Ahmad SARDOUK
Responsable *UTT*
Dominique GAÏTI

Branche : SIT6–IR–RACOR
Année : 2006-2007
Semestre : Printemps 07

Jeux vidéo multi joueurs sur réseau Ad hoc

Résumé

Ce stage rentre dans le cadre d'un projet RIAM MAD GAMES. L'objectif du projet est de développer un middleware réseau permettant le support de jeux vidéo multi-joueurs sur une infrastructure de réseaux sans fil, en mode ad-hoc.

Mon travail consiste à améliorer la qualité de service au niveau routage par le développement d'un protocole de routage adapté aux applications de jeux multi-joueurs.

Le stage a commencé par une élaboration d'un état de l'art sur les extensions QoS existantes pour réseaux ad hoc. Remarquant que les communications dans un jeu passe en majoritairement par des diffusions entières dans le réseau et quelques communications unicasts, nous nous sommes intéressés en premier à voir ce qui se fait dans la littérature sur la diffusion optimisée.

Dans une seconde étape, nous avons étudié les performances de plusieurs algorithmes de diffusions optimisées (*Probability Based*, *Area Based*, *Neighbour Knowledge*) sous le simulateur Glomosim. Pour cela, nous avons également proposé un modèle de mobilité approprié à ce genre d'environnement.

La troisième partie de stage a consisté à la proposition et le développement d'un protocole de routage, unicast/broadcast, avec qualité de service.

France Telecom R&D
2 avenue Pierre Marzin
22307, Lannion

Responsable direct :
Sidi-Mohamed SENOUCI

Mots clés
1- Recherche appliquée, développement
2- Transports et télécommunication, poste
3- Réseaux d'ordinateurs, Informatique
4- Gestionnaires de réseaux (logiciels)

Remerciement

Je tiens tout d'abord à adresser mes plus vifs remerciements à monsieur *Sidimohammed SENOUCI*, et monsieur *Nadjib ACHIR* et *Khaled BOUSSETTA*, pour avoir suivis ce travail de près, pour leur esprit de travail dynamique et rigoureux et pour l'autonomie d'action qu'ils m'ont accordée tout au long de la durée de ce stage,

Je remercie également Madame *Dominique GAITI* mon responsable pédagogique pour l'intérêt porté à ce stage.

Ma gratitude est également adressée à l'ensemble de personnel de l'équipe M2I à France Telecom R&d Lannion pour leur accueil et convivialité et plus particulièrement *Moez JERBI* et *Riad KORTEBI* pour leurs conseils et aides techniques.

Tous mes remerciements a mon ami *Tamim KHODER*, ma famille et tous ceux qui m'ont soutenue et qui de près ou de loin ont contribué à l'aboutissement de ce travail.

Merci a tous

Résumé

Ce stage rentre dans le cadre d'un projet RIAM MAD GAMES. L'objectif du projet est de développer un middleware réseau permettant le support de jeux vidéo multi-joueurs sur une infrastructure de réseaux sans fil, en mode ad-hoc.

Mon travail consiste à améliorer la qualité de service au niveau routage par le développement d'un protocole de routage adapté aux applications de jeux multi-joueurs.

Le stage a commencé par une élaboration d'un état de l'art sur les extensions QoS existantes pour réseaux ad hoc. Remarquant que les communications dans un jeu passe en majoritairement par des diffusions entières dans le réseau et quelques communications unicasts, nous nous sommes intéressés en premier à voir ce qui se fait dans la littérature sur la diffusion optimisée.

Dans une seconde étape, nous avons étudié les performances de plusieurs algorithmes de diffusions optimisées (*Probability Based, Area Based, Neighbour Knowledge*) sous le simulateur Glomosim. Pour cela, nous avons également proposé un modèle de mobilité approprié à ce genre d'environnement.

La troisième partie de stage a consisté à la proposition et le développement d'un protocole de routage, unicast/broadcast, avec QoS.

Mots clés : réseau Ad hoc, jeu vidéo multi joueurs, QoS, diffusion optimisée

Table de matière

Résumé ..... **i**

1. Introduction **1**
1.1 Contexte économique 1
1.2 Le jeu vidéo sur réseaux ad hoc 1
1.3 Problématique 1
1.4 Solution 2
1.5 Plan 2

2. Etat de l'art **3**
2.1 Les différents types de jeux 3
2.2 Les paramètres de QoS pour les jeux en réseau ad hoc 3
2.3 Architecture réseau 3
2.3.1 Architecture Client/Serveur 4
2.3.2 Architecture Pair à pair (P2P) 4
2.3.3 Architecture hybride 5
2.4 Diffusion dans les réseaux ad hoc 6
2.5 Modèle de mobilité de joueurs dans un réseau ad hoc 7
2.6 Solution proposée dans la littérature pour l'amélioration de QoS dans les jeux vidéo multi joueurs sur MANET 9

3. Travail effectué **11**
3.1 Choix des paramètres 11
3.2 Cadres de simulation 11
3.2.1 Protocole de routage 11
3.2.2 Modèle de mobilité 13
3.3 Paramètres de simulation 15
3.3.1 Scénario 15
3.3.2 Critères de performance 16
3.4 Résultats de simulation 18

4. Conclusion et perspectives **22**

5. Liste des abréviations **23**

6. Références **23**

ANNEXES ..... **I**

Figures

Figure 1 Architecture Client/Serveur 4
Figure 2 Architecture Pair à Pair 4
Figure 3 Architecture Hybride 5
Figure 4 Simple Flooding 6
Figure 5 L'algorithme *'Distance Based'* 6
Figure 6 RWP vs NGMM 8
Figure 7 Traces de mobilité 8
Figure 8 Fonctionnement de l'algorithme de routage 13
Figure 9 Taux de livraison de paquets dans une architecture C/S sans mobilité 18
Figure 10 Taux de livraison des paquets dans une architecture P2P sans mobilité 18
Figure 11 Taux de livraison des paquets dans une architecture C/S avec notre modèle de mobilité 18
Figure 12 Taux de livraison des paquets dans une architecture P2P avec notre modèle de mobilité 18
Figure 13 Délai de bout en bout dans une architecture C/S sans mobilité 19
Figure 14 Délai de bout en bout dans une architecture P2P sans mobilité 19
Figure 15 Délai de bout en bout dans une architecture C/S avec notre modèle de mobilité 19
Figure 16 Délai de bout en bout dans une architecture P2P avec notre modèle de mobilité 19
Figure 17 Consommation d'énergie par nœud dans un réseau de 16 nœuds, architecture C/S sans mobilité 20
Figure 18 Consommation d'énergie par nœud dans un réseau de 16 nœuds, architecture P2P sans mobilité 20
Figure 19 Consommation d'énergie par nœud dans un réseau de 16 nœuds, architecture C/S avec notre modèle de mobilité 20
Figure 20 Consommation d'énergie par nœud dans un réseau de 16 nœuds, architecture P2P avec notre modèle de mobilité 20
Figure 21 SRB dans une architecture C/S sans mobilité 21
Figure 22 SRB dans une architecture C/S avec mobilité 21
Figure 23 SRB dans une architecture P2P sans mobilité 21
Figure 24 SRB dans une architecture P2P avec mobilité 21

1. Introduction

Ce stage se place dans le cadre d'un projet RIAM MADGAMES (Middleware for AD-hoc networked vidéo GAMES). Les partenaires du projet sont France Telecom R&D, l'université de Paris 13 (chef de file), l'université de Paris 6 et un éditeur de jeux Fandango. L'objectif de MADGAMES est de développer un réseau middleware permettant le support de jeux vidéo multi-joueurs sur une infrastructure de réseaux sans fil, en mode ad hoc.

1.1 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 multi-joueurs 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 vidéo multi-joueurs 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 multi-joueurs sur réseaux Wifi en mode ad-hoc.

1.2 Le jeu vidéo sur réseaux ad hoc

Actuellement dans les parties multi-joueurs, les consoles portables utilisent un mode ad-hoc mono-saut (n'impliquant donc pas de routage). Les consoles portables s'utilisent de façon spontanée entre amis, entre écoliers dans la cours de récréation ou dans les transports en commun. Ainsi les réseaux ad-hoc répondent 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.

1.3 Problématique

Afin de permettre à un nombre relativement élevé de joueurs de se connecter à une telle partie, dans la cours de récréation ou autre, qu’ils soient éloignés ou pas, plusieurs questions doivent encore être adressées par la communauté de recherche. Plusieurs difficultés viennent des contraintes inhérentes qui caractérisent la technologie de MANET. En particulier la mobilité des nœuds et les limitations d'énergie. Ces problèmes mènent à des connectivités non garanties et provoquent des taux de latence et perte de paquets non supportables dans un contexte de jeu vidéo multi-joueurs.

Un autre problème vient de l'architecture client serveur utilisé actuellement dans la majorité de jeux vidéo multi-joueurs. Dans cette architecture seul le serveur représente le coordinateur/gestionnaire du jeu. Dans un contexte d'un réseau LAN filaire, la machine serveur est toujours joignable et possède l'énergie suffisante pour jouer ce rôle. Dans un MANET, les ressources limitées du serveur plus précisément en termes d'énergie et de possibilité de se déplacer vers des zones non joignables représentent des vrais problèmes à résoudre. En plus, la position géographique du serveur par rapport aux autres nœuds est un facteur important à prendre en considération. Le nombre de sauts nécessaires pour se connecter au serveur peut causer un vrai problème. Les nœuds à un saut du serveur communiquent avec lui sans latence ni perte. Les nœuds éloignés du serveur (en termes de saut) peuvent souffrir de taux remarquables de latence, de perte de route et de perte des paquets.

1.4 Solution

Nous proposons d'améliorer la QoS au niveau routage par le développement d'un protocole de routage adapté aux applications de jeux multi joueurs. Ce nouveau protocole de routage doit respecter les contraintes de délai, perte, et ne doit pas consommer beaucoup d'énergie. Il devra acheminer les paquets de bout en bout avec un délai borné et acceptable. Les paquets doivent être routés avec le moindre possible de pertes. Il ne doit pas surcharger le réseau ni consommer beaucoup d'énergie en envoyant un grand nombre de messages de contrôle.

Pour l'architecture du réseau, nous trouvons le problème dans la centralisation du serveur. Pour résoudre ce problème nous allons examiner au début les architectures alternatives existantes, Pair à Pair et hybride.

1.5 Plan

Dans ce rapport je présenterai mon stage de Master2 recherche et d'ingénierie réseaux sur l'amélioration de QoS pour les jeux vidéo multi joueurs sur réseau Ad hoc. La partie recherche bibliographique de ce stage est présentée dans la section "Etat de l'art". Dans cette section je présente une étude sur l'environnement de jeu vidéo multi joueurs sur MANET en termes de types des jeux vidéo, de paramètres de QoS à prendre en compte, d'architecture réseau, de diffusion optimisée et de modèles de mobilité. A la fin de l'état de l'art je présenterai les travaux d'un autre groupe de recherche sur la QoS pour les jeux dans MANET. Après je présente le « Travail effectué » : les choix que nous avons pris, les algorithmes implémentés et les scénarios de simulations avec les résultats obtenues. Je terminerai ce rapport par une conclusion et les perspectives que nous avons jugées importantes.

2. Etat de l'art

Un état de l'art nous a été indispensable pour comprendre les contraintes de QoS pour les différentes types de jeux, leurs modèles de communications qui se basent sur l'architecture réseau utilisé, comment optimiser la diffusion qu'on a trouvée indispensable dans un contexte de jeu en réseau, le modèle de mobilité de joueurs dans un contexte de jeu vidéo multi joueurs sur MANET et finalement la contribution des autres groupes de recherches dans ce domaine.

2.1 Les différents types de jeux

Avant d'aborder le contexte réseau pour les jeux vidéo multi joueurs, il est intéressant de définir les différents types de jeux qui nous intéressent dans ce travail. L'ancien stagiaire sur ce projet, Pierre CHECA, a présenté dans son rapport [1] les différents types de jeux vidéo connus de nos jours. Dans ce paragraphe nous définissons en quelques mots les jeux qui sont contraignants en termes de QoS. Premièrement, les jeux de stratégie aussi appelés **RTS** (*Real Time Strategy*). Le joueur 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: *Starcraft*, *Warcraft III* et *Age of Empires*). La deuxième famille intéressante est : les jeux de tir. Dans cette famille on s'intéresse aux jeux appelés **FPS** (*First Person Shooter*) le but est simplement de tuer les autres. Les jeux phares sont : *Counter Strike*, *Quake*, *Doom*, *Half Life* et *Unreal Tournament*. Finalement les jeux de sports comme les jeux de voitures, (exemple : *Gran Turismo* et *Mario Kart*).

2.2 Les paramètres de QoS pour les jeux en réseau ad hoc

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 pas de la même manière sur la jouabilité [2] [3] [4] [5] [6] 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 500ms pour les RTS) implique nécessairement une dégradation de la jouabilité, même si la limite d'acceptabilité varie selon le jeu.

En ce qui concerne les autres paramètres tels que la gigue et le taux de perte, ils interviennent moins dans la dégradation du jeu. [3] [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.

2.3 Architecture réseau

L'architecture Client Serveur (C/S) est l'architecture dominante dans le monde des jeux. L'architecture Pair à Pair (P2P) représente une architecture alternative. Nous allons étudier dans cette partie les principes de fonctionnement de chaque architecture, les comparer et distinguer leurs modèles de communications. Cette étude va être faite du point de vue architecture du jeu vidéo multi joueurs sur réseau ad-hoc. A la fin de cette partie, nous présenterons également une architecture hybride proposée comme solution aux problèmes rencontrés dans les architectures C/S et P2P.

2.3.1 Architecture Client/Serveur

Dans cette architecture tous les clients se connectent sur une machine centrale, le serveur, qui est installé sur une simple machine d'un des joueurs. Le serveur est responsable de maintenir l'état global du jeu alors que les clients sont responsables de le mettre à jour, en lui communiquant l'ensemble de leurs actions.

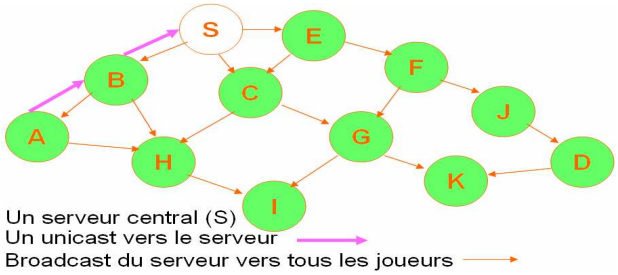

Figure 1 Architecture Client/Serveur

La figure 1 présente une architecture client serveur. Le nœud S représente le serveur qui communique avec ses clients en utilisant un mécanisme de diffusion, alors que ses clients lui acheminent leurs paquets en utilisant une communication unicast.

L'une des avantages les plus importants de cette architecture est la cohérence du jeu qui vient de la centralisation du calcul de l'état global du jeu au niveau de serveur. Cette centralisation de calcul permet aux nœuds de ressources limités d'économiser le maximum possible de leur énergie.

Malheureusement, la centralisation du serveur a aussi des inconvénients. Ceci représente un point de défaillance, car si le serveur tombe en panne (par exemple pour manque de batterie venant de la forte consommation de maintient et de diffusion d'état de jeu) ou se déplace vers des zones non couvert par le réseau, le jeu va s'arrêter obligatoirement. Il faut noter encore que plus le nombre de joueurs est grand plus le serveur doit assurer une bande passante et une énergie élevée pour satisfaire les besoins de ses clients.

2.3.2 Architecture Pair à pair (P2P)

Cette architecture représente une alternative à l'architecture C/S. Elle est souvent utilisée dans les jeux de stratégie. L'établissement du jeu passe par une phase C/S, un joueur crée le jeu alors que les autres se connectent à lui pour participer au jeu. Une fois connectés, les joueurs communiquent en mode P2P.

L'architecture P2P représente une solution complètement distribuée (voir Figure 2) où chaque nœud est responsable de maintenir l'état global du jeu et de mettre à jour l'ensemble des nœuds du réseau.

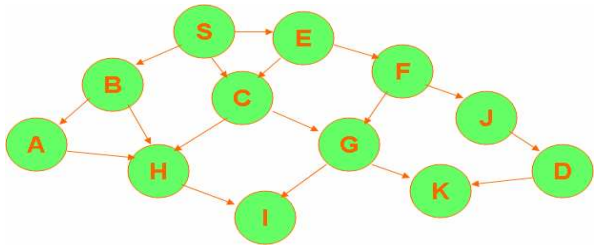

Figure 2 Architecture Pair à Pair

Pour créer un jeu, un trafic unicast est requis. Ce trafic n'est pas contraignant en termes de QoS. Pourtant, les communications entre les nœuds durant la session passent par des diffusions.

L'avantage principal de cette architecture vient de l'inexistence de point de gestion/calcul central. Comme il n'est pas nécessaire de maintenir une communication persistant vers un nœud central, cette architecture est plus adaptée au MANET.

Pourtant, la gestion de cohérence du jeu dans un contexte complètement distribué n'est pas une tache facile. Au niveau de l'énergie, dans cette architecture, chaque nœud fait plus de calcul et consomme donc plus d'énergie.

2.3.3 Architecture hybride

Une architecture hybride est un compromis entre C/S et P2P. Elle est proposée pour résoudre le problème de centralisation du serveur de C/S et de cohérence de P2P. Nous allons nous concentrer sur les travaux de Riera et al [7] qui proposent une architecture hybride pour les jeux vidéo sur réseaux ad hoc. Le principe est de diviser le réseau en zones, voir figure 3, et dans chaque zone de sélectionner un nœud comme serveur de zone. Le jeu est centralisé dans les zones et distribué entre les zones.

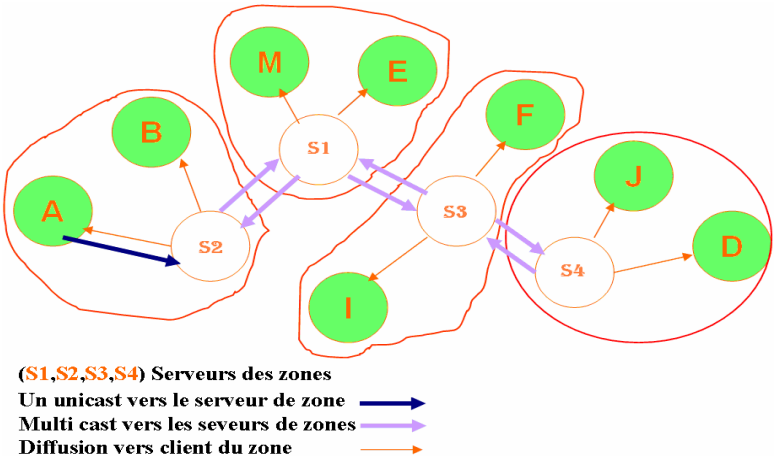

Figure 3 Architecture Hybride

Les difficultés de cette architecture sont la formation des zones et le choix des serveurs. Welnitz et Wolf [8] proposent un algorithme d'élection des serveurs et de création des zones pour les jeux vidéo multi-joueurs sur réseaux Ad-hoc. Leur algorithme utilise des informations de voisinage pour déterminer le rôle de chaque nœud (client ou serveur de zone). Il assure une distance de deux sauts au maximum entre un client et un serveur. Il gère la mobilité des nœuds et des serveurs. La courte distance, 2 sauts maximum, entre client et serveur réduit les problèmes de communication entre eux. Le vrai problème de communication se pose au niveau de la communication multicast entre les serveurs des zones.

Dans cette architecture, les clients communiquent avec leur serveur de zone par des trafics unicast. Pourtant, les serveurs des zones diffusent l'état global de jeux dans leur zone et communiquent en multicast avec les autres serveurs de zones pour maintenir l'état global du jeu.

Comme le nombre de joueur dans une session de jeu vidéo multi joueurs sur MANET, n'est pas très élevé nous ne trouvons pas l'intérêt d'implémenter ce type d'architecture.

2.4 Diffusion dans les réseaux ad hoc

Plusieurs algorithmes de diffusion dans les réseaux ad hoc ont été proposés dans la littérature [9] [10] [11] [12] [13]. Williams et Camp [14] catégorisent les algorithmes de diffusion en quatre familles: '*Simple Flooding*', '*Probability Based*', '*Area Based*' et '*Neighborhoud Based*'.

L'algorithme du *Simple Flooding*, (cf. Figure 4), est la solution de diffusion par défaut dans les réseaux ad hoc. Dans cet algorithme, chaque nœud rediffuse tous les messages reçus, une seule fois, pour tous ces voisins. Les avantages de cet algorithme sont sa simplicité et son haut pourcentage de livraison. Pourtant, quand on considère un réseau dense, Ni et al [9] montrent que le *Simple Flooding* conduit à des problèmes de redondance, de contention et de collision. Ces problèmes sont connus sous le nom de tempête de diffusion ou (*Broadcast Storm problem*).

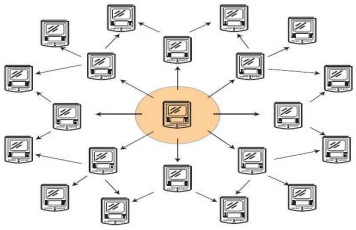

Figure 4 Simple Flooding

Dans l'algorithme de '*Probability Based*', chaque nœud rediffuse les messages reçus avec une probabilité $\underline{p}$ prédéfini ou bien il les supprime avec une probabilité (1-$\underline{p}$). Quand $\underline{p}$=1 cet algorithme se comporte comme *Simple Flooding*.

L'algorithme '*Counter Based*' est un autre algorithme qui appartient à la famille '*Probability Based*'. Avant de rediffuser un message reçu *M*, chaque nœud définit un temps d'attente aléatoire. Pendant ce temps, il y a une forte probabilité que le nœud reçoit plusieurs copies du même message, plus précisément dans un réseau dense. Le nœud compte ainsi le nombre de copies reçus *C* pendant ce temps. Le nœud rediffuse le message *M* seulement si *C* est plus grand qu'un seuil prédéfini $\underline{c}$, sinon il le supprime. En comparant ce dernier algorithme avec les précédents, on peut dire que la réduction du nombre de messages redondants est fortement liée à la densité du réseau. En effet, dans les réseaux ad hoc denses, un faible nombre de nœuds rediffuse les messages. Dans les réseaux non denses la majorité des nœuds auront à rediffuser les messages.

Dans la troisième famille, '*Area Based*', les décisions se basent sur l'estimation de la zone de couverture des nœuds. Deux algorithmes appartiennent à cette famille [9]: '*Distance Based*' et '*Location Based*'.

L'algorithme '*Distance Based*' se base sur la distance *d* entre l'émetteur et le récepteur. Dans les réseaux ad hoc deux nœuds peuvent être très proches l'un de l'autre, et peuvent ainsi couvrir la même zone de couverture. La figure 5 montre la zone de couverture des nœuds *A* et *B* : plus la distance *d* est petite plus la zone de couverture commune est grande. Supposons que *A* envoie un message au nœud *B*. Si *d* est plus petit qu'un certain seuil prédéfini $\underline{d}$, *B* considère qu'il couvre à peu prés la même zone que *A*, et il supprime donc le message, sinon il le rediffuse. Notons que la distance peut également être remplacée par la puissance du signal capté.

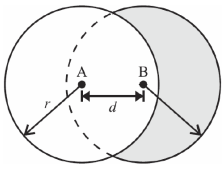

Figure 5 L'algorithme *'Distance Based'*

L'algorithme 'Location Based' offre des informations plus précises que la distance. Chaque nœud ajoute ses coordonnées géographiques dans l'entête de chaque message avant de l'envoyer. En utilisant ces coordonnées et à partir de la portée radio prédéfini des nœuds, le récepteur calcule la zone de couverture de l'émetteur. Puis, le récepteur calcule la zone de couverture Z supplémentaire qu'il va couvrir s'il transmet le message. Si Z ne dépasse pas un seuil prédéfini, le récepteur considère qu'il couvre la même zone et il supprime le message sinon il le rediffuse. Malheureusement, cet algorithme fait l'hypothèse que chaque nœud connait ses coordonnés géographiques (en utilisant un GPS - Global Positioning System- par exemple). Pour des raisons de coût, cette supposition ne peut pas être raisonnable dans notre contexte.

Les performances des deux dernières familles, '*Probability Based*' et '*Area Based*', dépendent largement du bon choix des valeurs des seuils prédéfinis. Le seuil choisi doit fortement être adapté à la densité du réseau.

La dernière famille de ces algorithmes de diffusion est '*Neighborhoud Based*' qui exploite les informations de voisinage. Les principaux algorithmes dans cette famille sont : *Self Pruning* [10], *Dominant Pruning* [10], *Scalable Broadcasting* [12], *Multi Point Relay - MPR* [13], *ad hoc Broadcast Protocol* [11] et *Simplified Multicast Forwarding* [15]. Dans le *Self Pruning*, chaque nœud ajoute à l'entête du paquet les adresses de ses voisins. Le nœud récepteur vérifie si tous ses voisins sont dans la liste, si oui il supprime le message sinon il le rediffuse. Dans le *Self Pruning* les nœuds cherchent leurs voisins à un saut. Dans les autres algorithmes, chaque nœud doit connaître tous les nœuds qui se trouvent à un et deux sauts. Malheureusement, tous ces algorithmes chargent le réseau par des messages de control et consomment beaucoup d'énergie pour collecter les connaissances de voisinage.

2.5 Modèle de mobilité de joueurs dans un réseau ad hoc

Pour une étude de simulation efficace de jeux vidéo multi joueurs sur les réseaux ad hoc, il faut assurer les conditions les plus proches de la réalité. La mobilité des joueurs est un point important qu'on ne peut pas ignorer dans ce genre d'études de performances.

Camp et al [16] présentent un état de l'art sur les deux catégories de mobilité dans les réseaux ad hoc:

- mobilité de l'entité: ce modèle s'intéresse au mouvement de chaque nœud sans prendre en compte la coopération avec les nœuds voisins ou distants;

- mobilité de groupe : ce modèle définit les directifs de mobilité d'un ensemble de nœuds. La mobilité d'un nœud dans un groupe doit respecter les règles de mobilité du groupe.

Dans les deux catégories il y a un grand nombre de modèles proposés. Dans ce rapport, pour des limitations de nombre de pages, on discutera seulement les modèles les plus utilisés.

La majorité des chercheurs utilisent un modèle de mobilité de l'entité et plus spécialement le *Random Way Point* (RWP). Dans ce modèle un nœud se déplace vers une direction aléatoire avec une vitesse uniformément distribué entre une valeur minimale et une valeur maximale. Entre les changements de direction et/ou vitesse, un nœud se repose un temps prédéfini (pause time). Dans RWP, on ne peut contrôler que la vitesse des nœuds et le temps de repos.

Dans les jeux en réseaux, on a généralement deux équipes ou plus qui se battent entre eux, à travers leurs joueurs virtuels. De ce point de vue, on peut considérer que l'ensemble des joueurs qui

appartiennent à la même équipe, bougent en groupe. Etre dans la même équipe est une sorte de relation sociale entre les joueurs. Musolesi et Mascolo [17] proposent ainsi un modèle de mobilité de groupe basé sur les relations sociales entre les nœuds, '*Community Based*'. Les auteurs divisent les nœuds en communautés. Chaque communauté est un ensemble de nœuds ayant une relation sociale entre eux. Une fois les communautés constituées, ils les placent dans des zones géographiquement séparés. Pour se déplacer la première fois, le nœud choisit aléatoirement sa destination (en termes de X et Y) dans la zone de sa communauté. Les prochains déplacements se feront dans la zone de la communauté caractérisée par la plus haute attractivité sociale. Cette zone pourra très bien être la zone où il se situe déjà, mais aussi une autre, avec X et Y choisis aléatoirement. Cette approche nous a semblé logique. Dans pas mal de jeu, les joueurs de la même équipe parlent entre eux et préfèrent que les autres ne les entendent pas. Mais, suite à une enquête, nous avons trouvé que cette logique ne représente pas vraiment la réalité. Les joueurs se déplacent aléatoirement dans tous les jeux.

Tan et al [18] ont proposé un modèle de mobilité appelé '*Networked Game Mobility Model*' (NGMM). Ce modèle est basé sur le RWP. Dans la figure 6 on expliquera la différence de fonctionnement entre RWP et NGMM. La position initiale du nœud étant le point X, le nœud a choisi aléatoirement sa destination Y. En RWP, ligne orange, le nœud se déplace linéairement de X à Y avec une vitesse aléatoire. NGMM définit un ensemble de point de passage avant d'arriver à la destination Y. Dans chaque axe, il choisit une vitesse aléatoire. La vitesse de déplacement est alors la moyenne des vitesses utilisées pour arriver à destination.

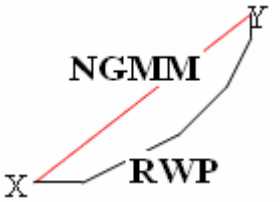

Figure 6 RWP vs NGMM

Afin de valider leur modèle de mobilité, ils ont pris les traces de mobilité d'une personne virtuelle du jeu Quacke2. Ils ont récupéré les traces d'un nœud en utilisant leur modèle, puis en utilisant le modèle RWP. La figure 7 montre les traces de mobilité d'un nœud dans les 3 modèles pendant 60s. On peut distinguer dans cette figure la longueur des lignes linéaires et la densité de mobilité. On peut voir que RWP est un modèle de forte mobilité avec des distances linéaires relativement grandes. NGMM est moins dense que RWP et la distance linéaire moyenne parcourue par un nœud est beaucoup plus petite qu'en RWP.

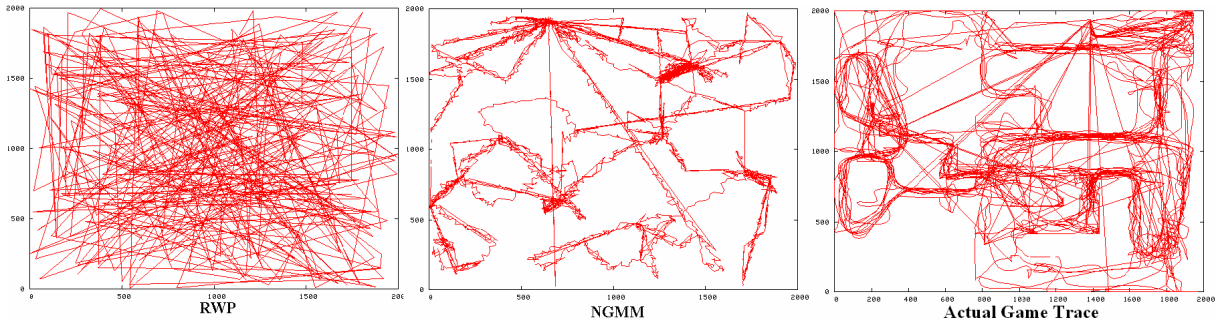

Figure 7 Traces de mobilité

2.6 Solution proposée dans la littérature pour l'amélioration de QoS dans les jeux vidéo multi joueurs sur MANET

Les travaux sur la qualité de service sur réseaux ad hoc sont nombreux, mais rares sont ceux qui discutent le contexte des jeux vidéo multi-joueurs. Dans cette partie nous allons présenter les travaux de Budke et al [19] qui nous semblent être les plus importants du domaine. Les auteurs de cet article essayent de satisfaire les besoins des jeux vidéo multi-joueurs sur réseaux Ad Hoc en termes de QoS. Pour cette finalité, ils proposent un modèle sur 3 niveaux : routage, liaison et un niveau intermédiaire entre eux.

Au niveau liaison de données, ils ont décidé d'abandonner, tout d'abord, la technique *RTS/CTS* (*ready to send/clear to send*) utilisé par le MAC de 802.11 pour éviter les problèmes des terminaux cachés. L'émetteur vérifie par ce mécanisme la disponibilité du récepteur pour commencer une communication. Les simulations ont prouvé que ce mécanisme dégrade la QoS dans un contexte de jeu. En outre, ils ont décidé d'utiliser le mécanisme de surveillance de signal radio dans la recherche de route pour éviter les routes qui ont des problèmes radio. Pour éviter la surcharge qui résulte des messages de création et de maintien des tables routages, ils ont utilisé un mécanisme de niveau 2 de détection du voisinage, *neighbor detection*. Ce mécanisme utilise des messages dits *'Beacon'* définis dans l'IEEE 802.11. Le *'Beacon'* est un message que chaque nœud envoie toutes les 100ms pour informer les nœuds voisins de son existence. Et un dernier mécanisme, *Broken link Detection*, est utilisé pour supprimer les liens perdus des tables de routage. Au niveau liaison, si après un certain délai d'attente on ne reçoit pas d'acquittement sur le message envoyé, on considère que la route vers cette destination est perdue, et ils ont proposé de remonter cette information au niveau routage pour la supprimer.

Ils ont proposé de résoudre le problème de congestion sur un niveau intermédiaire entre les couches liaison et réseau. L'IEEE 802.11 donne à tous les nœuds la même probabilité d'accès au support, indépendamment du type de trafic à envoyer. Pour ne pas surcharger le réseau par des trafics non prioritaires, alors qu'il y a des trafics prioritaires dans les files d'attentes, ils ont proposé le *Real time aware rate control*. Ce mécanisme permet à chaque nœud de connaître le type de trafic de ses voisins. Le nœud qui a du trafic faible priorité laisse l'accès au support au nœud de trafic plus prioritaire. Pourtant, ils ont utilisé les files d'attente prioritaires pour acheminer plus rapidement les paquets prioritaires. Cependant, pour ne pas surcharger le réseau par un trafic inutile à cause d'un retard, ils ont proposé un nouveau mécanisme '*constrained queuing timeouts*'. Ce mécanisme surveille les files d'attente et supprime les paquets qui ont dépassé un temps maximal d'acheminement (par exemple : 100ms pour un trafic de jeu FPS).

A travers des simulations, ils ont trouvé que AODV était le meilleur protocole de routage pour les jeux vidéo sur réseau Ad hoc. Au niveau de ce protocole de routage ils ont trouvé que le mécanisme de *Local repair* dégradait le service dans un contexte de jeu sur réseau Ad-hoc. Ce mécanisme permet au nœud intermédiaire qui détecte un lien perdu de sauvegarder temporairement le paquet jusqu'au la réparation du lien. Pourtant, ils ont utilisé le mécanisme de *backup route* pour éviter le problème de lien perdu. Ce mécanisme permet au protocole de routage, lors de découverte de la route, de sauvegarder deux routes, vers la même destination, dans la table de routage, de façon à ce que si la première tombe en panne la deuxième reste disponible.

Afin d'évaluer leur proposition, ils ont simulé leur modèle en utilisant un trafic unicast entre un ensemble de nœuds éloignés de plusieurs sauts l'un de l'autre. Ils ont trouvé que leur modèle pouvait satisfaire les besoins de jeu en dessous de 3 sauts. Malheureusement, ils n'ont pas évalué leur

algorithme dans une architecture réelle comme C/S ou P2P. Ils n'ont pas pris en compte la diffusion que l'ont trouve indispensable dans un contexte de jeux sur MANET. Aussi, plusieurs autres paramètres, comme le trafic et le modèle de mobilité, ne sont pas adaptés au jeu.

3. Travail effectué

Pour évaluer la performance de diffusion dans les MANET dans un contexte de jeu vidéo multi-joueurs, nous avons décidé de simuler quelques algorithmes de diffusion. Dans cette partie nous expliquerons comment nous avons choisi les paramètres les algorithmes de diffusion et la modèle de mobilité. Nous parlerons de leur implémentation et principe de fonctionnement. Puis nous montrerons les scénarios et paramètres de simulation. Nous terminerons cette partie par les résultats obtenus.

3.1 Choix des paramètres

Algorithmes à simuler

Afin d'évaluer leur performance dans le contexte des jeux multi-joueurs sur MANET, nous avons choisi de chaque famille l'algorithme le plus simple, puisque nous croyons que la complexité réduite est un critère fondamental en considérant l'exécution dans des consoles portatives. Les algorithmes choisis sont : *Simple Flooding*, *Probability Based*, *Distance Based* et *Self Pruning*.

Choix d'un modèle de mobilité

Dans l'état de l'art nous avons présenté les modèles les plus proches en terme logique et statistique du monde des jeux. L'inaccessibilité du code, de NGMM, nous a pas permit de tester ce modèle. De plus, NGMM nous semble un modèle de forte mobilité, basé sur des personnages virtuelles qui ne reflètent pas forcément la mobilité des joueurs humains. Les personnages virtuels ont plusieurs objectifs supplémentaires comme le ramassage des trésors. En inspirant de leur méthode de travail d'une part et d'une enquête entre joueurs des jeux réseaux sur WiFi avec infrastructure fixe d'autre part, nous avons décidé de développer notre modèle que j'expliquerai dans la partie cadres de simulation.

3.2 Cadres de simulation

Dans cette partie j'expliquerai les implémentations que j'ai faites pendant ce stage. Dans la première partie, j'expliquerai le principe de fonctionnement du protocole de routage implémenté avec les différents algorithmes de diffusion choisis. Dans la deuxième partie, j'expliquerai notre modèle de mobilité et comment je l'ai implémenté.

3.2.1 Protocole de routage

L'objectif principal de ce stage est de proposer un protocole de routage qui satisfait les besoins des jeux en termes de délai de bout en bout et de perte, et qui respecte les ressources limitées des nœuds. D'après notre état de l'art nous pourrons dire que le nouvel protocole de routage doit, en respectant la QoS prédéfinie, assurer:

- Routage unicast : cas du trafic client vers serveur, ou trafic d'établissement de jeu en pair à pair;
- Diffusion optimisée : pour acheminer les messages entre les pairs ou des serveurs vers les clients;

Tout d'abord, nous avons décidé de créer notre protocole de routage qui dans la partie unicast va reprendre la fonctionnalité d'un protocole déjà existant et pour la partie diffusion qui contient l'algorithme de diffusion le plus adéquat.

Pou choisir un protocole de routage unicast, nous nous sommes basés sur les travaux présentés en [1] et [19] qui comparent plusieurs protocoles de routage dans le contexte de jeu. Les résultats, de ces deux travaux, montrent qu'AODV est le protocole le plus adéquat pour les réseaux de jeux.

Dans la partie diffusion, nous avons décidé d'évaluer la performance de : *Simple Flooding*, *Probability Based*, *Distance Based* et *Self Pruning*. Pour cela, nous les avons implémentés sous Glomosim 2.0 [20].

Afin de faciliter l'explication, la figure 8 montre un organigramme de fonctionnement de l'algorithme de routage. Notre protocole fonctionne en trois modes source, destination et/ou relais:

1. Quand un nœud reçoit un message de niveau application, il connaît qu'il est la source. Il génère un numéro de séquence unique et il l'ajoute à l'entête de message pour l'identifier. D'autres champs optionnels peuvent être ajoutés au message selon le protocole. *Distance Based* ajoute les coordonnées *X* et *Y* du nœud pour faciliter le calcul de distance. *Self Pruning* ajoute la liste d'adresse de ces voisins à un seul saut;

2. Mode destination, c'est quand l'adresse destination égale l'adresse de son nœud ou l'adresse de diffusion. Dans ce cas là le protocole enlève l'entête IP et remonte le message au niveau supérieur. Ce mode n'est exécuté que dans le routage unicast;

3. le mode relais est équivalent au mode destination mais avec la possibilité de rediffuser les messages si nécessaire. Quand un nœud reçoit un message il fait une copie du message sans l'entête qu'il envoie à la couche transport. Si l'algorithme de diffusion permet la rediffusion, le protocole envoie une copie du message au niveau liaison. La décision de retransmission dépend de l'algorithme de diffusion:

    a. En *Probability Based*, on calcule toujours le pourcentage de paquets supprimés *S*. Tant que *S* est inférieur à un seuil prédéfini, notre protocole prend une décision de rediffusion aléatoire (0 ou 1). Si S est égal au seuil il rediffuse directement le message,

    b. En *Distance Based*, on utilise les coordonnées géographiques *X* et *Y* qui se trouvent dans l'entête pour calculer la distance *D* qui sépare l'émetteur précédent du message du nœud courant. Si *D* est supérieur à un seuil *d* prédéfini, il remplace les *X* et *Y* de l'entête par celui du nœud courant et il rediffuse le message, sinon il le supprime,

    c. En *Self Pruning*, on compare la liste des adresses qui se trouvent dans l'entête avec les adresses des voisins du nœud courant. S'il y a des nœuds non couverts, il remplace la liste existante par la liste du nœud courant et il rediffuse le message, sinon il le supprime. Il faut noter qu'en *Self Pruning*, chaque nœud envoie des messages '*Hello*' à des intervalles de temps réguliers pour construire et mettre à jour la table des voisinages,

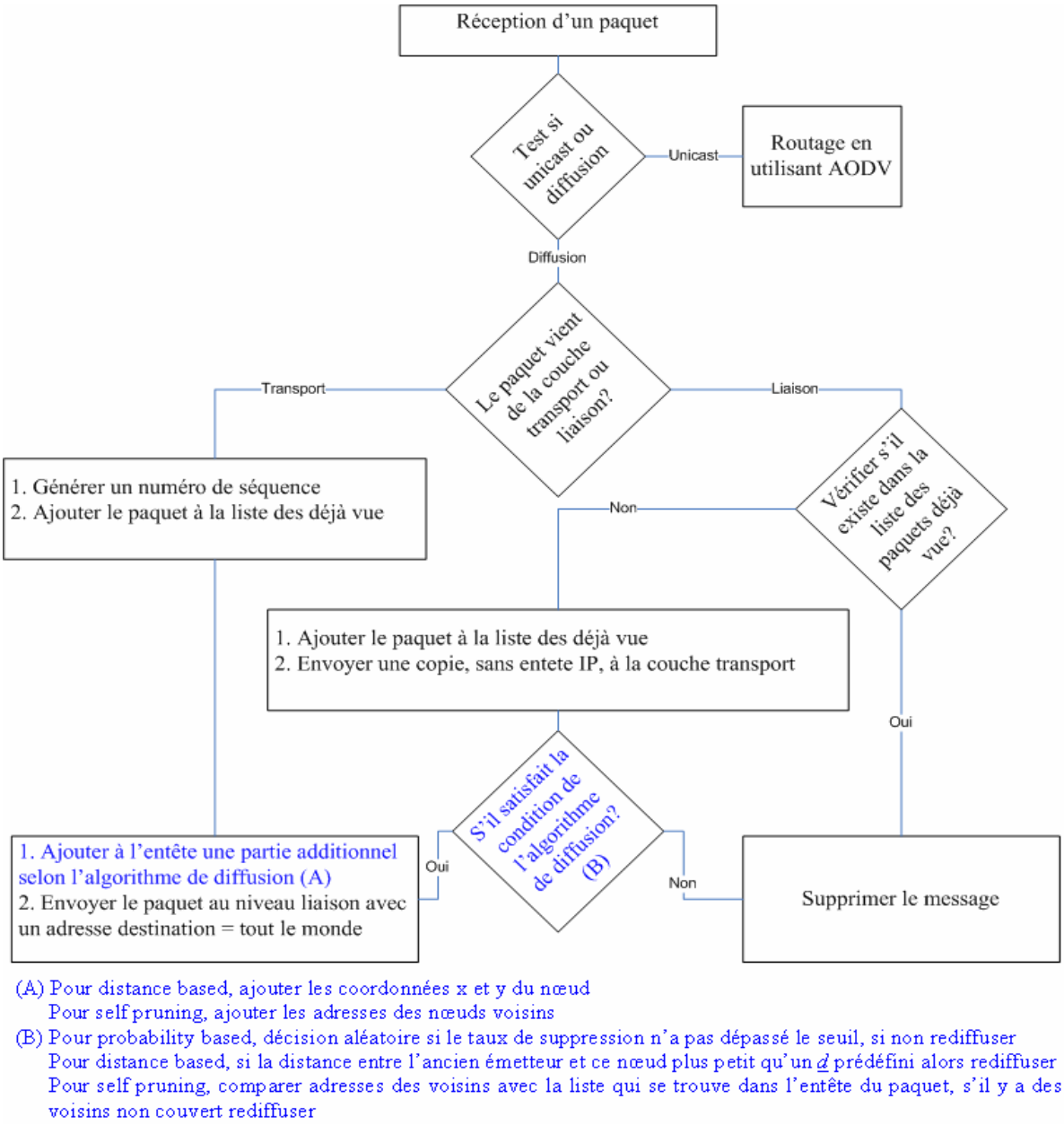

Figure 8 Fonctionnement de l'algorithme de routage

3.2.2 Modèle de mobilité

Afin d'implémenter un modèle de mobilité adéquat aux jeux vidéo multi-joueurs sur les réseaux ad hoc, nous avons conduit une petite enquête entre les joueurs sur les réseaux sans fil à infrastructure fixe. Nous avons cherché, comment ils s'installent au début du jeu, la fréquence de leurs mobilités avec quelles distances et vitesses et qu'est-ce qu'ils pensent de la mobilité. L'important pour chaque joueur est de s'installer dans un lieu où les autres ne peuvent pas voir son écran et quand il se déplace, même s'il se déplace rarement, il se déplace une faible distance avec une vitesse et destination aléatoire. Avant de présenter le modèle de mobilité qu'on a implémenté, en utilisant ces résultats, nous allons présenter le modèle de distribution initiale des nœuds.

Distribution initiale des nœuds

Pour créer une session de jeu, tous les joueurs doivent être joignables. Les modèles de distribution aléatoires existants ne garantissent pas cette connectivité entre les nœuds (supposant un faible nombre de joueurs dans un terrain relativement grand). Notre modèle de distribution consiste à distribuer les nœuds aléatoirement dans la zone de la simulation, en garantissant au même temps une connectivité entre les nœuds. Ainsi, un nœud est sûrement dans la zone de couverture d'au moins un autre nœud.

Modèle de mobilité

Nous avons développé un modèle de mobilité basé sur les modèles de mobilité aléatoire. Un nœud choisi sa direction et le temps de repos entre deux déplacements aléatoirement. La vitesse de mobilité est choisie aléatoirement entre une vitesse minimale et une vitesse maximale. Pour limiter la distance de déplacement, le nœud choisit aléatoirement une distance de déplacement en dessous d'une borne supérieure, définit par l'utilisateur. La fréquence de mobilité de nœud est affaiblie dans le code sans influencer le mode aléatoire du déplacement.

3.3 Paramètres de simulation

Dans cette partie je présenterai les scénarios de simulation utilisés ainsi que les différents critères de performance choisis, pour évaluer la performance des algorithmes de diffusion dans les deux architectures C/S et P2P.

3.3.1 Scénario

Afin d'évaluer efficacement la performance des différents algorithmes de diffusion choisis (*Simple Flooding*, *Distance Based*, *Probability Based* et *Neighborhood Based*) nous avons essayé de créer le contexte le plus réaliste possible.

Nous avons considéré que ce type de jeux se joue souvent dans une cour de recréation, voire un terrain vague. Nous prendrons ici une surface qui sera en quelque sorte une limite maximale concevable pour ce genre de simulation. Prenons donc par exemple un terrain qui fait (300*300) m^2 de surface. Nous avons configuré chaque nœud par une zone de couverture de 120m et un débit de 2Mbps.

Nous avons simulé 4 scénarios. La différence entre ces scénarios est le nombre de nœuds (4, 8, 12 et 16 joueurs). Nous croyons que ce nombre de nœuds représente approximativement le nombre de joueurs possible dans une session de jeux vidéo multi joueurs sur MANET. Varier le nombre de nœuds nous permet d'évaluer les algorithmes de diffusion choisis avec des réseaux de différentes densités (faible densité à 4 nœuds jusqu'à la haute densité à 16 nœuds). Comme nous l'avons vu dans la partie « Diffusion sur MANET », la densité du réseau influence quelques algorithmes de diffusion comme *Distance Based* et *Probability Based*.

Pour la distribution initiale des nœuds et leurs mobilités nous avons utilisé les modèles de distribution et mobilité présentés dans la partie précédente.

Dans une architecture C/S le modèle de trafic peut être résumé par : un nœud (serveur) qui diffuse vers tous les autres (les clients) et tous les autres lui envoient du trafic unicast. Les auteurs de [21] donnent le résultat d'une expérimentation de 36 heures sur un jeu d'architecture C/S de type FPS (Counter/Strike). Ils ont trouvé que le serveur diffuse des *Constant Bit Rate* (CBR) de 127 octets tous les 62ms et que chaque client lui envoie des unicast CBR de 82 octets tous les 40ms.

En une architecture P2P, chaque nœud diffuse vers tous les autres. Pour comparer cette architecture avec l'architecture C/S il faut utiliser le même jeu ou au moins le même type de jeu. Malheureusement, aucune expérimentation ni proposition n'existe dans la littérature. L'architecture réseau est transparente par rapport aux joueurs humains. Le joueur joue de la même manière donc dans les deux architectures. De cette logique, nous avons estimé que le joueurs derrière une machine client ou une machine paire va envoyer le même trafic. Chaque paire envoie donc des CBR de 82 octets toutes les 40 ms.

Simple Flooding et Self Pruning, ces deux algorithmes de diffusion ne demandent pas de configurations spéciales par rapport aux scénarios choisis. Pourtant, comme nous l'avons vu dans la section « Diffusion sur MANET » les seuils en *Probability Based* et *Distance Based* sont liés à la densité du réseau. En *Probability Based*, nous avons choisi deux probabilités de transmission 0,4 et 0,8, donc typiquement avec une faible et haute probabilité. Ces valeurs vont nous permettre de choisir la probabilité adéquate pour un contexte de jeu. Pour l'algorithme *Distance Based*, le seuil

est en plus limité par la couverture radio du nœud. Nous avons simulé cet algorithme avec trois valeurs de distance seuil, *d*=40m, *d*=70m et *d*=100m.

Table 1 résume les paramètres de simulation choisis.

Paramètres		**Valeurs**	
Simulateur		Glomosim-2.0	
Temps de simulation		300second	
Surface de terrain		300*300m^2	
Distribution initiale des nœuds		Notre modèle	
Mobilité	1ère simulation	Notre modèle de mobilité	
	2éme simulation	Sans mobilité	
Porté radio de nœud		120metre	
Débit		2Mbps	
Trafic	Serveur	127 octet CBR toutes les 62 ms	
	Client	82 octet CBR toutes les 40 ms	
	Pair	82 octet CBR toutes les 40 ms	
Routage unicast		AODV	
Diffusion		*Simple Flooding*	
		Self Pruning	
		Probability Based	*p*=0.4
			p=0.8
		Distance Based	*d*=40m
			d=70m
			d=100m

Table 1 Paramètres de simulations

3.3.2 Critères de performance

Afin d'évaluer la performance des algorithmes de diffusion dans les différents scénarios, les critères de performance suivants ont été utilisés :

- Délai de bout en bout : c'est le délai de transmission moyen d'un paquet de la source jusqu'à la destination. Pour calculer ce délai nous avons utilisé les informations données par Glomosim au niveau application. La couche application donne la moyenne de l'ensemble des délais pris par l'ensemble des paquets reçus;

- Taux de livraison : c'est le pourcentage des paquets reçus avec succès par la destination, par rapport à l'ensemble des paquets envoyés. Pour le calculer on cherche tout d'abord le nombre total de paquets envoyés puis pour chaque nœud on calcule le pourcentage des paquets reçus. Une fois que nous avons le pourcentage de livraison de paquets pour chaque nœud, on cherche la moyenne des pourcentages obtenus, pour avoir un pourcentage moyen;

- Saved ReBroadcast (SRB) : ce critère nous permet d'évaluer l'algorithme de diffusion en termes de trafic réseau. Il représente le pourcentage des messages non rediffusé par des nœuds intermédiaire. Il est définit par [(r-t)/r]*100, r représente le nombre de message à diffuser, t est la moyenne de rediffusion par nœud. Plus petit le SRB mieux est le résultat;

- Consommation de l'énergie : ici on calcule l'énergie en Joule consommée par chaque nœud pour satisfaire les émissions/réceptions nécessaires pendant une session de jeu. Ce calcul se base sur la spécification de l'IEEE 802.11 *WaveLAN-II* [22] de Lucent. Pendant une **émission** le nœud consomme :

$$\frac{1.50 watt \otimes T}{BP}$$

T: taille de paquets en bit
BP : Bande Passante en bit par second
Résulta obtenu est en joule

Pour la **réception**, un nœud consomme:

$$\frac{1.25 watt \otimes T}{BP}$$

3.4 Résultats de simulation

Dans cette partie je présenterai les résultats des simulations en termes de taux de livraison, délai et consommation d'énergie, dans les deux architectures C/S et P2P sans et avec mobilité. Les résultats des simulations sans mobilité sont utilisés comme référence pour comprendre l'influence de la mobilité sur ces critères de performance. Pour faciliter l'explication j'ai utilisé des abréviations qui représentent certains algorithmes : D40 = *Distance Based* avec un seuil= 40 mètre, D70 seuil = 70 mètre, D100 seuil=100 mètre, P04 = *Probability Based* seuil=0.4, P08 seuil = 0.8, Self= *Self Pruning* et Simple pour *Simple Flooding*.

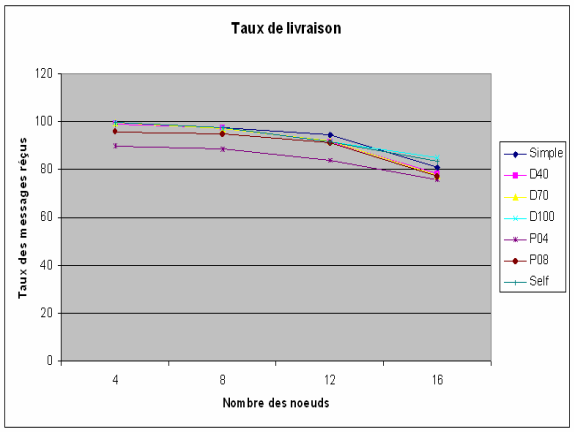

Figure 9 Taux de livraison de paquets dans une architecture C/S sans mobilité

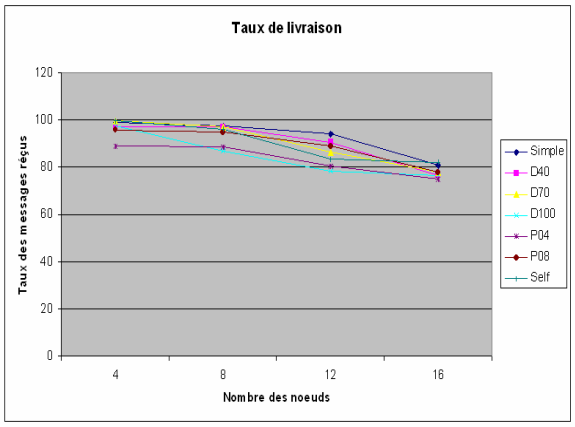

Figure 11 Taux de livraison des paquets dans une architecture C/S avec notre modèle de mobilité

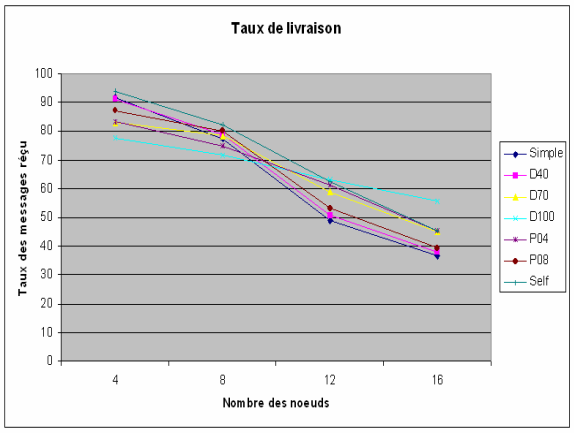

Figure 10 Taux de livraison des paquets dans une architecture P2P sans mobilité

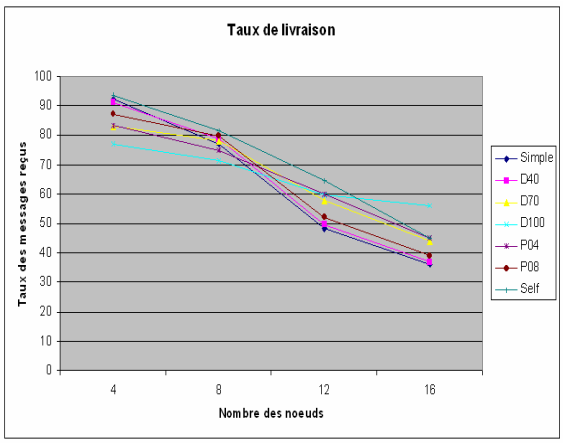

Figure 12 Taux de livraison des paquets dans une architecture P2P avec notre modèle de mobilité

Les figures 9 et 11 représentent respectivement le taux moyen des messages reçus par chaque nœud dans une architecture C/S avec et sans mobilité. Les taux de livraison varient entre 100% et 75%. En appliquant notre modèle de mobilité, nous remarquons (cf. figure 11) que dans les réseaux de faible densité (4 et 8 nœuds) la majorité des algorithmes montrent des résultats prometteurs sauf (P04 et D100). P04 et D100, grâce à leur principe de fonctionnement expliqué en « Etat de l'art », ne sont pas prévus de données des bons résultats dans les réseaux de faible densité. Nous pouvons remarquer aussi que plus le réseau est dense plus le taux de livraison est faible. En se basant sur le fait que dans les réseaux denses il y a plus de collision et contention, nous pouvons justifier cette dégradation du taux de livraison au niveau des réseaux de 12 et 16 nœuds. La Figure 11 montre que le *Simple Flooding* est le meilleur par rapport aux autres algorithmes. D40, D70 et P08 ont donnée

des résultats intéressants. L'algorithme *Self Pruning*, malgré la dégradation de taux de livraison dans un réseau de 12 nœuds par rapport à 8 nœuds, il a donné le meilleur résultat dans un réseau de 16 nœuds.

Pour comprendre l'influence de la mobilité sur le taux de livraison, nous avons comparé les figures 9 et 11. On peut remarquer que la dégradation causée par la mobilité ne se voit vraiment qu'au niveau des réseaux de 12 nœuds. Au niveau des algorithmes, seul D100 est vraiment influencé.

Dans le cas des réseaux P2P (figure 10 et 12), les résultats prouvent que plus le réseau est dense plus on est loin de la qualité de service souhaitée. Pour justifier cette grande différence entre les 2 architectures nous reviendrons sur le nombre d'événements générés dans chacune. en P2P tous les nœuds génèrent des messages de diffusions tous les 40 ms alors que seul le serveur génère des messages de diffusion dans le contexte de C/S. Nous pensons que, ce grand nombre des diffusions à conduit à des taux de collision et contentions élevés qui diminuent le taux des messages reçus.

Malgré les "mauvais résultats" de P2P nous pouvons dire que jusqu'a présent D100 et *Self Pruning* ont été mieux que les autres algorithmes. Dans cette architecture comme montre les figures 10 et 12 la mobilité n'a pas une vraie influence sur le taux de livraison offert par chaque algorithme.

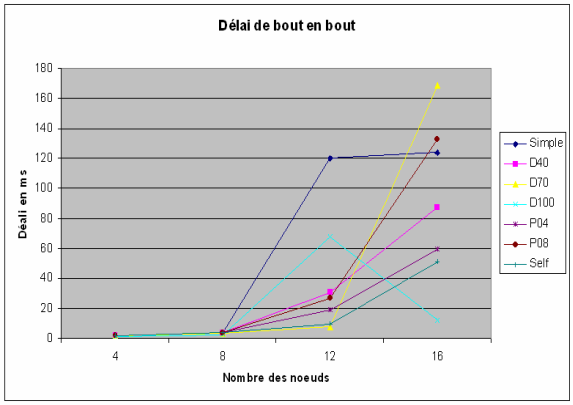

Figure 13 Délai de bout en bout dans une architecture C/S sans mobilité

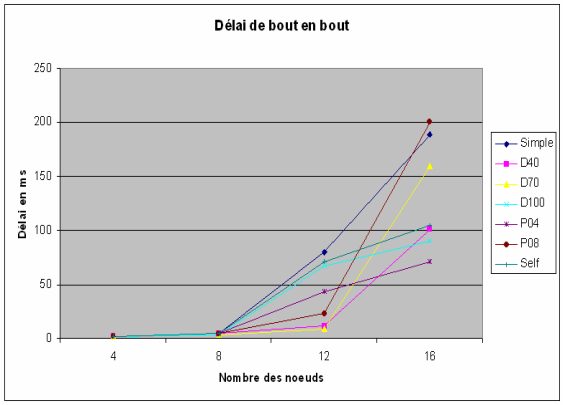

Figure 15 Délai de bout en bout dans une architecture C/S avec notre modèle de mobilité

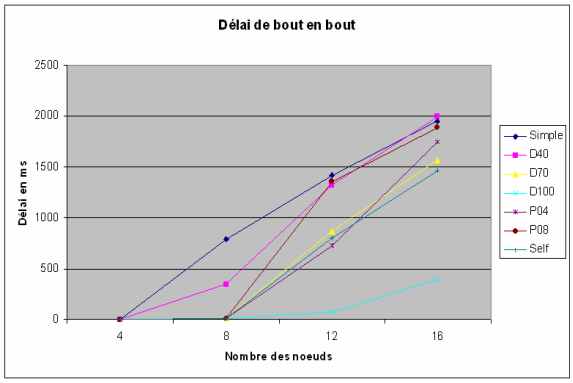

Figure 14 Délai de bout en bout dans une architecture P2P sans mobilité

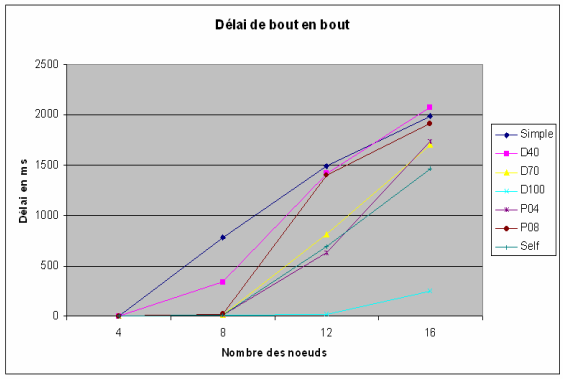

Figure 16 Délai de bout en bout dans une architecture P2P avec notre modèle de mobilité

Les figures 13, 14, 15 et 16 donnent les performances en terme de délai dans les différents contextes simulés (C/S et P2P sans et avec mobilité). Pour un jeu de FPS nous supposons que plus le délai est en dessus de 100ms moins le jeu est jouable. Pour les réseaux de faible densité (4 et 8 nœuds), les résultats obtenus montrent que la majorité des algorithmes donnent des bons résultats dans tous les cas simulés. La différence entre les deux architectures se voit clairement à 12 nœuds où les algorithmes en C/S restent prometteurs tandis qu'en P2P ils ne répondent pas aux attentes.

Dans une architecture C/S et avec16 nœud, la qualité de service, en termes de délai offert par les différents algorithmes se dégradent fortement. Certains algorithmes comme *Simple Flooding*, D70 et P08 donnent des résultats insupportables dans un contexte de jeu type FPS. L'algorithme P04 est le meilleur pour cette densité. Le *Self Pruning* et D100 restent acceptables avec ce nombre de nœuds. Dans cette architecture la mobilité a dégradé le service mais les résultats en général sont restés acceptables.

Dans les réseaux denses (12 et 16 nœuds) et une architecture P2P, seuls les résultats de D100 sont prometteurs. Dans ce protocole l'optimisation du modèle de trafic nous semble le premier pas à faire pour résoudre les problèmes de délai et pertes de paquets. La mobilité (cf. figure16) a avantagé de plus le D100 sans vraiment influencer les autres algorithmes.

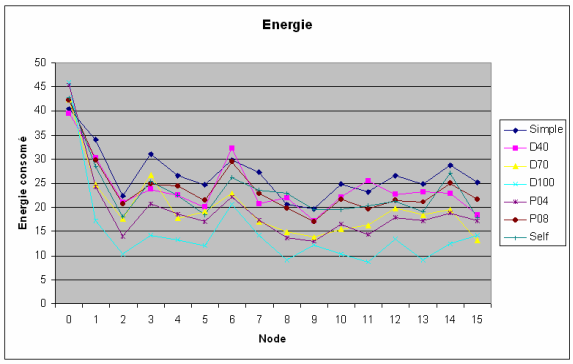

Figure 17 Consommation d'énergie par nœud dans un réseau de 16 nœuds, architecture C/S sans mobilité

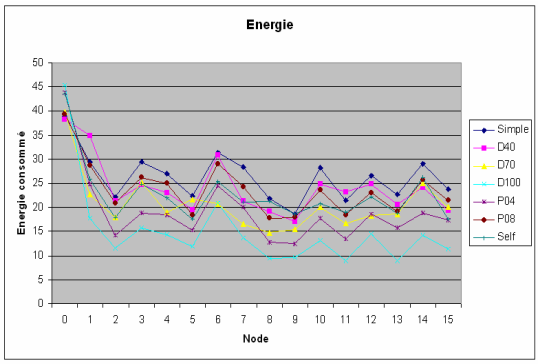

Figure 19 Consommation d'énergie par nœud dans un réseau de 16 nœuds, architecture C/S avec notre modèle de mobilité

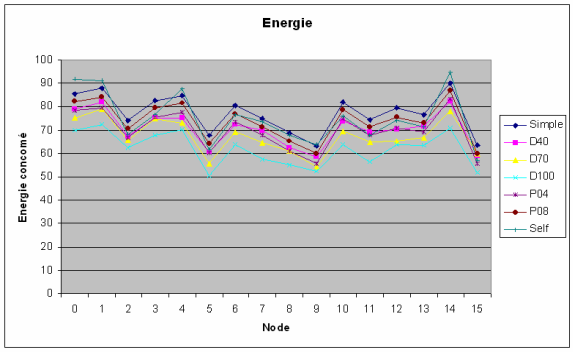

Figure 18 Consommation d'énergie par nœud dans un réseau de 16 nœuds, architecture P2P sans mobilité

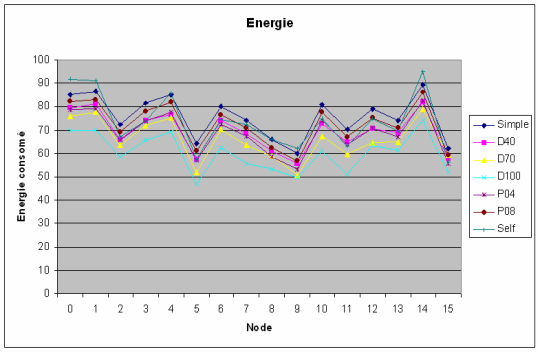

Figure 20 Consommation d'énergie par nœud dans un réseau de 16 nœuds, architecture P2P avec notre modèle de mobilité

Au niveau de la consommation d'énergie, nous présentons seulement dans cette partie la distribution des nœuds dans un réseau de 16 nœuds. Les figures 17, 18, 19 et 20 représentent cette consommation dans les différentes architectures simulées avec et sans mobilité. Les résultats prouvent que D100 est le meilleur dans tous les contextes en termes de consommations d'énergie.

L'algorithme de *Self Pruning* consomme plus d'énergie pour la construction et le maintien de ces tables de voisinages. Malgré cet aspect, en C/S la consommation du *Self Pruning* est restée acceptable par rapport aux autres algorithmes. Pourtant en P2P, *Self Pruning* ne semble pas être une bonne solution en termes de consommation d'énergie. D'un autre coté, nous pourrons voir clairement que la mobilité ne cause pas des consommations supplémentaires importantes.

Comme il a été prévu en C/S le serveur a consommé plus d'énergie que les autres nœuds (cf. figures 17 et 19), et les pairs dans une architecture P2P ont consommé plus d'énergie (cf. figures 18 et 20) que les nœuds dans une architecture C/S.

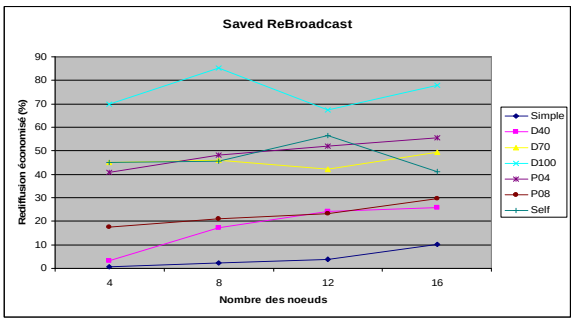

Figure 21 SRB dans une architecture C/S sans mobilité

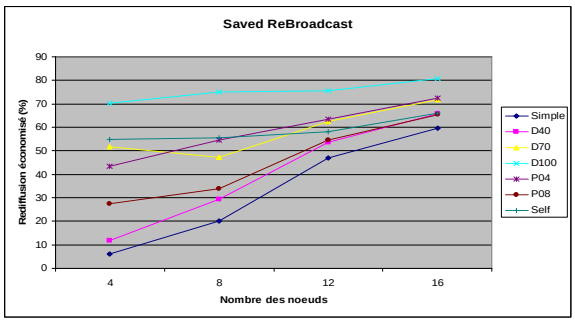

Figure 23 SRB dans une architecture P2P sans mobilité

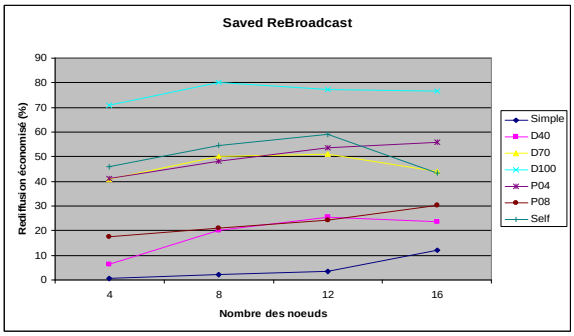

Figure 22 SRB dans une architecture C/S avec mobilité

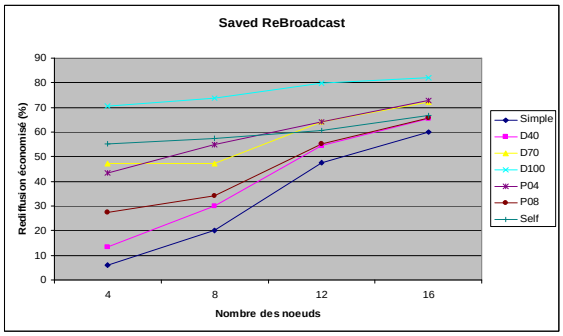

Figure 24 SRB dans une architecture P2P avec mobilité

Diminuer le nombre de rediffusions est un défit dans notre contexte. Cette diminution nous permet de réduire le trafic réseau non utile, d'éviter des collisions et donc de gagner de temps de communication de bout en bout. Le SRB le plus élevé est important à condition de ne pas influencer le taux de livraison demandé dans un contexte de jeu vidéo multi joueurs sur MANET.

Dans les figures (21, 22, 23 et 24), nous utilisons le SRB pour évaluer la performance des algorithmes de diffusion simulés dans les deux architectures C/S et P2P. Il faut dire que ce critère est influencé par le taux de livraison. Un nœud qui ne reçoit pas un message ne va pas le rediffuser. Cette influence se voit clairement quand on regarde le résultat de *Simple Flooding*. Dans un contexte idéal sans perte en *Simple Flooding* le SRB = 0 (tous les nœuds rediffusent tout ce qu'ils reçoivent pour la première fois). Les résultats obtenus montrent des SRB plus grand que 0 pour *Simple Flooding*. Si on regarde le taux de livraison (figures 9, 10, 11 et 12) on remarque que quand on a une forte perte on a une haute SRB et vice versa, ce qui explique pourquoi le SRB est plus grand que 0 en *Simple Flooding*.

Si on regarde les performances de *Distance Based* avec un *d*=100 en SRB et en taux de livraison on remarque qu'il est toujours l'un des meilleurs algorithmes et c'est le cas pour *Self Pruning*, *Probability Based* (avec *p*=0,4) et *Distance Based* (avec d=70).
Au niveau des architectures, ces résultats remontrent que jusqu'au présent, l'architecture C/S (voir figure 21 et 22) est meilleure que P2P (voir figure 21 et 22) dans un contexte de jeux vidéo sur MANET.

4. Conclusion et perspectives

Dans ce stage nous avons essayé de proposer un nouvel protocole de routage unicast/Broadcast qui satisfait les besoins des jeux vidéo multi joueurs sur réseaux ad hoc, en termes de QoS. Une grande étude bibliographie détaillée nous a permis de comprendre tout d'abord les contraintes de QoS nécessaires dans ce type des réseaux. L'étude bibliographie sur les architectures réseaux utilisés dans les jeux nous a permis de comprendre le modèle de communication dans ce genre des réseaux et de savoir l'importance de la diffusion.

Comment optimiser la diffusion dans un contexte de jeux vidéo multi joueurs sur réseau MANET, a fait l'objet de mon travail. Pour répondre à cette question une étude bibliographie sur les diffusions optimisées en MANET était indispensable.

L'étude bibliographie nous a montré qu'un protocole de routage unicast/Broadcast efficace est un élément clé dans un contexte de QoS pour les jeux vidéo multi joueurs sur MANET. Dans la littérature les chercheurs ont proposé des améliorations sur un protocole de routage unicast (AODV) pour satisfaire les besoins de jeu. Dans ce stage nous proposons un protocole de routage qui dans la partie unicast reprend les fonctionnalités d'AODV et dans la partie Broadcast utilise un algorithme de diffusion optimisé. Nous avons évalué et analysé la performance de plusieurs algorithmes de diffusion optimisé (*Silmple Flooding*, *Probability Based*, *Distance Based* et *Self Pruning*) via des simulations sur Glomosim. Nous avons trouvé que dans une architecture client serveur le *Distance Based* et *Probability Based* avec des faibles seuils ont donné des résultats suffisants.

La performance de ces nouveaux protocoles n'était pas suffisant dans une architecture pair à pair. Après une certaine analyse des résultats nous avons trouvé que le problème vient du nombre d'événements générés dans cette architecture où chaque nœud diffuse un message toutes les 40 ms. Pour résoudre ce problème, il faut trouver un modèle de trafic adéquat pour l'architecture pair à pair qui diminue le nombre d'événement. Pourtant, proposer un modèle de trafic fera partie des travaux futurs.

Nous comptons également proposer des seuils dynamiques qui varient automatiquement selon la densité du réseau pour les algorithmes *Probability Based* et *Distance Based*.

5. Liste des abréviations

AODV	Ad hoc On demand Distance Vector
C/S	Client Server
CBR	Constant Bit Rate
FPS	First Person Shooter
GPS	Global Positioning System
MANET	Mobile Ad hoc NETwork
NGMM	Networked Game Mobility Model
P2P	Peer to Peer
RTS	Real Time Strategy
RWP	Random Way Point
SRB	Saved ReBroadcast

6. Références

[1] Pierre CHECA, Sidimohamed SENOUCI, "Jeux vidéo sur réseaux ad-hoc". *Rapport technique France Telecom*. Août 2006

[2] 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

[3] 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

[4] 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

[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] Sebastian Matas Riera, Oliver Wellnitz, Lars Wolf, "A Zone-based Gaming Architecture for Ad-Hoc Networks". *In proceedings of the 2nd Workshop on Network and System Support for Games, NETGAMES 2003, Redwood City, California, USA.* May 2003

[8] Oliver Welnitz, Lars Wolf, "Assigning Game Server Roles in Mobile Ad-Hoc Networks". Proceedings of the *16th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'06), Newport, USA.* Mai 2006

[9] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network". *In Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, Seattle, Washington, United States.* 1999

[10] H. Lim and C. Kim, "Multicast tree construction and flooding in wireless ad hoc networks". In MSWIM ’00: Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems. Pages 61–68, New York, NY, USA. 2000

[11] W. Peng and X. Lu, "Ahbp: An efficient broadcast protocol for mobile ad hoc networks". J. Comput. Sci.Technol., 16(2):153–167, 114-125

[12] W. Peng and X.-C. Lu, "On the reduction of broadcast redundancy in mobile ad hoc networks". In MobiHoc ’00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing, pages 129–130, Piscataway, NJ, USA. 2000

[13] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying: An efficient technique for flooding in mobile wireless networks". Technical Report Research Report RR-3898, INRIA. february 2000

[14] Brad Williams, Tracy Camp, "Comparison of broadcasting techniques for mobile ad hoc networks". In Proceedings of the third ACM international symposium on Mobile ad hoc networking & computing. June 2002

[15] "Simplified multicast forwarding for manet". IETF Internet-Draft. March 2007

[16] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research". Wireless Communications & MobileComputing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, no. 5, pp. 483–502, 2002

[17] Mirco Musolesi, Cecilia Mascolo, "A Community Based Mobility Model for Ad Hoc Network Research". REALMAN’06. May 26, 2006, Florence, Italy

[18] Swee Ann Tan, William Lau, Allan Loh, "A MANET Mobility Model Based on First-Person-Shooter Games". 2005 Asia-Pacific Conference on Communications, Perth, Western Australia, 3 - 5 October 2005

[19] 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 Third Annual Conference on Wireless On demand Network Systems and Services (WONS 2006) Les Ménuires, France. January 2006

[20] U. P. C. Laboratory and W. A. M. Laboratory, "Glomosim: A scalable simulation environment for wireless and wired network systems". In Proceedings of the 3rd International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (Het-Net’05). 2002

[21] Johannes Färber, "Network Game Traffic Modelling". In proceedings of ACM Network and System Support for Games Workshop (NetGames2002). Braunschweig, Germany.April 2002

[22] A. Kamerman and L. Monteban. "Wavelan-ıı: A high-performance wireless lan for the unlicensed band". In Bell Labs Technical Journal, pages 118–133,Summer 1997

[23] Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561. IETF Network Working Group. July 2003.

[24] Ietf manet wg (mobile ad hoc network). IETF, www.ietf.ora/html.charters/manet-charter.html

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

ANNEXES

A. Les réseaux Wifi en mode ad-hoc

A.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 "mono-saut", c'est à dire que l'on essaie d'entrer en contact directement avec le destinataire, et si on n'y arrive pas on jette les paquets. Dans un réseau ad-hoc dit "multi-sauts", si un nœud A n'arrive pas à joindre directement son destinataire B, il enverra le paquet à une station tierce et le paquet ira de nœud en nœud 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 nés deux catégories de protocoles. Il y a les algorithmes proactifs et les algorithmes réactifs.

A.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 nœud connaît la topologie du réseau. Ici, sera présenté les protocoles OLSR, TBPRF et DSDV.

A.2.1 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, i.e. le nombre de nœud à traverser pour atteindre un autre nœud.

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

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

Les nœuds 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 nœud doit élire le minimum de multipoints réseaux, qui sont les seuls à pouvoir retransmettre dans le réseau. La suite présente donc cette élection.

Tout nœud émet des paquets Hello présenté ci après, indiquant la liste de ces voisins. Ainsi, si un nœud 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 nœud puisse accéder à tous ses voisins à deux sauts. Pour illustrer ceci, la figure A.1 montre une situation. Nous allons nous intéresser au cas du nœud du milieu. Celui-ci est à deux sauts de tous les nœuds, 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 nœud central. Si on élit C et D, on se rend compte que les nœuds 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 nœuds 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 nœuds utilisent une heuristique.

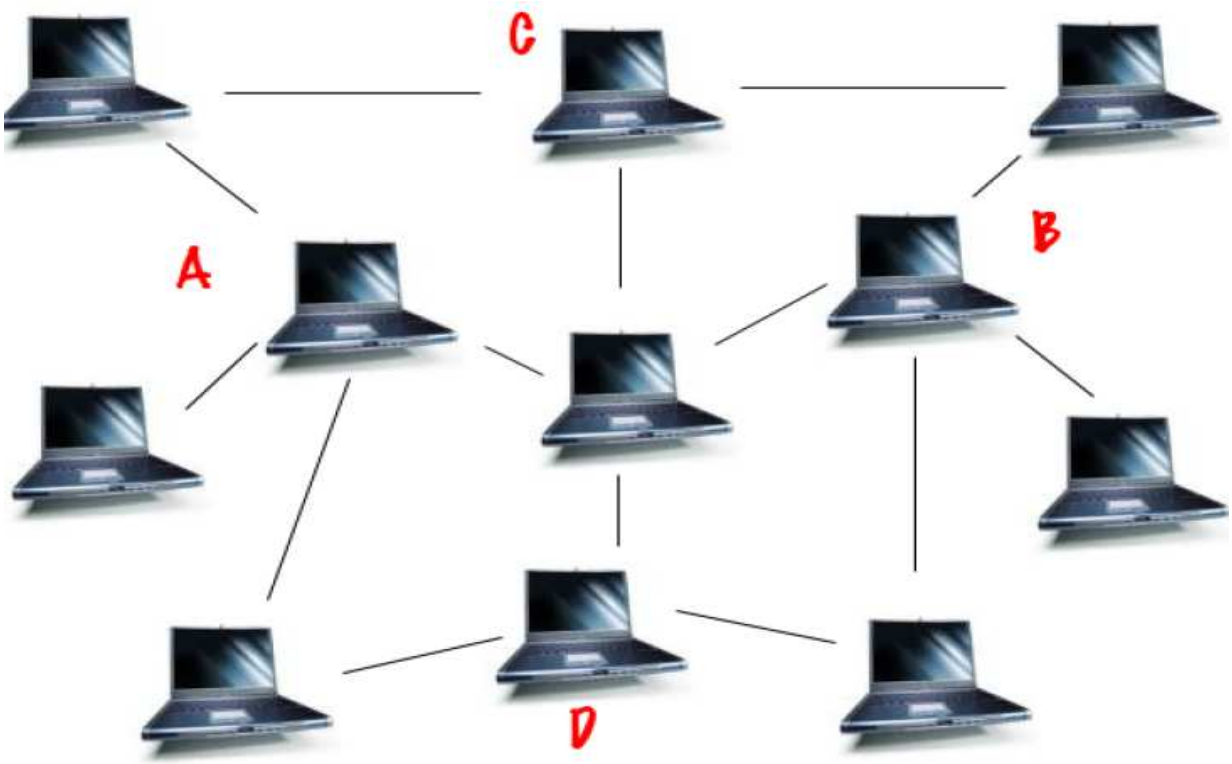

Figure A. 1 L'élection des multipoints relais

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

A.2.2 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 [25].

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. [1]

Pour maintenir les arbres de plus court chemin de chaque nœud valide, un nœud, détectant une modification de topologie, envoie un message de mise à jour aux autres nœuds. Pour éviter le bouclage de ce genre d'informations dans le réseau, le mécanisme illustré par la figure A.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 nœuds internes de l'arbre retransmettent cette modification. Pour ce faire, chaque nœud, à la réception du message de A calcule l'arbre associé à A et décide s'il est une feuille ou un nœud de l'arbre. Dans le cas ou c'est un nœud et seulement dans ce cas, il émet le paquet. Dans notre exemple, seuls les nœuds C et G retransmettent aux nœuds E, F et H.

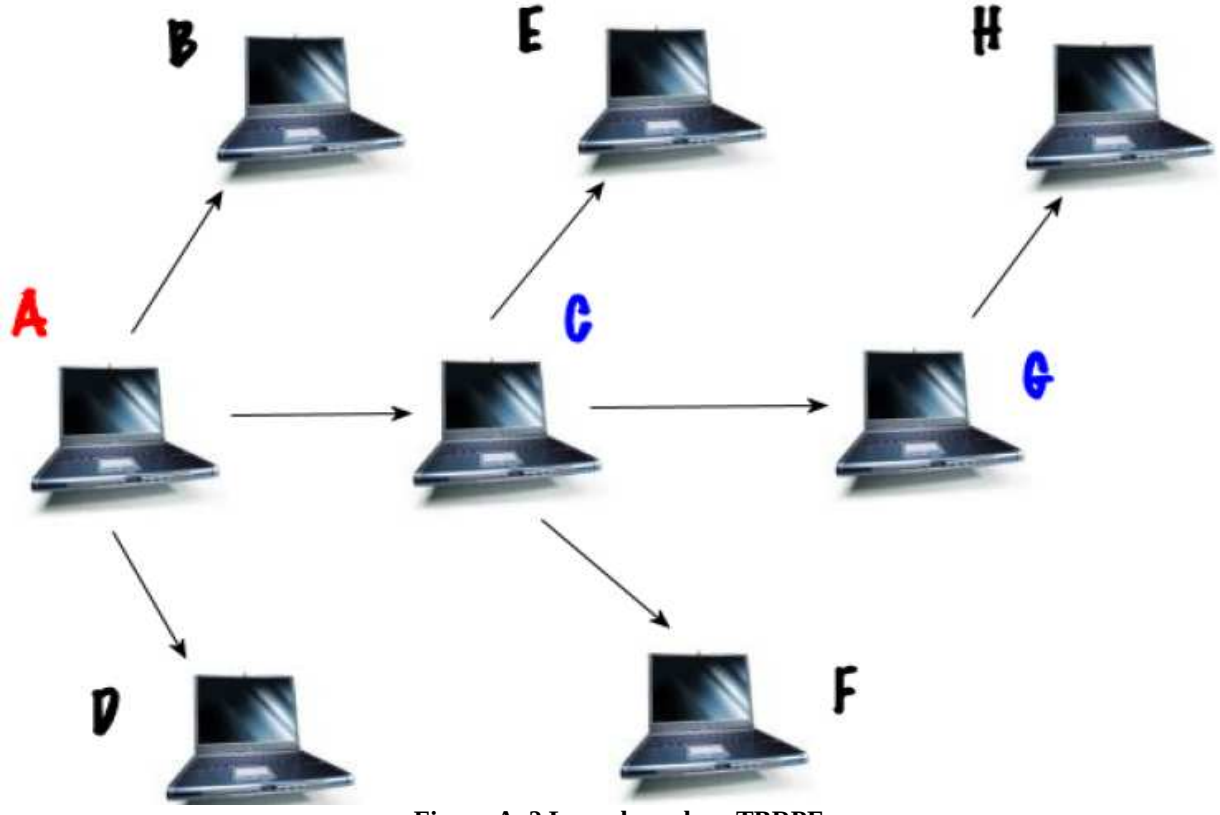

Figure A. 2 Les arbres dans TBRPF

A.2.3 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 nœud 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.

A.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 nœud le demande. Les nœuds ne connaissent pas la topologie du réseau à tout moment, ils définissent le chemin à utiliser dans le cas où un nœud 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 nœud veut émettre vers un destinataire inconnu. Il envoie une requête RREQ en broadcast sur le réseau.

Chaque nœud 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 nœuds 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 nœud reçoit des paquets du nœud suivant dans la route. Si le nœud trouve des paquets de ce nœud, cela signifie que le lien est valide.

Si les deux tests échouent, alors DSR réémet le paquet en demandant un acquittement (AckReq). Le nœud suivant devra répondre par un acquittement. En cas d'échec, le nœud 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.

A.3.1 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 [23].

Le protocole AODV implémente les types de messages suivants :

- Route Request (RREQ)
- Route Reply (RREP)
- Route Error (RERR)
- Route Reply Acknowledgement (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 nœud 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 constituée 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 nœud 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 nœud 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 nœud qui a bougé et n'est plus atteignable. L'efficacité de ce mécanisme dépend de la mobilité des nœuds.

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.

A.3.2 DSR : Dynamic Source Routing

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

De plus, DSR utilise la source routing. Dans les faits, quand un nœud 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 nœuds qui voit passer le paquet, ajoute son adresse à cette liste des nœuds 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 les 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 nœud 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 A.3.

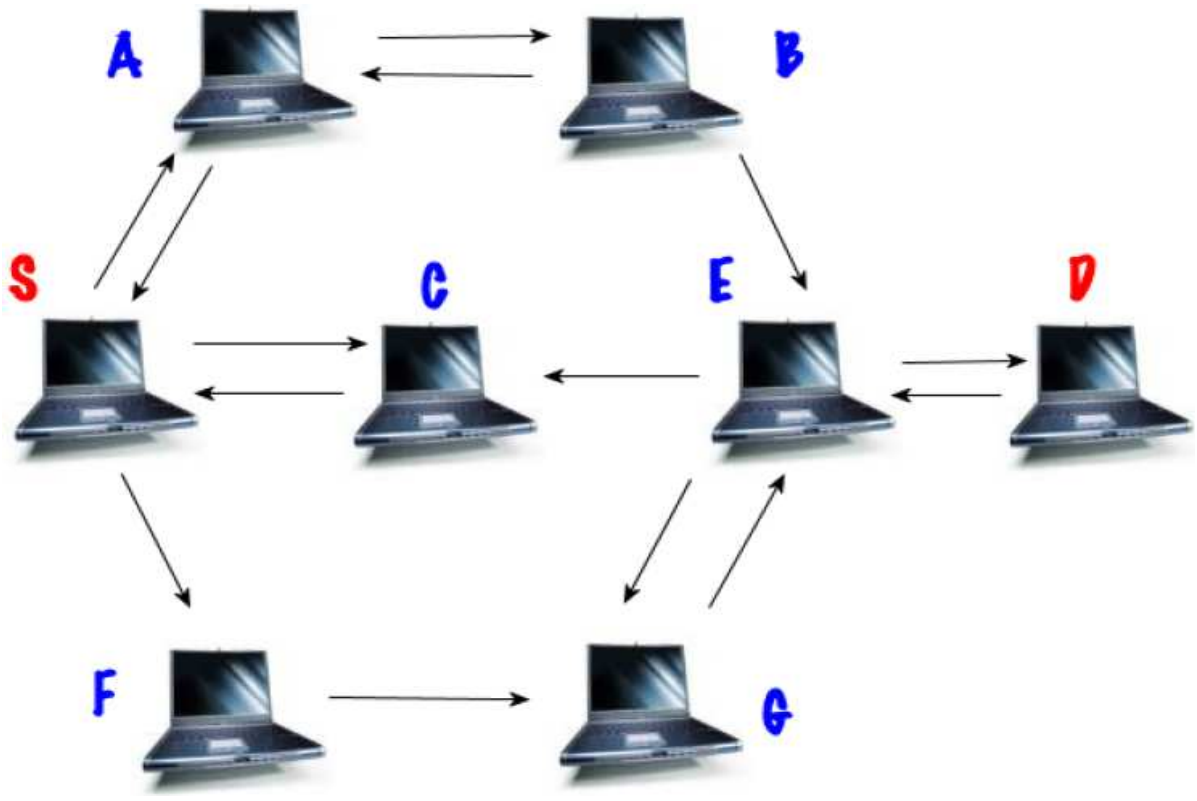

Figure A. 3 Le fonctionnement de DSR

Le nœud source S veut envoyer un paquet au nœud destination D. Les flèches entre les nœuds signifient que les nœuds 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, i.e. 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 nœud 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 résout donc le problème de l'équilibrage des charges du réseau [25].

A.3.3 Conclusion

Vis à vis des deux protocoles présentés ici, on peut indiquer qu'originellement AODV 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.

A.4 Un algorithme hybride : ZRP (Zone Routing Protocol)

A travers les deux sections précédentes, nous avons présenté deux approches qui avaient chacune 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 nœud créé une zone autour de lui qui correspond à tous les nœuds 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 nœud 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 nœud source demande aux nœuds 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 nœuds bordures demandent à leurs nœuds bordures respectifs et ainsi de suite jusqu'à trouver la destination. Pour éviter la surcharge du réseau, chaque nœud bordure traite une recherche que s'il ne vient pas juste de la faire [25]