objectifs à tenir. Pour résumer le travail à fournir, voici une copie de ce qui a été envoyé par mail à mon tuteur enseignant de l'Ecole :

Objectifs du stage

"Ici, le but sera dans un premier temps de voir l'impact qu'ont les jeux vidéos sur les réseaux (filaires ou radios) ; j'ai une assez grande quantité de documents à lire sur le sujet. La prochaine réunion avec les tuteurs m'amenera donc à synthétiser toutes ces lectures, à les interpréter et à y apporter d'éventuelles remarques. Voici le type d'articles que j'ai à lire : (j'en ai pour l'instant plus de 25, si vous le désirez je peux vous donner des liens vers tous ces articles)

http ://www.research.ibm.com/netgames2005/papers/claypool.pdf

D'ici une bonne dizaine de jours j'espère avoir lu une grande partie voire l'intégralité de ces articles et les avoir résumés.

Ensuite, à partir de là, je devrai installer un jeu sous linux (en l'occurrence Quake III Arena si vous connaissez, dont le code source est disponible gratuitement sur le net). J'essaierai ainsi d'installer le jeu dans un premier temps sur une architecture fixe, et ainsi sniffer le réseau pour voir le trafic obtenu. Ensuite je passerai sur une architecture ad'hoc. Tout cela me prendra un

bon mois et demi à deux mois à temps plein et constituera l'essentiel du rapport bibliographique, avec bien sûr tout ce qui concerne l'aspect ad'hoc.

Dans la seconde partie, on essaiera de voir expérimentalement avec 7 ou 8 PC l'influence des algorithmes de routage ad'hoc sur la jouabilité et la fluidité du jeu. On essaiera de tirer de là un modèle de trafic relativement cohérent pour les jeux vidéos.

Ensuite, sur un simulateur réseau (là ce sera certainement glomosim), on simulera l'utilisation d'un tel jeu sur un réseau ad'hoc avec un serveur de jeu, et ainsi de voir quel serait potentiellement l'algorithme le plus adapté à l'utilisation de ces jeux vidéos. Enfin, la finalité serait de proposer des optimisations à l'algorithme choisi et autres solutions permettant d'améliorer la qualité du "gameplay", à savoir la fluidité du jeu..."

Ce qui est prévu la semaine à venir

- Le travail principal de la semaine prochaine consistera à continuer la phase de lecture bibliographique, en essayant de lire et de synthétiser la plupart des papiers, voire la totalité.

semaine 7 (du lundi 13 au vendredi 17 février 2006)

Ce qui a été fait

- Fin de la lecture des articles bibliographiques ; résumé de ces 24 papiers terminé.
- Prise d'informations sur les forums de discussion concernant Quake et le développement des jeux vidéos en général.
- Démarrage de la phase d'analyse du code source de Quake III Arena ; début d'installation de gcc 2.95.

Ce qui est prévu la semaine à venir

- Lundi, réunion sur l'avancée du travail et sur la première phase de bibliographie.
- Essayer d'installer gcc 2.95 et de compiler Quake.
- Essayer de comprendre la structure du code du jeu.
- Eventuellement, installer le serveur de jeu Quake sur un poste

Problèmes rencontrés

- Comme prévu, l'utilisation de gcc 2.95 et la compilation du code n'est pas évidente. En ce qui concerne gcc, j'ai tenté plusieurs choses. Tout d'abord, j'ai téléchargé le package à l'adresse suivante : http ://www.fr.linuxfromscratch.org/view/blfs-5.1-fr/general/general/gcc2.html, puis suivi les instructions. A la fin du makeinstall, j'ai deux messages d'erreurs, semblant indiquer un problème au niveau des librairies libstdc++.
  J'ai également tenté d'installer cette version de gcc via un package rpm, en allant à l'adresse suivante : http ://rpm.pbone.net/index.php3/stat/4/idpl/217013/com/gcc-2.95.3-1owl.i386.rpm.html
  mais là encore il y a des soucis, il semble me manquer des librairies "exotiques" comme il est indiqué dans la section "requires".
- Je tenterai de régler une partie de ces problèmes dès la semaine prochaine ; ces points seront certainement évoqués lors de la prochaine réunion.

semaine 8 (du lundi 20 au vendredi 24 février 2006)

Compte rendu de réunion du lundi 20 février 2006

Deux principaux sujets de discussion : le rapport rendu le jeudi 16, ainsi que le travail à venir.

- En ce qui concerne le rapport, il a été dit que celui-ci manquait de structuration et de critique personnelle des articles ; ces écueils ont été rectifiés dans une seconde version de celui-ci
- Vis à vis du travail à venir, à court terme il s'agissait de mener les premières expériences sur le jeu Quake III. Il n'était dans un premier temps pas envisagé d'étudier le code, il s'agissait de voir le comportement du réseau vis à vis du jeu. A partir des premiers résultats obtenus, il serait intéressant dans un second temps de parvenir à les expliquer en regardant le code source
- La prochaine réunion aura lieu soit le jeudi 2 mars à 14 heures, soit le lundi 6 mars en fonction de l'état d'avancement. Le but est de lier les lectures réalisées avec des premières expériences concluantes, et d'en discuter à ce moment là

Ce qui a été fait

- Installation d'ubuntu (distribution linux) sur l'ordinateur : plus de problème de GCC, puisque cette distribution permet d'installer de façon automatique gcc 2.95.
- Réunion lundi après-midi
- Suite à la réunion, refonte du rapport (structuration, incorporer des critiques aux travaux effectués). Ebauche du futur rapport bibliographique
- recherche d'outils de supervision de réseau. Test et prise en main de ceux-ci : Ethereal, AirMagnet, CommView, Ntop entre autres
- Installation de cartes Wifi compatibles avec les outils de supervision
- Installation de quake III Arena en demo sur les ordinateurs
- Premières parties en réseau, en Ethernet puis en wifi via un mode ad'hoc point à point (sous Windows)
- Premiers tests, en capturant le trafic avec les différents outils disponibles ; deux machines jouent, et sur le serveur on capte et analyse le trafic qui s'écoule
- Premiers résultats obtenus, début de rédaction de la partie correspondante sur le rapport
- Vendredi après-midi, insertion d'une troisième machine, le LAN fonctionne très bien en ad'hoc à 3. En revanche, soucis pour intercepter les messages qui ne nous sont pas destinés

Ce qui est prévu la semaine à venir

- Continuer les tests sur Quake III en décortiquant ses caractéristiques réseau
- Résumer et rédiger clairement sur un rapport les expérimentations réalisées
- Eventuellement pouvoir donner des explications à ces résultats à partir du code source

Problèmes rencontrés

- Quake III ne fonctionne pas sous linux : l'exécutable de la démo obtenu ne se lance pas. Même résultat sous distribution ubuntu et mandriva
- Incompatibilité entre certains logiciels de supervision et les cartes wifi par défaut sur les postes
- Problème avec Ethereal pour réussir à sniffer le trafic qui ne nous est pas destiné (typiquement 2 machines jouent pendant que la troisième sniffe). A voir la semaine prochaine.

semaine 9 (du lundi 27 février au vendredi 3 mars 2006)

Ce qui a été fait

- Problème concernant le sniffe sur les autres PC réglé grâce à l'outil airodump
- La plupart des incompatibilités de carte Wifi avec les logiciels de supervision a été réglée via l'installation de nouveaux drivers
- Suite de la recherche d'outils adaptés à l'analyse du trafic ; synthèse des outils disponibles
- Les premiers tests sur Quake III sont terminés, cela en suivant un scénario simple décortiquant le jeu en différentes phases
- Analyse des résultats obtenus et rédaction claire de l'expérience sur un rapport
- Suite à la réunion, poursuite de la recherche sur les différents outils de supervision disponibles
- Lecture d'un précédent rapport d'étudiants en lien avec le sujet
- Prise en main de la console Quake III, permettant d'intervenir sur bon nombre de paramètres du jeu

Compte rendu de réunion du jeudi 2 mars 2006

Lors de cette réunion, l'ordre du jour était d'une part la présentation des premiers résultats à travers un powerpoint, et d'autre part discuter de la suite à court terme et à plus long terme du projet.

- A court terme, il a été décidé de continuer l'expérience en faisant varier différents facteurs comme le SNR, et voir l'impact que ces paramètres avaient sur le jeu.
- Pour cela, il est necessaire de trouver un outil permettant de calculer les paramètres comme la latence, le taux de perte ou la gigue
- Après ces expériences, il a été convenu de se focaliser sur le rapport bibliographique, en y ajoutant la partie "ad'hoc".

Ce qui est prévu la semaine à venir

- Continuer la recherche pour avoir des outils calculant les paramètres voulus
- Commencer une nouvelle batterie de tests
- Comprendre le mécanisme de console sous Quake et assimiler les différentes commandes

Problèmes rencontrés

- Difficulté à trouver des outils et méthodes permettant d'analyser simplement les différents paramètres du trafic réseau (latence, gigue...)

semaine 10 (du lundi 6 au vendredi 10 mars 2006)

Ce qui a été fait

- Lecture du rapport des étudiants ayant travaillé sur le sujet
- Reflexion quant à l'étude à mener : quels paramètres prendre en compte ?
- Recherche et obtention de tous les outils nécessaires à l'expérience, notamment vis à vis du ping et du FPS
- Préparation des tests en réalisant un tableau de synthèse à remplir

– Assimilation du mécanisme d'échanges entre client et serveur dans le jeu Quake III (et plus généralement dans les jeux FPS)

Compte rendu de réunion du vendredi 10 mars 2006

Lors de cette réunion, il a principalement été évoqué les résultats des derniers travaux sur les mécanismes d'échange de données dans quake III. En effet, cela s'est avéré primordial pour réellement comprendre l'impact que peut avoir le réseau sur la jouabilité du jeu.
A travers cette réunion, nous nous sommes rendus compte que le FPS (nombre d'images par seconde) n'était pas le seul indicateur valable pour traiter de la qualité et de la jouabilité du jeu, bien au contraire. D'autres paramètres entrent en jeu, comme le nombre de rafraîchissements par seconde de l'information côté client, ou le nombre d'informations par seconde envoyées du serveur vers le client, et du client vers le serveur.
Toute la difficulté qui réside est donc d'avoir un critère cohérent et objectif pouvant caractériser la jouabilité du jeu ; le FPS en est un, mais n'est pas suffisant. Enfin, il est à noter qu'un FPS relativement faible mais stable est préférable pour la jouabilité qu'un FPS qui varie constamment entre bon et excellent.

Il a également été évoqué le billet de recherche à paraître pour la conférence globecom de l'IEEE. Le but est de mener une étude cohérente indiquant l'impact du réseau sur le jeu. Une proposition qui a été faite est de calculer le ping, la gigue, le débit, le nombre d'informations échangées entre client et serveur et le pourcentage de paquets reçus et le nombre moyen de FPS. Après la réunion, il a été question d'introduire aussi la notion d'écart-type des FPS, critère tout aussi pertinent voire même plus pertinent qu'une simple moyenne.

L'objectif est d'obtenir de premiers résultats le plus rapidement possible, et à partir de là pouvoir analyser ces résultats pour en tirer une éventuelle amélioration. Une idée qui a été proposée était de modifier les paramètres réseau du jeu pour améliorer la fluidité du jeu à qualité de réseau égal.
Enfin, il a été convenu de fixer une prochaine réunion rapidement, dès le mardi 14 mars.

Ce qui est prévu la semaine à venir

– Réalisation des expérimentations
– En tirer des conclusions et y apporter d'éventuelles améliorations en vue de la rédaction du billet globecom.

Problèmes rencontrés

– La complexité des échanges du jeu nous ont quelque peu retardé
– Ceci a été accentué par la découverte que le nombre moyen de FPS ne soit pas forcément lié directement à la qualité du jeu

semaine 11 (du lundi 13 au vendredi 17 mars 2006)

Ce qui a été fait

– Réalisation des expérimentations évoquées fin de semaine 10 en outdoor
– Synthèse des résultats obtenus
– Réflexion quant à une nouvelle étude à mener en indoor

- Recherche d'outils compatibles avec nos cartes wifi disponibles pour calculer le SNR
- Réalisation de scripts bash pour lancer périodiquement la commande iwspy, et récupérer les résultats
- Documentation sur gnuplot pour le tracé des résultats donnés par iwspy
- Préparation de la présentation powerpoint en vue de la réunion d'URD du lundi 20 mars 2006

Compte rendu de réunion du mardi 14 mars 2006

Lors de cette réunion, nous avons évoqué nos résultats quant à l'étude menée en extérieur. De cette réunion, nous avons conclu qu'il n'y avait pas le moindre problème pour jouer sur un simple lien ad'hoc en extérieur, dans un périmètre de 180 mètres ; au delà, des difficultés de connexion se font sentir. Nous avons ainsi conclu que n'ayant pas de problème, il était difficile de réaliser un article pour la conférence Globecom'. Nous avons cependant envisagé de réitérer ces tests en intérieur.

De plus, nous nous sommes mis d'accord sur le fait que ces tests en intérieur terminés, il fallait mettre l'accent sur la partie ad'hoc du stage et sur le rapport bibliographique. La semaine à venir sera donc destinée à la rédaction de ce rapport, et à un état de l'art sur la partie purement ad'hoc. Il sera également question de trouver des informations sur d'éventuelles plateformes de simulation / d'émulation de réseaux ad'hoc.

Compte rendu de réunion du mercredi 15 mars 2006

Lors de cette réunion, nous avons discuté d'émulateurs de réseau ad'hoc, et de leur intérêt quant à notre étude. Il faudra dès la semaine prochaine voir si les utiliser serait une bonne idée ou pas.

Ce qui est prévu la semaine à venir

- Phase de recherche sur les réseaux ad'hoc
- Rédaction du rapport bibliographique
- Recherche sur les plateformes d'émulation

Problème rencontré

- Incompatibilité entre notre matériel Wifi et le logiciel Network Stumbler censé nous fournir le SNR des liaisons

semaine 12 (du lundi 20 au vendredi 24 mars 2006)

Ce qui a été fait

- Créaction de scripts bash (suite et fin) pour obtenir périodiquement dans un fichier texte le Signal et le Bruit sur la liaison via l'outil iwspy
- Réalisation des tests en intérieur et analyse des résultats
- Rédaction du rapport bibliographique de mercredi à vendredi sur tout ce qui a été fait jusqu'à présent

Ce qui est prévu la semaine à venir

- Recherche bibliographique sur les réseaux ad-hoc
- Suite de la rédaction du rapport bibliographique
- Eventuellement début de recherche sur les plateformes d'émulation, en vue du déplacement sur Paris

Problème rencontré

- Tests d'intérieur peu concluants, remettant fortement en cause le fait d'utiliser le FPS comme critère de jouabilité du jeu vidéo

semaine 13 (du lundi 27 au vendredi 31 mars 2006)

Ce qui a été fait

- Suite de la rédaction du rapport de bibliographie ; rapport en très bon état d'avancement
- Recherches sur la qualité de service sur réseaux ad-hoc
- Recherches sur les différentes plateformes d'émulations de réseaux ad-hoc
- Installation et tests du jeu Quake IV

Ce qui est prévu la semaine à venir

- Semaine à Paris, ayant pour but de définir les futurs axes de recherche
- Recherches sur la compilation de Quake et sur la fonction Rollback de Quake 3
- Présentation et réunion autour du stage à l'université de Paris 13

semaine 14 (du lundi 3 au vendredi 7 avril 2006)

Semaine à Paris, à l'université de Paris XIII avec Khaled Boussetta et Nadjib Achir

Ce qui a été fait

- Discussions quant à la suite du projet
- Recherche d'une plateforme d'émulation : test de l'émulateur de olsr.org et de ModelNet
- Test en réel du jeu Quake IV avec 4 PC sur réseau ad-hoc OLSR
- Recherches sur les différentes plateformes d'émulations de réseaux ad-hoc
- Installation et tests du jeu Quake IV

Ce qui est prévu la semaine à venir

- Etude du code source de Quake III : possibilité de compiler ?
- Réaliser des tests de jeu sur réseau ad-hoc multisauts
- Présentation du projet au L2TI à l'université de Paris 13

semaine 15 (du lundi 10 au vendredi 14 avril 2006)

Seconde semaine à Paris, à l'université de Paris XIII avec Khaled Boussetta et Nadjib Achir

Ce qui a été fait

- Arrêt définitif de la conduite à suivre pour la réalisation du projet
- Présentation du projet au L2TI devant les autres stagiaires du laboratoire
- Recherches concernant le code source de Quake III. Réussite dans la compilation sous Windows et sous MacOS
- Test du réseau ad-hoc en utilisant OLSR avec Quake III
- Fin de la rédaction du rapport bibliographique

Ce qui est prévu la semaine à venir

- Etude du code source de Quake III

semaine 16 (du lundi 17 au vendredi 21 avril 2006)

Ce qui a été fait

- Refonte du rapport de bibliographie suite à différentes remarques (confidentialité, ordonnancement)
- Débuts des recherches sur le code source de Quake :
- Compréhension de l'architecture globale du code
- Compréhension des différents mécanismes de compilation et de jeu client / serveur (mécanismes anti-triche, utilisation des différentes extensions du jeu...)
- Etude de la faisabilité d'une implémentation de bots côté client

Ce qui est prévu la semaine à venir

- Accueil du tuteur de l'école Vincent Ricordel dans les locaux de France Telecom
- Préparation de la soutenance bibliographique
- Poursuivre l'étude du code source

semaine 17 (du lundi 24 au vendredi 28 avril 2006)

Ce qui a été fait

- Recherche des paramètres de qualité récupérables au sein du code source
- Etude en particulier du lagomètre, très bon indicateur de qualité du jeu
- Préparation de la visite de Vincent Ricordel et de la soutenance en réalisant une présentation
- Accueil du tuteur mardi 25 après-midi
- Soutenance bibliographique à l'Ecole des Mines de Nantes le vendredi 28 avril

Ce qui est prévu la semaine à venir

- Compréhension du lagomètre et de ses mécanismes
- Savoir quels élements et variables pourront nous amener à avoir un critère de jugement de qualité

semaine 18 (du mardi 2 au vendredi 5 mai 2006)

Ce qui a été fait

- Compréhension quasi complète des mécanismes mis en jeu dans le lagomètre
- Possibilité de récupérer les données pertinentes de qualité au sein de ce lagomètre
- Début de rédaction d'un document de synthèse sur toutes les connaissances acquises sur le code
- Lecture d'articles sur la définition de la qualité (objective et subjective) d'un jeu vidéo

Ce qui est prévu la semaine à venir

- Fin de rédaction du document de synthèse sur le code source de Quake
- A partir des données récupérées du code source, indiquer les paramètres de qualité que nous utiliserons
- Essayer de définir complètement les différentes métriques de qualité qui seront utilisées pour les tests, qu'elles soient objectives ou subjectives

semaine 19 (du mardi 9 au vendredi 12 mai 2006)

Ce qui a été fait

- Fin de rédaction de synthèse sur le code source de Quake
- Lecture de différents articles sur le jugement de qualité du jeu vidéo
- Définition des métriques à utiliser

Ce qui est prévu la semaine à venir

- Trouver des informations quant à l'implémentation de bots côté client
- Entériner cette phase sur le jeu et je code source de Quake pour passer à une autre partie du projet

semaine 20 (du lundi 15 au vendredi 19 mai 2006)

Ce qui a été fait

- Présentation succinte devant l'URD des avancées sur le code source et les métriques de qualité
- Recherche sur l'implémentations de bots côté client :
- Etude du code source et des paks d'extension pour comprendre le fonctionnement des bots
- Envoi de mails à différents développeurs qui avaient réalisé des bots côté client sur Quake et Quake II
- Recherches quant à l'automatisation des mouvements du client chargés à partir de fichiers textes
- Téléchargement du débuggueur MSVC++, conseillé pour le test de Quake III, non encore testé

Ce qui est prévu la semaine à venir

- A définir dès lundi avec Khaled et Nadjib

semaine 21 (du lundi 22 au vendredi 26 mai 2006)

Ce qui a été fait

- Dernières recherches sur une possibilité d'implémenter des bots côté client
- Installation de FreeBSD
- Compilation du noyau pour pouvoir utiliser le firewall système ipfw ainsi que DummyNet
- Prise en main de DummyNet et étude des différentes commandes utilisables
- Test sur installation réelle entre deux PC connectés en Ethernet : résultats encourageants

Ce qui est prévu la semaine à venir

- Continuer l'étude de DummyNet en essayant de monter la plateforme complète, avec à chaque bout des joueurs jouant à Quake III Arena
- Eventuellement voir si des premiers résultats intéressants peuvent être obtenus

semaine 22 (du lundi 29 mai au vendredi 2 juin 2006)

Ce qui a été fait

- Montage complet de la plateforme, après quelques soucis matériels : Routage au niveau de la machine FreeBSD
- Premiers tests réels avec le jeu Quake
- Recherches sur Internet de papiers évaluant les réseaux ad-hoc
- Ecriture de scripts en vue des tests à venir

Ce qui est prévu la semaine à venir

- Lancer et analyser les scripts réalisés

semaine 23 (du lundi 5 au vendredi 9 juin 2006)

Ce qui a été fait

- Fin de la rédaction des scripts de test
- Lancement de tous les tests concernant la gigue, le délai, la bande passante et la perte
- Récupération de tous les critères pertinents de jouabilité dans les fichiers de log du jeu
- Analyse des fichiers de log et tracé de courbes
- Rédaction d'un document sommaire présentant ces premiers résultats

Ce qui est prévu la semaine à venir

- Réfléchir quant à la parution d'un papier dans NetGames
- Ajouter à ces recherches une composante "Réseaux Ad-Hoc"

semaine 24 (du lundi 12 au vendredi 16 juin 2006)

Ce qui a été fait

- Installation de gloMoSim
- Prise en main du simulateur
- Modification du code source du simulateur pour avoir des traces conformes à ce que nous avions besoin
- Développement de l'ensemble des scripts nécessaires
- Lancement de la batterie de test
- Développement de scripts shell et awk permettant d'analyser les résultats des simulations
- Analyse des résultats

Ce qui est prévu la semaine à venir

- A discuter en réunion en début de semaine prochaine

semaine 25 (du lundi 19 au vendredi 23 juin 2006)

Ce qui a été fait

- Interprétation des premiers résultats de GloMoSim : il faudra renouveler l'expérience avec une méthodologie stricte et un grand nombre de fois par points
- Mis en place des expérimentations finales à mener
- Début de rédaction du rapport de stage final

Ce qui est prévu la semaine à venir

- Réaliser tous les scripts de test, les lancer, les analyser et les interpréter
- Rédiger le rapport final de façon conséquente

semaine 26 (du lundi 26 au vendredi 30 juin 2006)

Ce qui a été fait

- Création du programme C générateur de scripts
- Créaction de tous les fichiers de script nécessaires pour l'analyse des derniers résultats
- Analyse des résultats GloMoSim
- Rédaction du rapport de stage

Ce qui est prévu la semaine à venir

- Finir le rapport de stage et préparer la soutenance
- Réaliser d'autres expériences pertinentes et essayer de trouver des pistes d'amélioration sur les réseaux ad-hoc en vue du papier