République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira de Bejaia
Faculté des Sciences Exactes
Département d'Informatique

ECOLE DOCTORALE RESEAUX ET SYSTEMES DISTRIBUES

Mémoire de Magister

En Informatique

Option : Réseaux et Systèmes Distribués

Thème

Prevention and Detection of Cheating in Peer-to-Peer Online Games

Présenté par

DJEDJIG Nabil

Devant le jury composé de :

Président	TARI Abdelkamel	MCA	Université de Bejaïa
Rapporteur	KHELALFA Halim	MC (HDR)	Université de Wollongong
Examinateur	BAGHDAD Rachid	MCA	Université de Bejaïa
Examinateur	DJENOURI Djamel	MRA	CERIST, Alger
Invitée	BENMEZIANE Souad	CR	CERIST, Alger

Promotion : 2008/2009

REMERCIEMENTS

Je remercie en premier notre grand Dieu pour m'avoir donné le courage et la volonté pour terminer ce modeste travail.

J'adresse tout d'abord mes remerciements à mes directeurs de recherche, pour avoir accepté de m'encadrer et m'avoir aidé à conduire ce travail jusqu'au bout : Monsieur Halim KHELALFA, maître de conférence à l'université Wollongong de Dubaï pour m'avoir proposé le sujet et donné l'occasion de travailler sur une thématique pleine de perspectives, aussi pour son aide appréciable, son suivi continu, ses orientations et ses remarques pertinentes.

Je remercie Madame Souad BENMEZIANE, chargée de recherche au CERIST pour m'avoir orienté, guidé, corrigé mon travail et mis à ma disposition tous les moyens pour la finalisation de ce travail, une riche documentation, soutenu et supporté dans des moments délicats, je lui suis profondément reconnaissant.

Je remercie : Mr. Abdelkamel TARI, Mr. Rachid BAGHDAD et Mr. Djamel DJENOURI, pour avoir accepté de juger ce modeste travail.

Un grand MERCI à Mes Parents, Ma Femme, Mes sœurs et Mes frères pour m'avoir soutenu afin que ce travail arrive à sa fin.

Je remercie ma belle-famille pour m'avoir encouragé.

Je remercie mes amis pour leur aide appréciable, leurs encouragements continus et leur soutien moral ininterrompu.

Je voudrais remercier vivement mes collègues du CERIST, pour m'avoir poussé à faire le magister, encouragé, motivé et surtout soutenu et aidé avant et durant la préparation de mon mémoire.

Et pour être sûr de n'oublier personne, que tous ceux, qui de près ou de loin, ont contribué par leurs conseils, leurs encouragements ou leur amitié, à l'aboutissement de ce modeste travail, trouvent ici l'expression de ma profonde reconnaissance.

RÉSUMÉ

Les jeux en ligne, particulièrement les jeux en ligne massivement multi-joueurs (MMOG) sont confrontés de plus en plus au risque d'être attaqués par des pirates et des tricheurs qui veulent acquérir une position dominante et maintenir un avantage déloyal par rapport aux autres joueurs. Traditionnellement, les tricheurs peuvent attaquer toutes les parties impliquées dans un jeu en ligne : les clients, le(s) serveur(s) et le réseau de communication.

La plupart des recherches dans la prévention de la tricherie dans les jeux en ligne multi-joueurs a été axée sur un modèle client/serveur. Néanmoins, les systèmes MMOG sont basés sur une architecture plus complexe, qui utilise des serveurs basés sur une architecture P2P, où une partie de la charge de jeu est stockée localement sur l'ordinateur du joueur.

Dans un MMOG basée sur P2P, les joueurs sont affectés à des groupes dynamiques. Un joueur sera affecté à un groupe qui inclut les joueurs et les objets dans son domaine visible.

L'objectif de ce travail est de :

- Etudier les différents types de tricherie dans les jeux en ligne, dans les traditionnels jeux C/S (client/serveur) comme dans l'émergente architecture basée sur P2P.
- Etudier les approches courantes de prévention/détection de la tricherie dans les MMOG P2P.
- Proposer une nouvelle approche au problème, en prenant compte des facteurs de sécurité tels que la cohérence, la confidentialité et l'authentification, et de performance tels que l'équilibrage de charge, un temps de transfert réduit, un temps de réponse minimisé et assurer l'équité.

Mots clés : MMOG, Tricherie, P2P, C/S, Tricheur, Attaque, Prévention, Détection.

ABSTRACT

Online games, especially massively multi-player online games (MMOG) are increasingly facing the risk of being attacked by hackers and cheaters who want to achieve a dominating position and hold unfair advantages over the other players.

Traditionally, cheaters can attack all the parties involved in an online game: the clients, the server(s), and the underlying communication network.

Most of the research in the prevention of cheating in multi-player online games has focused on a Client/server model. However, MMOGs systems are based on a more complex architecture that uses not only multiple servers, but also is based on a P2P architecture, where part of the game load is stored locally on the player's computer.

In a P2P based MMOG, players are assigned to dynamic groups. A player will be assigned to a group that includes the players and objects within his/her visible range.

The objective of this work is to:

- Review the different types of cheating in online games in traditional C /S based games as well as in emerging P2P based architecture.
- Review the current approaches to prevent/detect cheating in P2P based MMOG
- Propose a new approach to the problem taking into account both the security as well as the performance factor.

Keywords: MMOGS, Cheating, P2P, C/S, cheater, attack, Prevention, Detection.

الملخص

الألعاب عبر الإنترنت, وخاصة في الألعاب عبر الإنترنت متعددة اللاعبين على نطاق واسع **(أملو)** تواجه بشكل متزايد خطر التعرض لهجوم من قبل القراصنة والغشاشين الذين يرغبون في الحصول على مركز مهيمن والحفاظ على ميزة غير عادلة على اللاعبين الأخرين. تقليديا, يمكن للغشاشين مهاجمة جميع الأطراف المشاركة في اللعبة على الانترنت: الزبائن, الخادم, وشبكة الاتصالات.

وقد تركز معظم الأبحاث في مجال الوقاية من الغش في الألعاب عبر الإنترنت متعددة اللاعبين على نطاق واسع على نموذج الزبون/ الخادم **(ز/خ)** .ومع ذلك, تستند أنظمة أملو على هندسة أكثر تعقيدا, والتي تستخدم خادم يستند على هندسة زوج/ زوج **(ز/ز)** , حيث يتم تخزين محليا جزء من عبء اللعبة على جهاز كمبيوتر اللاعب.

في أملو على أساس **(ز/ز)**, يتم تعيين اللاعبين إلى مجموعات ديناميكيا .يتم تعيين اللاعب لمجموعة تضم لاعبين والكائنات في المدى المرئي.

الهدف من هذا العمل هو:

- دراسة مختلف أنواع الغش في الألعاب عبر الإنترنت, في الألعاب التقليدية الزبون/ الخادم **(ز/خ)** كما في الهندسة الناشئة على أساس**(ز/ز)**.
- دراسة النهج الحالية وقاية /كشف الغش في **أملو**.
- اقتراح منهج جديد لهذه المشكلة, مع الأخذ بعين الاعتبار عوامل السلامة مثل التناسق والسرية والمصادقة, والأداء المتمثل في: موازنة التحميل, تقليل وقت نقل, حد الأدنى لوقت الإستجابة وضمان الإنصاف.

كلمات البحث: أملو, الغش, (ز/ز), (ز/خ), غشاش, هجوم, الوقاية, الكشف.

Table des matières

Table des illustrations

Tableaux

Introduction générale

Les jeux vidéo sur ordinateur constituent une industrie exceptionnelle qui nécessite de la créativité non triviale et de l'imagination. Parmi les nombreuses catégories de jeux, nous nous intéressons particulièrement à ceux couramment appelés les jeux en ligne massivement multi-joueurs (MMOG) en raison de leurs caractéristiques distinctes, leur complexité, et leurs impacts significatifs : commercial, social et académique. Beaucoup de challenges sont soulevés dans le domaine de recherche pour ce genre d'applications. Notons que la technologie MMOG est utilisée dans différents domaines d'applications. En effet, un nombre important d'applications non-jeu bénéficient également des technologies qui sont développées pour le MMOG telles que les simulations de vol militaire, de véhicule et de cabinet médical, l'enseignement à distance, le e-commerce et les systèmes de fabrication [Fan, 2009].

Avec l'utilisation répandue et croissante d'Internet, les MMOGs sont devenus de plus en plus populaires depuis le milieu des années 1990. Au cours de ces dernières années, le nombre total d'abonnés MMOG a augmenté d'une manière exponentielle [Woodcock, 2008].

Traditionnellement, l'architecture Client/serveur (C/S) a été le paradigme prédominant pour la mise en œuvre des MMOGs, car elle est relativement facile à mettre en œuvre et à sécuriser. Dans une telle architecture, un serveur de jeu centralisé est nécessaire pour héberger un monde virtuel persistant. Tous les joueurs envoient leurs mises à jour à ce serveur, puis ce dernier met à jour l'état du monde du jeu en conséquence et reflète le résultat à chaque client [James et Walton, 2004].

Néanmoins, l'architecture C/S reste vulnérable aux tricheurs qui profitent des failles des jeux et du système. D'autre part, l'augmentation explosive des joueurs fait que la scalabilité du serveur de jeu est devenue un défi crucial. Or, l'architecture C/S a rencontré beaucoup de problèmes qui restent non résolus, tels que la fiabilité, la surcharge du réseau (problème de bande passante) et du serveur de jeu, la tolérance aux pannes et le coût [Fan, 2009].

L'architecture Pair à Pair (P2P) se présente comme une alternative intéressante à celle C/S pour les MMOGs, car elle exploite les ressources informatiques des joueurs, et peut effectivement alléger la charge de travail imposée aux serveurs de jeu. Récemment, la

puissance de calcul disponible sur les ordinateurs personnels a augmenté de façon spectaculaire avec une bande passante réseau réduite. L'architecture P2P a non seulement le potentiel de réduire le coût d'un MMOG, mais peut aussi éviter les problèmes des architectures C/S comme la scalabilité, la fiabilité, la tolérance aux pannes et la redondance [Fan, 2009].

Dans l'architecture P2P, une partie de l'application est installée sur la machine des joueurs. Comme le joueur a un accès illimité sur sa propre machine, ce dernier est capable de manipuler et de faire des modifications sur sa machine, ce qui lui permet de contourner la protection de son PC. Par conséquent, les MMOGs basés sur les architectures P2P sont vulnérables à de nouvelles menaces de tricherie [Fan, 2009].

Il existe plusieurs approches qui traitent le problème de la détection et de la prévention de la tricherie dans les MMOGs ; certaines approches sont conçues pour les architectures C/S, d'autres pour les architectures P2P et d'autres pour les deux architectures. Dans ce contexte, l'objectif de notre travail est de proposer une approche pour détecter et prévenir la tricherie dans les MMOGs basés sur les architectures P2P.

L'étude des différentes solutions proposées dans la littérature nous a permis de proposer une classification selon un certain nombre de critères relatifs à l'architecture (C/S ou P2P), à la technique utilisée, à l'état du jeu, et au type de la tricherie. En plus de l'aspect sécurité inhérent à ce type de solutions, nous avons également pris en considération la notion de performance qui est importante pour classifier les solutions anti-tricherie. La synthèse des solutions proposées nous a amené à choisir l'une d'elles à savoir : DACA, *Dynamic Anti-Cheating Architecture for MMOGs*, proposée dans [Liu et Tang, 2009]. L'étude de cette approche nous a permis de relever certaines insuffisances relatives à son mécanisme d'anti-tricherie et qui concernent principalement son système de détection et les performances de ce dernier. Ainsi, nous proposons l'approche DACA-Extended qui apporte une amélioration à

l'approche DACA par la minimisation du temps de tricherie[1], le traitement du problème de visibilité d'adresse IP, et le problème du choix du Data-holder.

Ce mémoire est organisé comme suit :

- Le premier chapitre est divisé en deux parties : la première définit les jeux en ligne massivement multi-joueurs (MMOG), leurs domaines d'applications, leurs éléments essentiels et leurs objectifs. Cette partie introduit aussi des considérations telles que la cohérence, le temps réel, la scalabilité et la sécurité. Dans la deuxième partie, les architectures contemporaines des MMOG (C/S et P2P) sont définies ainsi que les problèmes inhérents à celles-ci.
- Le deuxième chapitre décrit les problèmes de tricherie dans les MMOGs. Une taxonomie des différents types de tricherie en ligne est présentée.
- Dans le troisième chapitre, nous présentons les différentes méthodes de détection de tricherie dans les MMOGs. Nous abordons celles basées sur les architectures client/serveur et celles reposant sur les architecture P2P.
- Le quatrième chapitre décrit notre contribution. Dans une première partie, une synthèse des différentes approches d'anti-tricherie est élaborée suivie d'une classification selon un certain nombre de critères que nous définissons. A l'issue de cette étude critique, la solution choisie est présentée et ses insuffisances sont relevées. Puis, nous abordons, dans une seconde partie, notre approche en présentant les différentes améliorations apportées par rapport aux solutions existantes.
- Et enfin une conclusion générale évoquant les perspectives de ce travail, clôture ce mémoire.

[1] Le temps de tricherie est défini comme étant le temps durant lequel le data holder peut mener une tricherie ou bien être l'objet d'une tricherie (attaque) [Norden et Guo, 2007]

CHAPITRE I : Les jeux en ligne massivement multi-joueurs

I.1. Introduction

Les jeux vidéo sur ordinateur constituent une industrie exceptionnelle qui nécessite de la créativité non triviale et de l'imagination. Dans la littérature relative au domaine du jeu, il existe une divergence sur les définitions formelles acceptées pour les différents genres de jeux, de sorte que tout classement individuel peine à être accepté comme complet ou exhaustif [Fan, 2009].

Parmi les nombreuses catégories de jeux, nous nous intéressons particulièrement à ceux couramment appelés les jeux en ligne massivement multi-joueurs (MMOG) en raison de leurs caractéristiques distinctes, leur complexité et leurs impacts significatifs : commercial, social et académique [Fan, 2009]. Beaucoup de challenges sont soulevés dans le domaine de recherche pour ce genre d'applications.

L'histoire des MMOG remonte a plus de trente ans et prend ses origines de ses antécédents, *multi-user Dungeon* (MUDs), cette histoire peut remonter encore plus loin jusqu'aux années 1970. Un MUD est un jeu d'aventure basé sur le mode texte qui se résume en une interface utilisateur simple. Les actions, comme tuer des monstres, compléter des enquêtes, se déplacer entre les chambres et parler avec d'autres joueurs, sont effectuées en tapant les commandes correspondantes.

Les MMOGs ont apporté une amélioration révolutionnaire en introduisant des représentations graphiques en 3D. Un MMOG moderne peut supporter beaucoup plus de joueurs que les MUDs. La Figure I. 1 représente deux exemples de jeux : *British Legends*[2] (un célèbre MUD qui a été créé dans les années 1970) et *World of Warcraft* (WoW)[3] (un MMOG commercial qui a été lancé en 2001) [Fan, 2009].

[2] http://british-legends.com
[3] http://worldofwarcraft.com

Figure I. 1 : MMOG : le passé (British Legends) ET le présent (World of Warcraft) [Fan, 2009].

Avec l'utilisation répandue et croissante d'Internet, les MMOGs sont devenus de plus en plus populaires depuis le milieu des années 1990. Au cours de ces dernières années, le nombre total d'abonnés MMOG a augmenté d'une manière exponentielle comme illustré sur la figure I.2. Un célèbre MMOG commercial comme *World of Warcraft* peut avoir plus de 10 millions d'abonnés payants. Cette réalité a attiré l'attention à la fois des entreprises et celle des scientifiques [Woodcock, 2008].

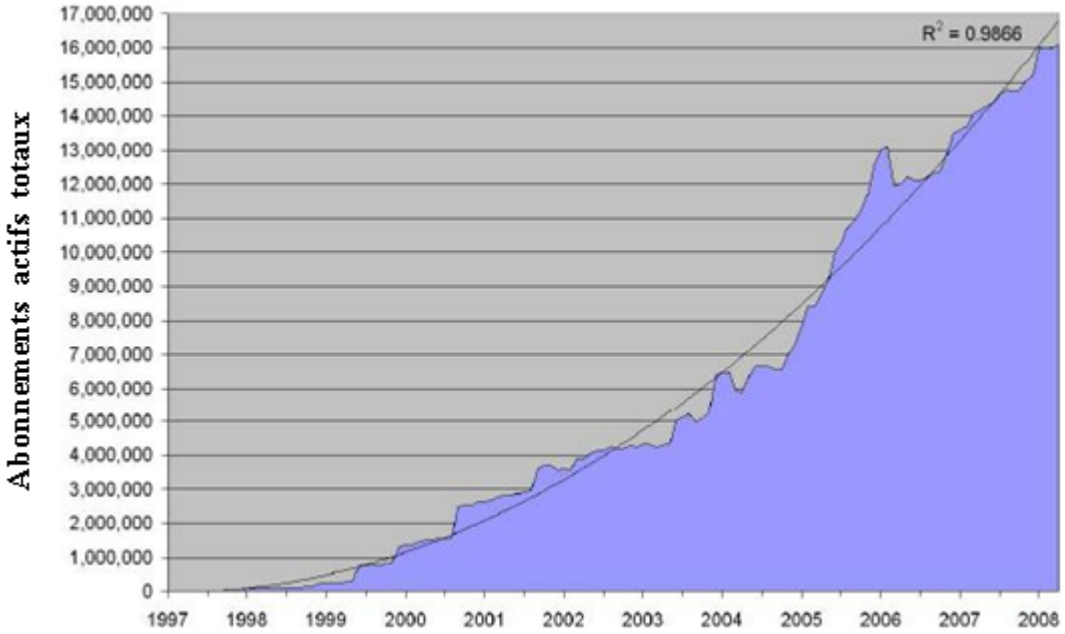


Figure I. 2 : Abonnements actifs totaux de MMOG [Woodcock, 2008].

D'un point de vue commercial, l'industrie du MMOG est actuellement l'une des entreprises en ligne les plus fructueuses et prometteuses, ayant démontré leur potentiel [Fan, 2009]. En 2009, il y'avait environ huit millions d'abonnés MMOG aux États-Unis, en Europe et à peu près le même nombre de joueurs en Asie [Woodcock, 2008]. La figure I.3 montre les rapports de croissance des revenus de jeu en ligne par an et qui est estimé à plus de quatre milliards de dollars en 2009, réalisé par la DFC Intelligence, un célèbre établissement de recherche sur l'industrie du divertissement et de jeu vidéo.

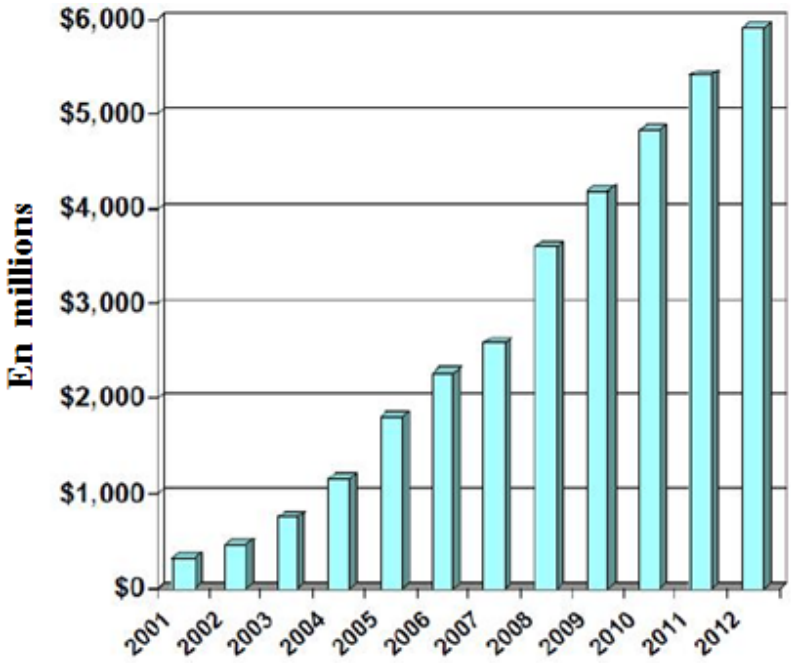


Figure I. 3 : Prévision du marché de DFC Intelligence MMOG 2008 [Fan, 2009].

I.2. Environnements virtuels et MMOG

I.2.1. Termes et définitions

Un environnement virtuel (EV) est un environnement créé artificiellement par un logiciel informatique et pouvant héberger une communauté d'utilisateurs présents sous forme d'avatars ayant la capacité de s'y déplacer et d'y interagir [OTT et Dillenbourg, 2001]. Dans ce contexte, un avatar – terme issue de la tradition hindoue où il désigne l'incarnation d'une divinité sur terre [Damer, 1997] – est défini comme étant la représentation informatique d'un internaute, que ce soit sous forme 2D, ou sous forme 3D. Une autre définition d'un *EV* l'assimile à un environnement *interactif* construit par un ordinateur en vue d'une *simulation* du monde réel. Cette réalité peut être *tridimensionnelle* (3D) et plonge l'utilisateur dans une immersion totale. Un environnement virtuel est une *modélisation* d'un environnement réel [Clarke et Dede, 2005].

Un EV peut être collaboratif (CVE, «*Collaborative VE*»), distribué (DVE, «*Distributed VE*»), immersif (IVE, «*Immersive VE*»), multi-utilisateurs (MUVE, «*Multi-User VE*»), ou en réseau (NVE, «*Networked VE*»).

Les environnements virtuels sont de plus en plus utilisés dans différents contextes, par exemple, la simulation des formations militaires [Tay, 2005], éducatifs et plates-formes de recherche [Hu et Chang, 2007] et à d'autres fins sociales [Liu et al., 2003].

La figure I.4 présente une boutique virtuelle et conférence de travail dans SecondLife [Fan, 2009].

Figure I. 4: E-commerce et conférence de travail dans SecondLife [Fan, 2009].

Dans la littérature, de nombreuses interprétations différentes ont été prévues pour les MMOG étant un genre d'applications complexes.

I.2.1.1. Définition d'un MMOG

Un MMOG est un type de jeu électronique qui permet à des milliers de joueurs d'interagir simultanément dans un univers de jeu persistant, connectés via un réseau tel qu'Internet [Fan et al., 2010].

Du point de vue académique [Fan et al., 2010]: Un MMOG est un type d'application réseau distribuée non triviale, car il propose une variété de complexités inhérentes qui méritent une étude approfondie.

Par exemple, un MMOG nécessite :

- Des interactions en temps réel dans un monde de jeu hautement réactif ;
- La fiabilité, l'extensibilité et des architectures de distribution auto-adaptative ;
- La cohérence, la sécurité et des protocoles de communication de la preuve de tricherie ;
- L'authentification, la comptabilité et la gestion des droits numériques ;
- L'intelligence artificielle (IA) pour les acteurs virtuels ;
- La création, le partage et le streaming de contenu généré par l'utilisateur.

Généralement, tout MMOG doit répondre à trois critères indispensables [Alexander, 2005] **:**

- l'univers n'est accessible que sous réseau ;
- l'univers est persistant, c'est-à-dire qu'il est tout le temps actif, quel que soit le statut des joueurs connectés ou non ;
- l'univers est accessible à un très grand nombre de joueurs simultanément.

On considère généralement qu'un jeu est massivement multi-joueurs lorsqu'il peut accueillir une centaine de joueurs simultanément [Alexander, 2005]**.**

I.2.1.2. Domaine d'applications de la technologie MMOG

La technologie propre aux MMOGs est utilisée dans plusieurs autres domaines d'application [Fan, 2009]. Ainsi, un nombre important d'applications non-jeu bénéficient également des technologies qui sont développées pour le MMOG, telles que les simulations de vol militaire, de véhicule et de cabinet médical, l'enseignement à distance, le e-commerce, les systèmes de fabrication, etc. D'autre part, la communauté *open source* est vivement intéressée par la perspective des infrastructures publiques d'un MMOG, ce qui rendrait

possible pour les joueurs adeptes (fan) de MMOG de concevoir et de mettre en œuvre leurs propres MMOG et les exécuter gratuitement.

I.2.2. Types de MMOG

Il existe plusieurs types de MMOGs [Alexander, 2005]. Dans ce qui suit, nous présentons succinctement les types les plus populaires.

I.2.2.1. MMORPG

Les jeux de rôle en ligne massivement multi-joueurs, connus sous l'acronyme anglais MMORPG («*Massive Multi-Player Online Role-Playing Games*»), sont les plus connus et les plus répandus.

L'univers d'un MMORPG est persistant et les personnages (ou avatars) des joueurs se trouvent dans un monde souvent surnaturel où le temps évolue proportionnellement à celui de la réalité. Ce monde évolue en permanence avec les joueurs qui y sont connectés et lorsqu'un joueur quitte le jeu pour y revenir plusieurs heures ou plusieurs jours plus tard, il ne le retrouve pas forcément tel qu'il était lorsqu'il en est parti car le cycle des jours et des nuits a aussi continué dans le monde du jeu. Selon le jeu, le monde a ainsi pu être modifié en profondeur ou non.

Parmi les jeux MMORPG, on peut citer : *World of Warcraft*, *Anarchy Online*, *metin 2, EverQues*, *Dark Age of Camelot*, *Ragnarok Online*, *Guild Wars*, *Lineage 2* et *Star Wars Galaxies*.

I.2.2.2. MMOFPS

MMOFPS, abréviation de *Massively Multiplayer Online First-Person Shooter*, désigne les jeux vidéo massivement multi-joueurs qui reprennent le concept du jeu de tir subjectif. Quelques jeux ont été réalisés à partir des années 2000. Ces jeux présentent généralement un combat en équipe sur un vaste terrain. Le fait que la notion de persistance du monde soit présente ajoute des éléments que l'on trouve généralement dans les jeux de rôles, comme des

points d'expérience. Parmi ces jeux, on peut citer *World War II Online*, *PlanetSide* et *Dark Orbit*.

I.2.2.3. MMORTS

Sous l'acronyme de MMORTS, *Massively Multiplayer Online Real-Time Strategy*, un certain nombre de développeurs ont essayé d'unir les jeux de stratégie en temps réel avec les MMOG. On peut ainsi nommer *Mankind*, *Shattered Galaxy* ou *Stargate-race*.

Un exemple bien connus inclut *World of Warcraft* (WoW), qui est représenté par la figure I.5 [Fan, 2009].

Figure I. 5: "World of Warcraft".

I.2.3. Éléments essentiels d'un MMOG

Il ya plusieurs éléments essentiels dans un MMOG. Les plus essentiels sont ceux relatifs au monde de jeu persistant, un nombre considérable d'avatars des joueurs, personnages non-joueurs, des artéfacts du joueur, bâtiments, installations et caractéristiques du terrain [Fan, 2009].

I.2.3.1. Monde de jeu persistant

Un MMOG dispose d'un monde de jeu virtuel persistant, qui est un monde virtuel utilisé comme environnement dans ces derniers sur ordinateur, basé sur un monde artificiel des espaces en 3D qui offrent un cadre pour des scénarios de jeu fantastique. La particularité de ce monde virtuel est qu'il est *persistant*, c'est-à-dire qu'il ne s'arrête jamais. Il existe et évolue en permanence. Contrairement à d'autres genres de jeux, des parcelles et événements des MMOGs continuent à se développer même si certains des joueurs ne jouent pas leurs personnages en ligne. Il est une sorte de monde parallèle, évoluant indépendamment du joueur, mais pouvant toutefois être modifié par celui-ci. L'influence qu'un personnage peut avoir sur le monde virtuel varie selon le jeu. Un MMOG est parfois appelé directement un monde persistant (PW, *Persistent World*).

Afin de recevoir un grand nombre de joueurs simultanés, un univers de jeu est généralement un vaste territoire. Cela rend nécessaire de diviser le monde du jeu entier en plusieurs sous-espaces, pour faciliter la gestion et la maintenance de ce dernier. La plus grande granularité de ces sous-espaces est nommée «zones de jeu», qui jouent un rôle important dans l'application d'un MMOG. Dès lors, de nombreux jeux MMOGs commerciaux ont adopté une conception basée sur la notion zone, cette solution attribue des zones de jeu spécifique à différents serveurs physiques ou processus. De cette façon, lorsque l'avatar se déplace entre les zones, la connexion des joueurs est redirigée à partir d'un serveur de jeu à un autre [Fan, 2009].

I.2.3.2. Avatars joueurs

Les joueurs dans un MMOG sont visuellement représentés par des avatars. Plus précisément, un joueur peut être désigné par deux types d'avatars. Un avatar pilote présent sur la machine propre du joueur, et un avatar drone qui duplique l'avatar pilote sur toutes les autres machines connectées au monde. Dans la couche de communication, un avatar pilote est contrôlé directement par l'utilisateur, alors qu'un avatar drone nécessite son état et ses mises à jour de comportement qui doivent être acheminés via le réseau, ce qui introduit un problème de latence.

Dans un MMOG, la latence se montre comme les temps de réponse des joueurs en interaction. Un certain nombre des facteurs contribuent à la latence.

Les concepteurs des MMOGs intègrent souvent une tolérance de latence dans le modèle de communication en utilisant diverses méthodes pour estimer les informations sur les autres joueurs dans le jeu, par exemple, la position actuelle d'un avatar du joueur peut être prévu en fonction de sa position, sa vitesse, sa direction et le temps écoulé déterminés précédemment [Fan, 2009].

I.2.3.3. Personnages non-joueur

En plus des personnages joueurs (PC) qui sont représentés par les avatars joueurs contrôlés, il y a aussi les personnages non-joueurs IA-contrôlée (NPC), qui sont des acteurs virtuels importants et qui suivent continuellement le scénario du MMOG. La figure I.6 illustre deux catégories typiques de NPCs. Celles de gauche sont des créatures légendaires dans *Word of Warcraft*. Ces créatures sont capables de suivre les PCs sur leurs déplacements, et de fournir des moyens de transport dans les différentes zones du monde du jeu. Celle de droite représente une capture d'écran montrant un groupe de PCs luttant contre une *sentinelle Anubisath* dans *Word of Warcraft* qui garde le *Temple d'Ahn'Qiraj*.

Il est nécessaire de distinguer les NPCs, parce qu'ils ont des exigences différentes pour la communication réseau. Afin de donner un sens partagé de l'espace entre les joueurs, chaque joueur doit conserver une copie locale de l'état de jeu. Parfois, quand un joueur effectue une action, l'état du jeu de tous les autres joueurs concernés par cette action doit être immédiatement mis à jour par les événements en temps réel des jeux. Si la transmission de tels événements est sérieusement retardée par le réseau (c'est le problème de latence), elle se traduira par des états de jeu incompatibles sur les machines des joueurs. Lutter contre la *Sentinelles Anubisath* est un exemple des interactions en temps réel, dans laquelle les multiples états du jeu des différents joueurs doivent être soigneusement synchronisés. Cependant, l'embauche d'une créature légendaire « *Hippogriffe* » est un exemple d'une interaction sans latence, car cette dernière action n'a pas d'effet sur l'état du jeu des autres joueurs, et les autres joueurs peuvent se rendre compte de cet événement à n'importe quel autre moment.

Généralement, une interaction entre deux PC est appelé un Joueur-vs-Joueur (PvP interaction), et l'interaction qui implique à la fois les PC et les NPC est appelé un Joueur-vs-Non-Joueur (*PVN interaction*) [Fan, 2009].

Figure I. 6 : NPCs typiques dans MMOGs [Fan, 2009].

I.2.3.4. Autres objets, entités et objets

Outre le PC et les NPC, il existe aussi un nombre considérable d'autres objets dans l'univers de jeu. Certains objets sont invariables comme les bâtiments, mais beaucoup d'objets sont des entités et des articles changeables, comme les armes, les équipements, les potions magiques et pierres précieuses. Beaucoup d'objets modifiables sont émis par des monstres vaincus du NPC, comme le butin pour les vainqueurs. Un joueur est en mesure de prendre ces éléments, les placer dans son propre inventaire, les consommer, ou bien les échanger contre des devises virtuelles.

Une répartition inégale des objets de jeu conduit souvent à une question de *flocking* dans un MMOG, qui désigne le passage de beaucoup de joueurs à un seul secteur (i.e. point chaud «hotspot») du monde du jeu. Le *flocking* se produit car il est inévitable pour un monde de jeu d'avoir quelques régions qui sont plus intéressantes, ou tout simplement plus rentables pour les joueurs en termes de trésors. L'une des conséquences des *flocking* est le problème de l'équilibre de charge, puisque ces régions attirent beaucoup plus de personnes que les autres régions qui restent faiblement occupés [Fan, 2009].

I.2.4. Thèmes et objectifs dans un MMOG

La majorité des MMOGs populaires est basée sur des thèmes de fantaisie traditionnelle impliquant des quêtes, des monstres et des butins. Toutefois, contrairement à d'autres genres de jeux où un joueur doit effectuer des tâches dans une séquence pour atteindre un but dans le jeu, dans un MMOG généralement il n'y a pas de but. En d'autres termes, les joueurs sont consignés dans un monde de jeu libre de faire ce qu'ils veulent. Il n'y a pas d'objectif global de "battre le jeu" à la fin. La plupart des joueurs s'installent dans un schéma d'élaboration de leurs personnages et presque tous les efforts MMOGs visent à fournir des systèmes de progression de personnage distinctif. Traditionnellement, les joueurs gagneront des points d'expérience pour rendre leurs personnages plus puissants en accomplissant des quêtes à la demande des NPCs. Il s'agit souvent de faire des combats avec les monstres ce qui offre des possibilités d'acquérir des richesses en termes de monnaie virtuelle, des armes rares et autres objets de valeur.

I.2.5. Considérations générales dans un MMOG

Un MMOG comporte des niveaux importants d'interactivité et de complexité et de ce fait, la réussite de toute conception et mise en œuvre d'un tel système doit considérer l'ensemble des points suivants [Fan, 2009] :

I.2.5.1. Cohérence

Dans un MMOG, tous les utilisateurs devraient voir la même séquence d'événements dans le même ordre, indépendamment de l'architecture ou la mise en œuvre. Par exemple, le joueur A peut tirer sur le joueur B si B est le tireur du joueur C. Cependant, le problème est que les événements peuvent avoir lieu sur des machines séparées géographiquement et des détails de l'événement doivent être envoyés entre machines d'un joueur par le passage de message. Cela implique qu'un mécanisme comme estampillage des événements doit exister sur des machines différentes avec des horloges qui sont globalement synchronisées. Un MMOG doit fournir une fonctionnalité de livraison d'événement cohérente, afin d'assurer que A, B, C et tous les autres utilisateurs dans le jeu voient la même séquence du même événement.

I.2.5.2. Temps réel

Dans un MMOG, lorsqu'un événement se produit à l'instant t pour un utilisateur, les autres utilisateurs devraient voir cet événement au temps t aussi. En temps réel, les interactions sont particulièrement vulnérables aux effets de latence. Par conséquent, la qualité de service de la performance du réseau est importante pour l'expérience de jeu d'un joueur, qui à son tour détermine la réussite de l'entreprise d'un projet de MMOG.

I.2.5.3. Scalabilité

Comme son nom l'indique, un MMOG permet à un nombre massif de joueurs d'interagir les uns avec les autres simultanément, de sorte qu'une architecture MMOG valide devrait être facilement extensible. En outre, la population actuelle en ligne peut varier considérablement au fil du temps. Bien que l'architecture soit en mesure de supporter le nombre maximum de joueurs pendant les heures de pointe, elle peut gaspiller une quantité considérable de puissance de calcul en dehors des heures de pointe. Soutenir un MMOG avec le moins de ressources possibles, et rendre l'application évolutive sont des défis majeurs.

I.2.5.4. Sécurité

Les questions de sécurité se présentent à tous les stades de la conception et la mise en œuvre d'un MMOG. Une attention particulière doit être accordée à la question de sécurité car, en dehors de l'univers du jeu, n'importe quel utilisateur peut être un tricheur potentiel ou un craqueur. Ce dernier peut avoir pour objectif de craquer une application de jeu ou généralement ruiner l'économie d'une société de jeu. Or, à l'intérieur du monde du jeu, nous ne pouvons pas imposer une norme qui gouverne le comportement des joueurs.

I.3. Les architectures MMOGs

Dans cette partie, nous citons les deux architectures Client/serveur et Pair à Pair sur lesquelles sont bâtis les MMOGs.

I.3.1. Architecture Client/serveur

I.3.1.1. Définition

L'architecture C/S a été le paradigme prédominant pour la mise en œuvre traditionnelle des MMOGs. Dans une architecture C/S, un serveur de jeu centralisé est nécessaire pour héberger un monde virtuel persistant. Tous les joueurs envoient leurs mises à jour à ce serveur, puis ce dernier met à jour l'état du monde du jeu en conséquence et reflète le résultat à chaque client. En d'autres termes, dans un jeu de N joueurs, un maximum de 2N messages sont échangés entre le serveur et les joueurs à chaque partie. Si n'importe quel joueur souffre d'une mauvaise connexion, ce sera seulement un impact sur le jeu propre du joueur, et dans une mesure limitée sur les jeux des joueurs qui tentent d'interagir avec ce joueur. La figure I.7 montre une vue très simplifiée d'une architecture C/S pour MMOG [James et Walton, 2004].

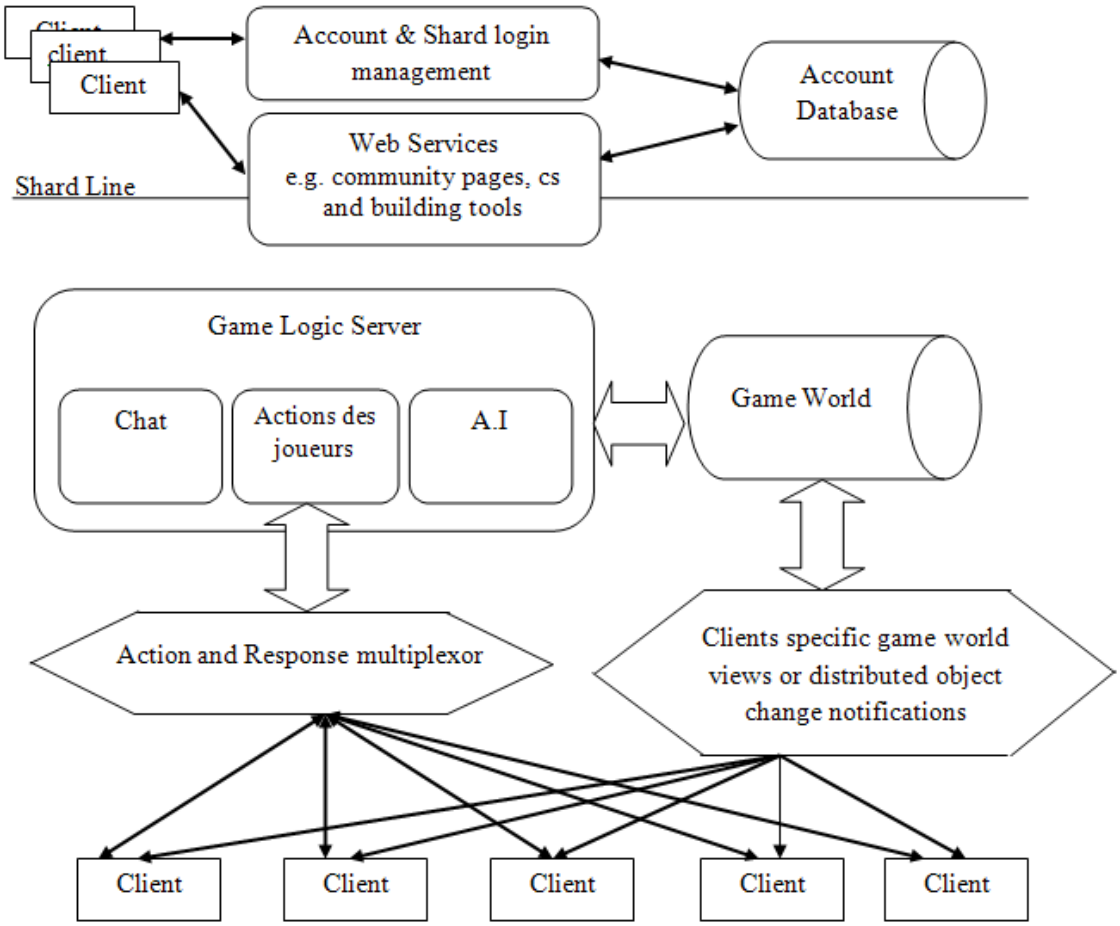


Figure I. 7 : Une vue simplifiée d'une architecture C/S pour MMOG [James et Walton, 2004].

Les fonctionnalités du serveur de jeu comprennent mais ne sont pas limités à [James et Walton, 2004] :

- Effectuer des tâches administratives et de contrôle, telles que la comptabilité du temps passé en jeu sur une base individuelle.
- Recevoir des commandes d'entrée de chaque client, transformant les touches en actions comme les sauts et les armes à feux et d'envoyer des mises à jour vers les clients concernés.
- Garder une trace de ces entités comme les avatars, les monstres, les balles, etc., et calculer la position des objets en mouvement.
- Réduire le trafic des messages seulement aux utilisateurs individuels concernés par ces paquets.
- Compresser plusieurs paquets dans un message unique pour éliminer le flux de messages redondants et réduire les frais de communication.

Dans cette architecture, le serveur de jeu prend en charge le calcul et les charges de travail du réseau. Cette situation n'était pas trop mal dans les années 90, quand le mot "massivement" n'impliquait qu'environ 100 joueurs simultanément. Actuellement, elle s'est dégradée pour devenir une véritable crise, car le nombre de joueurs a explosé à des centaines de milliers [Woodcock, 2008]. La Figure I.8 montre le maximum atteint du nombre d'utilisateurs simultanés de plusieurs jeux MMOG asiatiques de 2001 à 2006 [Woodcock, 2008].

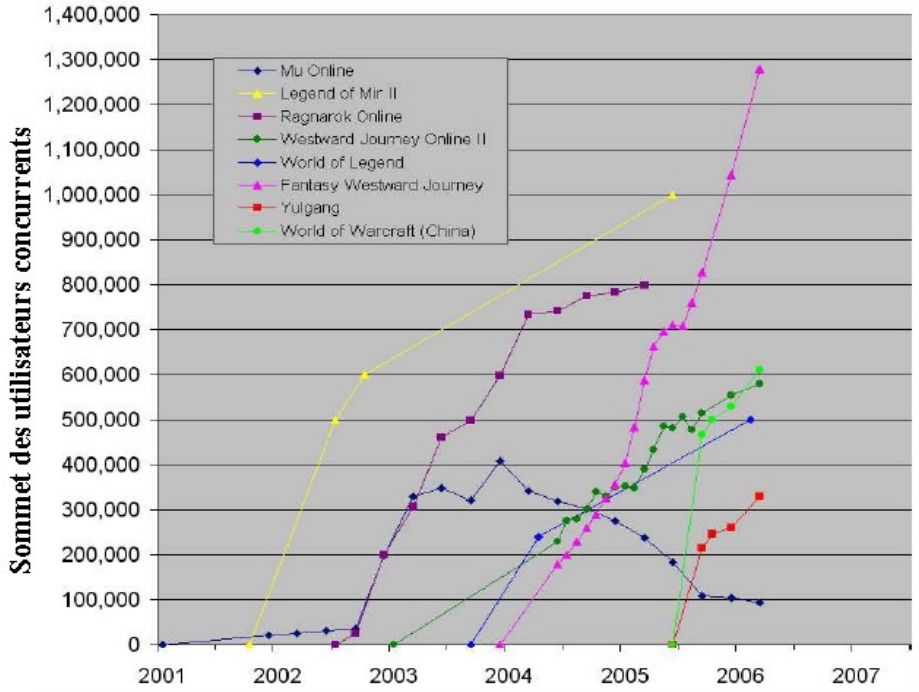


Figure I. 8 : Sommet des utilisateurs concurrents du marché asiatique [**Woodcock**, 2008]**.**

Bien que les développeurs de jeux continuent d'essayer d'optimiser leurs conceptions et de minimiser le trafic des messages, un seul serveur de jeu par lui-même est encore trop faible en matière de puissance pour pouvoir supporter la taille moyenne d'un MMOG. La solution à ce problème est de partager la charge du serveur sur un groupe de serveurs de jeux « *Server clusters* ».

Un Groupe de serveurs fournit une manière explicite pour un MMOG de partitionner un monde de jeu incroyablement vaste en plus petits morceaux qui peuvent être facilement mis en correspondance avec des serveurs distincts. La Figure I.9 illustre cette approche, dans laquelle des serveurs de jeu multiples sont introduits pour le partage de la charge de travail totale. Cependant, des mécanismes de synchronisation particulière doivent être appliqués à la connexion de ces serveurs de façon transparente, de manière à maintenir la cohérence de l'état de jeu. Un groupe de serveur de jeu est aussi appelé une ferme de serveur de jeu.

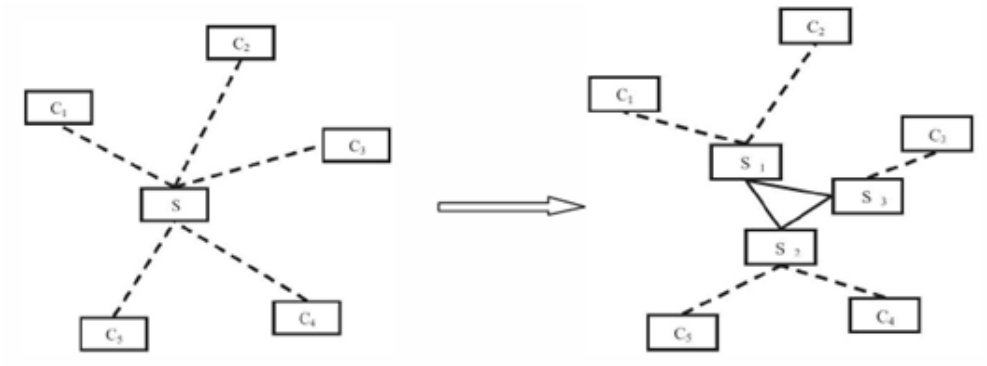


Figure I. 9 : Passage d'un serveur unique (à gauche) au groupe de serveur distribué (à droite) [Fan, 2009].

I.3.1.2. Avantages et inconvénients des architectures C/S

I.3.1.2.1. Avantages

- **Autorité :** Dans l'architecture C/S, l'état global du jeu est géré et stocké d'une manière centralisée, ce qui confère un avantage de donner aux fournisseurs de jeux un contrôle total sur leurs jeux [Merabti et Rhalibi, 2004].
- **Sécurité :** comme le serveur de jeu est le seul qui a le pouvoir exclusif à valider chaque demande d'action envoyée par les clients avant de les exécuter, il a de bonnes perspectives pour empêcher la tricherie [Kabus et al., 2005].
- **Simplicité :** l'architecture C/S est plus simple à mettre en œuvre par rapport à d'autres architectures [Mulligan et Patrovsky, 2003]. Il est largement admis qu'un MMOG est difficile pour la conception, la construction, le test, et le soutien.

I.3.1.2.2. Inconvénients

- **Fiabilité :** L'architecture C/S n'est pas intrinsèquement tolérante aux pannes. Quand le serveur d'une région s'arrête de fonctionner, tous les joueurs dans cette région sont bloqués et ne peuvent pas jouer jusqu'au rétablissement du serveur.

- **Scalabilité** : la scalabilité est un problème à multiples facettes. Il est difficile de prédire combien de joueurs peuvent être supportés par la plate-forme matérielle, et combien de bande passante réseau est suffisante pour assurer une expérience de jeu agréable [Fan, 2009].

En outre, d'une part, la plupart des projets de MMOG emploient le modèle d'affaires *pay-for-play* basé sur un abonnement mensuel, et d'autre part la plupart des joueurs paient aussi pour leurs connexions à Internet mensuellement, il devient flexible pour les joueurs de choisir leur temps favori pour jouer en ligne. Dans ce cas, bien que les prestataires de services MMOG puissent calculer le nombre total de joueurs qui ont souscrit à un MMOG, il est impossible de prévoir avec exactitude le nombre de joueurs simultanés à tout moment.

- **Redondance** : Les architectures C/S luttent pour faire face aux *flocking*, car il est difficile d'atteindre dynamiquement l'équilibrage de charge entre tous les serveurs de jeux. En conséquence, dans une zone de MMOG, tous les serveurs de région sont conçus pour être également capables d'accueillir un nombre maximum de joueurs dans les heures de pointe, mais dans la plupart des cas, ces ressources de calcul sont redondantes [Kesselman, 2005].
- **Le Coût** : Il est largement admis que les MMOGs sont coûteux, le lancement d'un MMOG aujourd'hui prend de 2 à 3 ans et environ 10 millions de dollars, et les coûts de soutien consommeront jusqu'à 80% de ses revenus [Kesselman, 2005]. Les MMOGs nécessitent à la fois la puissance de traitement intensif et la bande passante de réseau utilisé.

En réponse à ces problèmes, plusieurs nouvelles architectures ont été proposées comme des substitutions pour l'architecture classique C/S, parmi lesquels l'architecture P2P pour MMOG.

I.3.2. Architecture Pair à Pair

I.3.2.1. Définition

Pair-à-Pair ou P2P n'est pas un concept nouveau. Il existe depuis l'arrivée d'Internet (années 1970) [Webb, 2002]. Les serveurs de messagerie, les serveurs *USENET* et les serveurs de noms de domaine sont des services largement utilisés et qui fonctionnent d'une maniéré similaire au P2P. Aujourd'hui, de nombreux facteurs tels que l'explosion du nombre d'appareils connectés à Internet, l'augmentation rapide de la bande passante, la croissance

continue de la puissance de calcul et des capacités de stockage, assurent la pratique du P2P pour un grand nombre d'applications.

Il est utile de caractériser d'abord les systèmes P2P. En bref, P2P ne stipule pas d'éliminer la notion de client/serveur. Ce n'est pas une technologie, une application ou un modèle de travail unique. Il ne devrait pas être strictement caractérisé par son degré de décentralisation [Brookshier et al., 2002].

Les caractéristiques communes des systèmes de P2P d'aujourd'hui comprennent la plupart des éléments suivants :

- Les nœuds de Pair à Pair ont conscience des autres nœuds i.e. chaque nœud peut communiquer avec les autres nœuds du réseau via des nœuds intermédiaires.
- Chaque nœud agit à la fois comme un consommateur des services et un fournisseur des services.
- Les nœuds forment les communautés de travail de données et d'application qui peuvent être décrits comme des groupes de Pair.

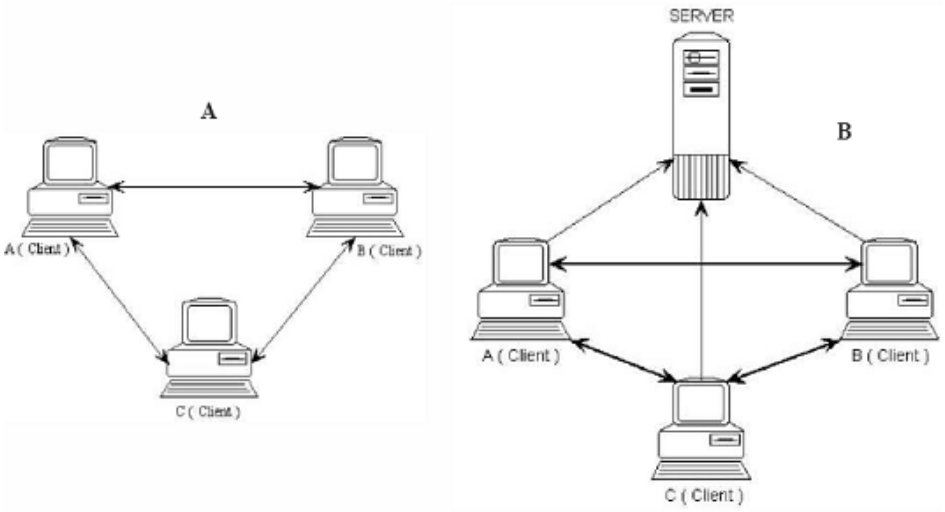


Figure I. 10 : Les modèles Pair à Pair [Webb, 2002]

Les Modèles de P2P peuvent être subdivisés dans les catégories suivantes [Webb, 2002]:

P2P pur : Comme le montre la figure I.10 partie A, toute la communication se produit entre les Pairs (nœuds) connectés sans assistance d'un serveur.

P2P avec un serveur de découverte (P2P hybride) : Comme le montre la figure I.10 partie B, ce modèle contient un serveur qui est limité à fournir les noms des Pairs (nœuds) qui sont déjà connectés aux Pairs entrants. Le serveur aide seulement les Pairs en fournissant une liste des Pairs qui sont connectés, mais l'établissement de la connexion et de la communication reste le travail des Pairs. Un tel modèle surpasse le modèle P2P pur en augmentant les chances de trouver un plus grand nombre de Pairs sur le réseau. Si le serveur venait à disparaître, ou venait à tomber entre les mains de partis hostiles, le réseau deviendrait non fonctionnel, et pour y remédier on devra prévoir une solution qui permet d'élire un nouveau serveur de découverte parmi les nœuds connectés.

P2P avec Serveur de découverte, et recherche de contenu : Dans ce modèle, un serveur est utilisé pour fournir la liste des Pairs qui sont connectés ainsi que les ressources disponibles dans chacun. Par conséquent, ce modèle permet de réduire la charge sur ses Pairs, car un Pair n'a plus besoin de visiter chaque autre Pair individuellement pour retrouver les informations convoitées. Certains serveurs peuvent même cacher les contenus qui sont échangés entre les Pairs. Ceci améliore encore l'efficacité et la disponibilité de l'ensemble du système. Le problème de confiance «trusts» de tels serveurs pousse à sécuriser le serveur central d'une manière optimale.

I.3.2.2. Les applications Pair-à-Pair

Des scénarios d'utilisation de P2P peuvent être classés en trois grandes catégories **[**Kant et al., 2002] :

1) **Le partage des contenus :** deux sous classes peuvent être distingué.

- **Partage de fichiers** : telles que Gnutella [Karbhari et al., 2004], eDonkey [Tutschku, 2004], Napster [Bengt et Rune, 2001] et BitTorrent [Izal et al., 2004], sont largement utilisés pour partager des fichiers médias.
- **Distribution et publication de contenu des réseaux :** comme OpenCola [Conn et al., 2003] et Blue Falcon Networks [Networks, 2008] ont été développé pour la distribution de contenu P2P. Ils sont différents du partage de fichiers P2P parce que les fichiers sont répliqués entre les Pairs, alors que le contenu est distribué à partir d'une seule entité, comme un journal, pour le Pairs. Dans ce cas, les éditeurs de

contenu n'ont plus besoin de grands volumes et de serveurs web coûteux : seule la première copie du contenu doit être récupérée par un Pair en premier lieu, puis les Pairs vont répliquer le contenu entre eux.

2) **Le partage de ressources matérielles** : les ressources partagées sont principalement le processeur (capacité de calcul). Le calcul de réseau de P2P est représentée par une série de projets *@HOME* bien connus [Anderson et al., 2002]. Il effectue des simulations intensives de calcul en utilisant des machines domestiques inutilisées sur Internet. Une caractéristique commune importante dans ces applications est de partager les charges de travail infini dans des unités qui sont suffisamment petites pour satisfaire les demandes de chaque participant dans un temps raisonnable. Par la suite chaque unité est traitée en même temps et avec très peu d'interactions, ces applications sont aussi appelées « calcul parallèle embarrassant ».
3) **L'informatique et les communications collaboratives :** Voici quelques exemples parmi les applications les plus connus de cette classe :

- **Messageries instantanées :** Les applications de messagerie instantanée sont souvent conçues pour fonctionner comme des applications P2P afin d'incarner la fonction qui permet l'interaction directe entre un nœud et un autre. Des exemples populaires de messageries instantanées comprennent MSN Messenger, ICQ et Jabber. Elles atténuent l'interaction P2P en s'appuyant sur un service de présence à partir d'un serveur centralisé.
- **Collaboration et tableau blanc « white boarding »** : La collaboration comprend une catégorie vaste d'applications qui sont P2P par nature. Dans un groupe de travail, la collaboration implique le partage d'idées et d'autres ressources. Par exemple, les membres d'une équipe peuvent échanger des messages ou des documents entre eux via courrier électronique, vidéoconférence, ou via des applications de Chat. Au cours des séances, il peut être utile d'avoir un tableau blanc virtuel partagé pour faciliter les communications entre les collaborateurs éloignés. P2P permet à tous de ces opérations de se produire indépendamment du serveur central.

I.3.2.3. La structure des réseaux de recouvrement de P2P

En dehors de l'organisation centralisée ou de contrôle, il est difficile pour un utilisateur de trouver une ressource spécifique dans un réseau de P2P à grande échelle. Les premières applications P2P recherchaient les Pairs et les ressources du réseau par inondation de requête sur tous les Pairs déjà connectés. Chaque Pair impliqué retransmet la requête à tous les Pairs voisins connus, qui à son tour décide s'il y a lieu de répondre à la requête, ou encore de la transmettre à ses propres voisins. Malheureusement, de tels mécanismes ne s'adaptent pas bien, et il n'est pas garanti qu'une ressource utile puisse être définitivement établie.

Deux problèmes existent avec les approches fondées sur les inondations [Doval et O'Mahony, 2003] :

- L'emplacement des ressources et la topologie du réseau ne sont pas corrélées : les ressources disponibles pourraient ne pas être accessibles à tous les nœuds du réseau, et le succès de requête ne peut pas être garantie, même si le nœud cible est connecté au réseau.
- Un réseau P2P non structuré : les requêtes sur un réseau P2P non structuré ont une tendance à rechercher une complexité de l'ordre de N saut, avec N étant le nombre de nœuds du réseau.

Afin de faire face à ces problèmes, un certain nombre de superposition des infrastructures de P2P structurés ont été proposés au début des années 2000, par exemple CAN [Ratnasamy et al., 2001], Pastry [Rowstron et Druschel, 2001], Chord [Stoica et al., 2001] et Tapestry [Zhao et al., 2001]. Ces infrastructures organisent des Pairs dans une topologie logique, selon le contenu, les ressources ou les services qu'un Pair peut fournir. Un « réseau de recouvrement » est une couche de la topologie du réseau virtuel au dessus d'une topologie de réseau physique, comme le montre la figure I.11. Les principales fonctions offertes par un réseau de recouvrement est le routage efficace des messages.

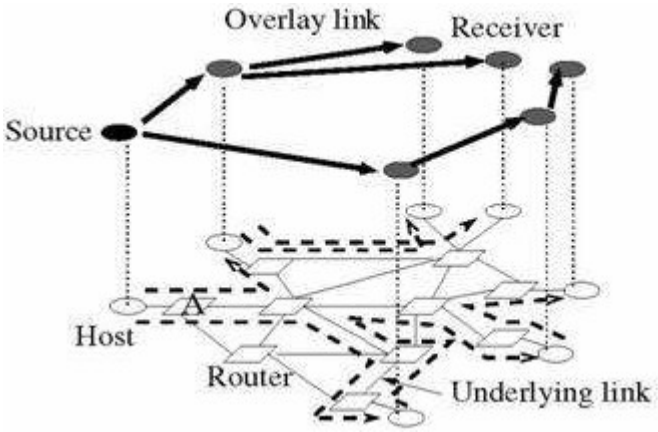


Figure I. 11 : Un réseau de recouvrement P2P sur un réseau physique [Doval et O'Mahony, 2003]

Nous citons comme exemple de réseau de recouvrement : le réseau nommé *Pastry*. Les Pairs sont organisés en anneau logique représenté par la figure I.12. Chaque nœud dans l'anneau a un identifiant aléatoire unique et uniforme (c.-à-d. *nodeId*), codifié sur 128 bits circulaire. L'acheminement d'un message d'un nœud à un autre se fait suivant la valeur numérique de la clé. Le prochain destinataire de message sera le nœud actif avec un numéro plus proche de la clé du nœud expéditeur. Par exemple, dans la figure I.12, lorsque le nœud 65a1fc envoie un message avec d46a1c clé, le message est routé (soit transmis) par nœud d13da3, d4213f, d462ba, et il est enfin reçu par nœud d467c4, car actuellement ce *nodeId* est numériquement le plus proche à la clé.

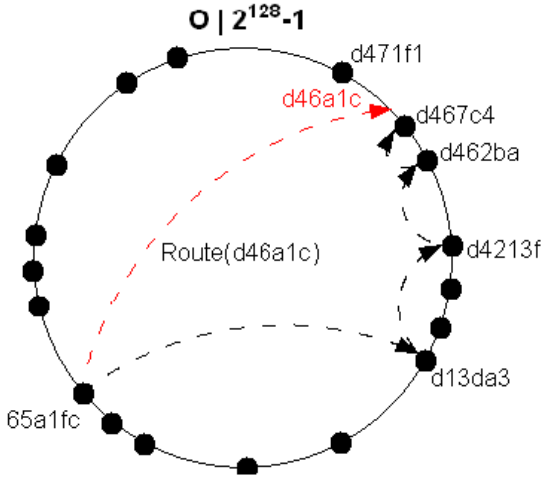


Figure I. 12 : Le réseau de recouvrement de Pastry [Rowstron et Druschel, 2001]

Les réseaux de recouvrement fournissent les fonctionnalités suivantes :

- **Auto-organisation :** Un Pair ne requiert que la configuration minimale et la connaissance des autres Pairs afin de rejoindre un réseau de recouvrement.

- **Equilibrage de charge Automatique :** Parce que les *nodeIds* et clés de message sont assignés tout les deux au hasard et sont uniformément distribuée, et sans coordination globale, cela aboutit à un bon équilibre du premier ordre de la charge de requêtes entre tous les nœuds, ainsi qu'à la charge réseau sur l'Internet [Fan, 2009].
- **La tolérance aux pannes :** Un réseau de recouvrement est assez résilient aux arrivées, aux départs et aux échecs de nœud. i.e. Quand une panne survient dans un nœud ou un message est altéré, ce type de réseau surpasse le problème avec les solutions de réparation après incident.
- **Efficacité et diffusion de l'information scalable :** les propriétés de localité du réseau structuré garanti que les informations peuvent être diffusées efficacement même dans des systèmes à grande échelle de P2P.
- **Cartographie des objets d'application des nœuds :** En plus des messages, tous les objets spécifiques à l'application peuvent être aussi bien identifiés par une clé (*objectId* «OID»). De cette manière, un objet peut être mappé à un «propriétaire» de nœud, dont *nodeId* est numériquement le plus proche de l'OID. L'insertion d'un objet est gérée comme l'acheminement d'un message, où l'OID est utilisé comme destination de l'objet. Accéder à un objet consiste à récupérer l'objet de son nœud propriétaire, où l'OID sert de la clé de recherche [Fan, 2009].
- **Amélioration de la disponibilité et la persistance :** Au lieu d'attribuer un objet à un propriétaire unique, des répliques d'un même objet peuvent être stockées au niveau de plusieurs utilisateurs. Dans ce cas, même si certains de ces propriétaires sont temporairement déconnectés, l'objet est toujours disponible dans divers emplacements.

I.3.2.4. Avantages et défis des MMOGs en P2P

I.3.2.4.1. Avantages

Par rapport à l'architecture classique C/S, l'attraction majeure de l'architecture P2P, est qu'elle offre une façon évolutive et à faible coût pour construire des MMOGs. Les systèmes P2P sont de plus en plus populaires, car ils sont fondamentalement bons pour partager les ressources informatiques hétérogènes d'une manière distribuée tout en maintenant les coûts bas. Vu que la puissance de calcul des ordinateurs personnels augmente considérablement avec la bande passante réseau à bas prix, il est devenu possible de faire migrer une grande

partie des fonctionnalités d'un serveur de jeu aux machines des clients riches en ressources. Les actifs informatiques participants (les cycles CPU, la mémoire, la capacité de stockage et la bande passante réseau) viennent soutenir le MMOG. D'un point de vue technique, l'architecture P2P a plusieurs avantages sur une architecture C/S :

- **Fiabilité :** Dans les jeux MMOG, les fonctionnalités du jeu ne sont pas centralisées dans un seul serveur, comme dans les jeux classique. Elles sont distribuées sur plusieurs nœuds du réseau (alias super-Pairs). Ainsi, la charge de travail du serveur est distribuée sur l'ensemble des nœuds.
- **Scalabilité :** Une architecture P2P est plus évolutive qu'une architecture C/S, car plus de joueurs apporteront également plus de ressources informatiques à l'application P2P. Actuellement, un ordinateur de jeu est souvent équipé d'un processeur assez puissant, d'une mémoire en giga-octets, et d'une bande passante réseau adéquate. Par conséquent, il est possible pour une machine participant d'effectuer d'autres tâches administratives ou informatiques pour le bien-être collectif en plus de sa propre gestion.
- **Redondance :** Un MMOG P2P repose sur les ressources disponibles sur les machines qui participent aux jeux. Un des systèmes de prévention contre la tricherie est la redondance des taches, possible grâce aux ressources de calcul des clients. Les serveurs assignent les mêmes tâches à plusieurs clients et attendent les résultats. La comparaison entre ces derniers permet de détecter le client tricheur [Matthew et al., 2005].
- **Coût :** Du point de vue commercial, un MMOG P2P offre la possibilité de déployer un MMOG sans les frais d'abonnement (le coût matériel est réduit du moment que le jeu est supportés par plusieurs machines déjà existantes sur Internet, sans dépenses supplémentaires pour des serveurs dédies), sans l'investissement incroyable nécessaire par les architectures précédentes. Il offre également des opportunités d'affaires pour l'industrie du jeu, parce qu'il est plus flexible et rentable pour les entreprises de commercialiser leurs logiciels de jeu, sans fournir le support matériel dédié. En d'autres termes, le risque d'affaires pour lancer un projet de MMOG peut être considérablement réduit.

I.3.2.4.2. Défis

Un MMOG P2P est plus complexe que toutes les applications P2P existantes, et la possibilité d'adapter MMOG à l'architecture P2P est encore dans le doute en raison des défis suivants :

- **Jeu de l'interactivité :** Une des caractéristiques les plus importantes d'un MMOG est l'interaction en temps réel. Un MMOG est vulnérable aux défis de latence.
- **Nombre minimal d'utilisateurs :** Pour de nombreuses applications P2P, le nombre minimum d'utilisateurs est l'un de leurs facteurs succès, car plus le nombre des utilisateurs qui participent est grand, plus l'application sera efficace. Quand il s'agit d'un MMOG P2P, il est important pour les concepteurs de jeu d'assurer la disponibilité d'un jeu. D'une part, n'importe quel échec de nœud ne doit pas influencer sur le processus de jeu. D'autre part, même si la population en ligne est petite, le monde du jeu devrait toujours être en mesure d'évoluer normalement et efficacement.
- **Bande passante du réseau asymétrique :** Un nœud dans un réseau P2P représente le client et le serveur à la fois, alors il faudra probablement utiliser autant de bande passante en entrée et sortie du nœud. En fait, la plupart des utilisateurs d'Internet souffre de bande passante asymétrique, ce qui exige à son tour que la surcharge de la communication imposée sur chaque machine joueur doit être réglée en dessous d'un seuil spécifique indépendamment du nombre de joueurs simultanés. Le réseau Internet ne permet pas des charges sur le réseau au-delà d'un certain seuil. Ce fait est dû à la nature asymétrique du download et l'upload. Cela affecte sûrement une application de jeu P2P, notamment quand le nombre de joueurs devient plus important.

- **Obstacles généraux :** Les réseaux P2P sont confrontés à certains obstacles inhérents qui doivent être pris en considération. Par exemple, les pare-feux et les serveurs proxy empêchent les communications directes entre les Pairs. Puisqu'un nœud n'est pas plus important que les autres dans les réseaux purs de P2P, beaucoup d'entre eux doivent être connectés les uns aux autres.
- **Sécurité :** Comme les responsabilités d'un serveur de jeu sont effectuées par les machines des joueurs, le contrôle du pirate sur le jeu devient difficile, du moment que les différentes parties du jeu sont distribuées sur plusieurs machines. Alors que dans

un jeu centralisé, le pirate doit maîtriser le serveur pour s'emparer du jeu, dans un jeu P2P, il lui suffit de modifier quelques paramètres du jeu en sa faveur.

I.4. Conclusion

Un MMOG P2P doit surmonter l'ensemble des problèmes suivants : interactivité, nombre minimal d'utilisateurs, bande passante du réseau asymétrique et sécurité. Dans ce mémoire, nous nous intéressons particulièrement à la sécurité dans les MMOG P2P, et plus précisément comment détecter et arrêter les tricheurs et quelle sont les solutions qui devraient être proposés.

CHAPITRE II : Les problèmes de tricherie

II.1. Introduction

L'industrie des jeux en ligne a beaucoup évolué cette dernière décennie. Tricher dans un jeu en ligne est un aspect de recherche dans l'étude des jeux d'ordinateur qui a reçu jusqu'à présent l'attention de beaucoup de chercheurs.

La tricherie est considérée comme une menace de sécurité très importante dans les jeux en lignes. Elle se traduit en une activité qui modifie l'expérience du jeu pour donner à un joueur un avantage sur d'autre joueur(s). Dès la sortie des premiers jeux multi-joueurs populaires sur Internet, la tricherie a pris de nouvelles dimensions. Auparavant, il était assez facile de voir si les autres joueurs trichent dans la plupart des jeux qui ont été joués sur des réseaux locaux ou consoles. L'Internet a changé tout cela, par l'augmentation de la popularité des jeux multi-joueurs, en donnant l'anonymat aux joueurs, et en offrant aux joueurs le moyen d'exécuter leurs tricheries. Le plus grand problème dans le jeu en ligne est de bien caractériser les opérations autorisées et celles qui ne le sont pas.

II.2. Les types de tricherie dans les jeux en lignes

Les problèmes de tricherie ont été en grande partie étudiés et traités cas par cas jusqu' à ce que Yan et Randell [Yan and Randell, 2005] ont présenté une liste étendue des différentes catégories de la tricherie dans les jeux en ligne. Elle est classée en 15 catégories.

II.2.1. Exploitation de la confiance mal placée

Les tricheurs peuvent modifier leur programme de client de jeu ou les données, puis ils remplacent l'ancienne copie avec la copie modifiée pour une utilisation future. Alternativement, ils peuvent modifier ou remplacer le code et les données à la volée.

Les tricheurs peuvent également falsifier leurs programmes client à la volée pour accéder aux états sensibles de jeu qui sont autrement indisponibles aux joueurs.

Une telle tricherie se produit principalement en raison de la confiance mal placée, les réalisateurs faisaient trop de confiance dans le côté du client. En réalité, on ne peut pas faire confiance aux joueurs car ils ont un contrôle total sur leurs programmes de jeu.

II.2.2. Collusion

Les joueurs peuvent s'entendre les uns avec les autres pour gagner des avantages injustes par rapport à leurs adversaires honnêtes. Par exemple, la prétendue "*win trading*" était une tricherie de collusion largement vue dans le jeu populaire de *StarCraft*, dans lequel deux tricheurs se sont entendus l'un et l'autre comme suit : Chacun perd des points au profit de l'autre alternativement. La perte de l'un donnerait à l'autre un point de victoire, soulevant son rang d'échelle, et vice versa. Ainsi, deux d'entre eux pourrait grimper à des postes élevés dans l'échelle sans jouer à un jeu légitime.

II.2.3. Abuser de la procédure de jeu

Les joueurs peuvent effectuer cette forme de tricherie sans n'importe quelle sophistication technique parce que le tricheur abuse du mode opératoire du jeu. Un cas commun observable dans beaucoup de jeux en ligne est la fuite : les tricheurs se déconnectent du système de jeu quand ils sont sur le point de perdre.

II.2.4. Tricherie liée aux actifs virtuels

Dans certains jeux en ligne, les joueurs peuvent échanger des personnages virtuels et des objets qu'ils ont acquis avec l'argent réel. Un tricheur pourrait offrir un objet virtuel (tout bien ou actif) et recevoir en contrepartie de l'argent réel sans jamais livrer l'objet comme convenu.

II.2.5. Exploitation de l'intelligence artificielle

Les joueurs tricheurs peuvent parfois exploiter des techniques de l'intelligence artificielle (AI). De nos jours, beaucoup de programmes peuvent concurrencer des joueurs humains de haut niveau. Par exemple, en jouant aux échecs en ligne avec d'autres joueurs humains, les tricheurs peuvent rechercher les meilleurs coups pour leur prochain mouvement en exécutant un programme qui exploite la puissance de la machine.

II.2.6. Modification de l'infrastructure client

Sans modifier les programmes, les configurations, ou des données de jeu du côté client, les joueurs peuvent tricher en modifiant l'infrastructure du client, tels que les pilotes de périphériques dans leurs systèmes d'exploitation. Par exemple, ils peuvent modifier un pilote graphique pour faire un mur transparent de sorte qu'ils puissent voir derrière lui, localisant d'autres joueurs qui sont censés être cachés derrière lui. C'est la prétendue hack de mur (*Wall-hacking*), une tricherie populaire dans certains jeux en ligne.

II.2.7. Déni de service aux joueurs de pair

Les tricheurs peuvent gagner des avantages par déni de service à d'autres joueurs. Par exemple, les tricheurs mettent en retard des réponses de leurs adversaires par l'inondation de la connexion au réseau de ces derniers.

II.2.8. La tricherie de synchronisation

Les tricheurs peuvent retarder leurs propres mouvements jusqu'à ce qu'ils connaissent les mouvements de tous leurs adversaires, et gagnent ainsi un avantage énorme. Par exemple, un tricheur peut par exemple retarder l'envoi d'une mise à jour jusqu'à voir les mouvements de ses adversaires. Il envoi ensuite la mise à jour.

II.2.9. Compromettre les mots de passe

Les mots de passe sont souvent la clé d'une bonne partie des données et des autorisations qu'un joueur possède dans un système de jeu en ligne. En compromettant un mot de passe, un tricheur peut accéder aux données et à l'autorisation de sa victime dans le jeu. En raison des limitations de la mémoire humaine, certain joueurs choisissent des mots de passe faciles à mémoriser, tels que des numéros de téléphone, des anniversaires, ou des noms des amis ou de la famille, ou des mots communs dans leurs langues maternelles. Les tricheurs peuvent souvent facilement deviner de tels mots de passe faibles.

II.2.10. Exploitation du manque de confidentialité

Quand les programmes de jeu échangent les paquets de communication au format texte en clair, des tricheurs peuvent écouter ces paquets et insérer, supprimer, ou modifier des événements ou des commandes de jeu transmis sur le réseau. Cette forme de tricherie peut également avoir comme conséquence des mots de passe compromis quand ils sont transmis en clair au serveur.

II.2.11. Exploitation du manque d'authentification

Si aucun mécanisme approprié n'est mis en place pour l'authentification des clients auprès du serveur, les tricheurs peuvent rassembler beaucoup de paires d'identificateur/mot de passe des joueurs légitimes en installant un faux serveur de jeu. De même, si un mécanisme approprié n'existe pas pour authentifier un client, les tricheurs peuvent exploiter ceci pour gagner des avantages.

II.2.12. Exploitation d'un bug ou d'un point faible

Les tricheurs peuvent exploiter un bug ou un point faible dans les programmes de jeu ou de conception sans apporter aucune modification au code ou aux données de jeu.

II.2.13. Compromettre des serveurs de jeux

Les tricheurs peuvent altérer des programmes de serveur de jeu ou changer leurs configurations une fois qu'ils ont obtenu l'accès au système hôte du serveur de jeu.

II.2.14. Abus interne

Un opérateur de jeu a souvent des privilèges d'administrateur système que les initiés (employés de l'opérateur par exemple) peuvent facilement abuser. Par exemple, ils peuvent générer des objets virtuels de valeur en modifiant la base de données de jeu du côté serveur.

II.2.15. Ingénierie sociale

Les tricheurs peuvent essaye d'escroquer un joueur en lui faisant croire quelque chose d'attrayant ou d'ennuyeux lui est arrivé et qu'il lui est indispensable de révéler son identification et son mot de passe.

II.3. Taxonomie

Selon [Yan et Randell, 2005] on peut classer les formes courantes de tricherie en quinze catégories (voir **Tableau II. 1**). Ces formes de tricherie ne sont pas spécifiques aux jeux P2P. La tricherie est classifiée ici selon la vulnérabilité, la conséquence de la tricherie, et le tricheur principal .Toute forme de tricherie apparaît au moins une fois dans chacune de ces catégories.

Classification des différents types de tricherie	Vulnérabilité				Echec possible					Exploiteur(s)			
	L'insuffisance de conception du système		La vulnérabilité humaine		Violation de l'équité	Masquerade	Violation de l'intégrité	Déni de Service	Vol d'information	Indépendant(s)		Coopératif(s)	
	La base du système	Systèmes de jeux	joueurs	Opérateur de jeux						Un seul joueur	Opérateur de jeu	Plusieurs joueurs	Opérateur et Joueur
A) Exploitation de la confiance mal placée		●					●		●	●			
B) Collusion		●			●				●			●	
C) Abuser de la procédure de jeu		●			●					●			
D) Triche liée aux actifs virtuels			●		●					●			
E) Exploitation de l'intelligence artificielle		●			●					●			
F) Modification de l'infrastructure client	●						●			●			
G) Déni de service aux joueurs de pair	●	●						●		●			
H) La tricherie de synchronisation		●			●		●			●			
I) Compromettre les mots de passe			●						●	●			
J) Exploitation du manque de confidentialité		●					●		●	●			
K) Exploitation du manque d'authentification		●				●				●			
L) Exploitation d'un bug ou d'un point faible		●			●					●			
M) Compromettre des serveurs de jeux	●						●			●			
N) Tricherie lié à l'abus interne							●				●		●
O) Tricherie par l'ingénierie sociale			●						●	●			

Tableau II. 2 Taxonomie des différents types de tricherie [Yan and Randell, 2005]

II.3.1. Vulnérabilité

Certains tricheurs exploitent les insuffisances de conception dans les systèmes de jeu. Par exemple, la tricherie par l'exploit d'un bug exploite des insuffisances dans la conception du jeu, la mise en œuvre, ou tous les deux. Cependant, l'ingénierie sociale ne se base sur aucune insuffisance technique de conception. Les vulnérabilités exploitées par la tricherie sont classé en deux divisions :

- **L'insuffisance de conception du système**, qui implique un défaut de conception technique qui se pose au cours du développement du système. Elle a deux subdivisions qu'un tricheur peut exploiter :
  - *Insuffisance dans le système de jeu*. L'exploitation de la confiance mal placée, le manque de confidentialité ou la tricherie d'authentification, de synchronisation, et l'exploitation d'un bug tirent profit des insuffisances techniques dans le système de jeu et appartiennent à cette subdivision. La collusion, l'abus de la procédure de jeu, le déni de service et l'exploitation de l'intelligence artificielle peuvent également résulter des échecs techniques de conception dans le système de jeu.
  - *Insuffisance dans les systèmes sous-jacents.* La modification de l'infrastructure client et la compromission des serveurs de jeux appartiennent à cette subdivision.
- **La vulnérabilité humaine**, impliquant ceux qui gèrent ou participent à des jeux en ligne.

Les formes de tricheries, comme l'ingénierie sociale, compromettre les mots de passe, les tricheries liées aux actifs virtuels, et l'abus interne, sont relatifs à n'importe quelle insuffisance technique de conception.

II.3.2. Conséquence

La classification de conséquence de la tricherie est basé en grande partie sur les quatre aspects traditionnels de la sécurité informatique : confidentialité (empêcher la révélation non autorisée de l'information), intégrité (empêcher la modification non autorisée de l'information), disponibilité (empêchement retenant non autorisée de l'information), et authenticité (assurant l'identité d'un utilisateur distant quel que soit l'hôte de l'utilisateur). Une violation de la confidentialité entraîne un vol ou possessions d'informations, une violation d'intégrité entraîne une modification incorrecte des caractéristiques du jeu, une violation de disponibilité entraîne un déni de service, et une violation d'authenticité permet aux tricheurs de dissimuler leur identité.

La collusion, la compromission des mots de passe, ou l'ingénierie sociale peuvent avoir comme conséquence le vol des informations dans un jeu ;

Le déni de service aux joueurs de pair implique des dénis de service sélectif. La compromission des serveurs de jeux par la modification de l'infrastructure client ou l'abus interne implique généralement une modification incorrecte des caractéristiques de jeu, ou échec d'intégrité.

II.3.3. Le tricheur principal

Dans les jeux en ligne multi-joueurs, un joueur peut tricher à lui seul (tricherie indépendante) ou en collaboration avec d'autres joueurs (tricherie coopérative). Cette dernière exige deux joueurs ou plus en coopération, ceci comporte typiquement la collusion et l'abus interne de jeu spécifique.

II.4. Discussion

Cette taxonomie apporte une vue systématique sur la tricherie dans les jeux en ligne.

- Premièrement, les tricheurs en ligne ont exploité des vulnérabilités dans les systèmes informatiques et les humains qui participent aux jeux. Les tricheurs peuvent exploiter des insuffisances de conception dans des systèmes de jeu et des faiblesses dans leurs infrastructures de système sous-jacent. Ils peuvent également exploiter les joueurs innocents et corrompre des initiés.
- Deuxièmement, la classification principale de la tricherie prouve qu'un joueur peut commettre indépendamment plusieurs tricheries de jeu courant mais qu'une partie de tricherie implique la collusion des joueurs de pair ou d'un initié.
- Troisièmement, la classification de conséquence démontre que la tricherie est, en fait, en grande partie due à de divers échecs de sécurité.

II.5. Prévention

Il existe de nombreuses techniques de tricherie dans les jeux en ligne ce qui rend très difficile la création d'un système d'anti-tricherie. Les développeurs des jeux et ceux des logiciels ont développés plusieurs technologies qui ont pour objectif de contrer la tricherie. Des logiciels d'anti-tricherie sont couramment utilisés dans les jeux populaires tels que *Half-*

Life, *Quake*, ou *World of Warcraft*. Des exemples de ces logiciels incluent DMW anti-tricherie, *GameGuard*, *PunkBuster*, *VAC*, ou *Warden* (logiciel) [Robles et al., 2008]. Les exploits des bugs sont généralement résolus/supprimés via un patch pour le jeu, mais pas toutes les entreprises obligent les patchs et les mises à jour aux utilisateurs, en laissant la résolution réelle pour les utilisateurs individuels [Robles et al., 2008].

Certaines entreprises choisissent d'interdire les tricheurs suspects de leurs serveurs. Lorsque cela est fait par une liste noire de clé de série du jeu, le joueur est effectivement empêché de jouer. *Blizzard Entertainment* et *Valve Software* sont connus pour avoir interdit les joueurs, mais le nombre réel des joueurs est inconnu [Robles et al., 2008].

II.6. Mesures disciplinaires

Les jeux devraient inclure une liste exhaustive des termes et conditions qui permettront la résiliation des joueurs qui enfreignent les règles. Cependant, il est essentiel que des erreurs ne soient pas commises, des utilisateurs bannis à tort peuvent causer un chahut. Aussi, il pourrait être difficile d'interdire et de stopper les utilisateurs d'adhérer de nouveau, en particulier à partir des systèmes libres, qui ne nécessitent pas de numéro de cartes de crédit.

II.7. Conclusion

Faire un jeu en ligne parfaitement sécurisé est probablement impossible. Toutefois, il est certainement possible et souhaitable de réduire et limiter les abus, permettant aux clients une bonne expérience dans un monde virtuel. Une bonne conception et programmation, augmente la sensibilisation des utilisateurs, la maintenance et la supervision aidera à atteindre cet objectif.

La sécurité est la préoccupation majeure dans tous les jeux en ligne, mais en particulier dans les mondes persistants. Un trou de sécurité unique peut faire disparaître vos clients du jour au lendemain.

CHAPITRE III : Les approches anti-tricherie

III.1. Introduction

Le manque de prévention des tricheries dans les jeux P2P ralentit leurs percées commerciales. Dans l'architecture P2P, une partie de l'application est installée sur la machine des joueurs. Ces derniers sont capables de manipuler et de faire des modifications sur leurs machines, ce qui leur permet de contourner la protection de leurs PCs. Par conséquent, les MMOGs basés sur les architectures P2P sont vulnérables à la tricherie.

III.2. Les mécanismes de détection de tricherie

En raison des nombreuses formes existantes de la tricherie, il existe plusieurs solutions qui traitent du problème de sa prévention et de sa détection dans les MMOGs. Certaines sont conçues pour les systèmes client/serveur, d'autres pour les systèmes P2P alors que d'autres encore sont utilisables avec n'importe quel système.

III.2.1. Mécanisme de détection pour l'architecture client/serveur

III.2.1.1. Approche comportementale

Cette approche, proposée dans [Laurens et al., 2007], permet la détection de tricherie des joueurs (spécialement le *Wall-hacking*) à travers le monitoring et l'analyse de leurs comportements en temps réel. Elle présente une preuve à base de données statistiques pour distinguer entre les joueurs tricheurs des non tricheurs. Pour cela, l'approche se base sur l'hypothèse qu'un tricheur présente un comportement distinguable d'un joueur normal. En utilisant cette méthode, un tricheur peut être identifié en faisant abstraction de la méthode d'exploitation. Cela ajoute un niveau de protection pour une classe entière de jeux en ligne. Dans son fonctionnement, cette approche est très semblable à un IDS (*Intrusion Detection System*).

Comme preuve de concept, les auteurs de l'approche ont développé un système de détection basé sur le comportement en tant que plugin dans le moteur de jeu. Ce système se base essentiellement sur un type de tricherie Wall-hacking. L'architecture de cette solution comporte trois composants dont seuls les deux premiers sont liés au moteur de jeu :

1. Le composant d'entrée : il permet de récupérer des données spécifiques du joueur à partir du serveur de jeu.
2. Le composant de sortie : il envoie les données de tricherie au serveur de jeu.
3. Le composant "moteur d'analyse" : il permet de détecter les tricheries en utilisant des algorithmes de détection. En faisant abstraction du moteur d'analyse il est possible de perfectionner ou de mettre à jour ces algorithmes sans affecter le reste du système ou les clients.

La Figure III.1 montre le fonctionnement de ces composants.

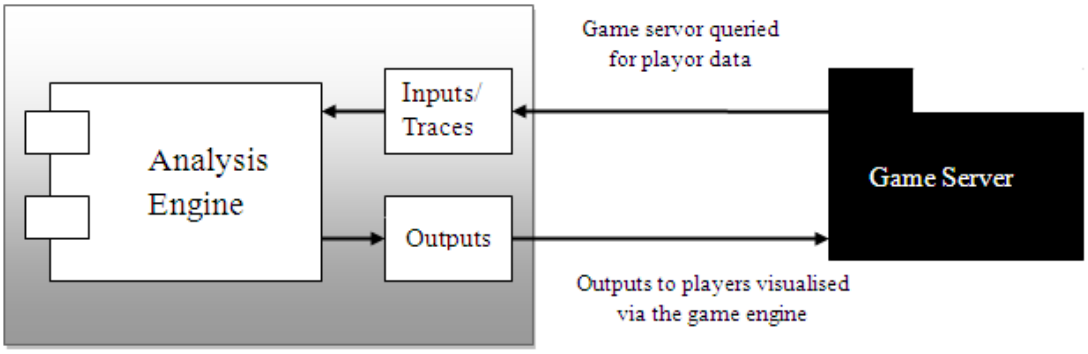


Figure III. 1 : Architecture du système à base d'approche comportementale [Laurens et al., 2007]

Le plugin doit être en mesure de détecter précisément ce que le joueur « contrôle » à tout moment, et identifie chaque objet dans l'univers du jeu. Ceci est possible grâce à un mécanisme prévu par le moteur Source appelé « trace ». Ce mécanisme prend la forme d'un vecteur qui, à son lancement, fournit des informations sur ce que le joueur a effectué comme opération dans l'univers du jeu.

Pour filtrer les objets qui sont opaques à une trace, un *Tracer* peut être combiné avec des filtres de trace. Dans ce cas, deux filtres sont utilisées, *World filter* qui renvoie uniquement les données sur la géométrie de base de l'univers du jeu que le joueur a touché (world trace) et *Entity filter* qui fait abstraction de la géométrie de l'univers, en retournant des informations sur l'entité touchée indépendamment du fait qu'elle soit derrière un mur (entité de trace).

Cette approche s'intéresse à la différence observable entre un joueur légitime et un tricheur à partir des données capturées. Quatre métriques ont été retenues à cause d'une différence notable entre les deux comportements :

- *Fréquence des traces illégales* : Une trace illégale est une trace dans laquelle le tir traverse un élément opaque de l'univers de jeu, par exemple un mur (tirer sur un joueur derrière un mur).

- Distance par rapport au matériel du monde du jeu : Alors qu'un joueur légitime tend à se focaliser sur le point le plus loin de son champ de vision, les éléments opaques du jeu (mur) ne sont pas un obstacle pour la vision d'un joueur qui fait du Wall-hacking. Un tel joueur montre de traces très petites vers les éléments opaques du monde du jeu (fixer un mur par exemple).

- *Distance aux traces illégales* : C'est la distance entre le joueur (derrière le mur) et sa cible. Un joueur qui fait du Wall-hacking (qui tire sur un joueur derrière un mur) est en général plus près du mur qu'un joueur qui tire sans faire exprès sur un autre derrière un mur. La distance est donc en moyenne inférieure pour un joueur qui triche.

- *Nombre de traces illégales consécutives* : un tricheur a tendance à tracer consécutivement un ennemi à travers le mur, ceci génère une suite de traces illégales. Un Wall-hacker génère un nombre de traces illégales (consécutives) beaucoup plus grand qu'un joueur honnête.

Pour combiner ces quatre métriques, un algorithme a été développé [Laurens et al., 2007]. Ce dernier enregistre les valeurs des quatre métriques dans un temps précis et calcule un paramètre de score de tricherie appelé « Cheat Score ». Ce paramètre permet de marquer un joueur comme un tricheur lorsque son Cheat score dépasse un certain seuil. Après des expérimentations sur deux cartes de jeu, les auteurs de l'algorithme ont fixé le seuil à environ 20 points.

III.2.1.1.1. Avantages

- Comme cette approche se base sur le comportement, elle fournit une méthode généralisée et abstraite de détection de tricherie. Elle fournit donc un niveau de protection pour une classe entière de jeu en ligne.
- Elle est indépendante de la technique de tricherie utilisée (Hook Hacks, etc …)
- Un nouvel exploit n'offre aucun avantage par rapport à un exploit connu.
- Elle diminue la pression sur les développeurs dans le cadre de développement de patch du moment qu'en général, les techniques de tricherie semblables peuvent être détectées automatiquement par un moteur de détection basé sur le comportement.
- Le système de détection peut être déployé complètement sur le serveur retirant au client la possibilité de modifier le code à son niveau. Ceci permet aussi de prévenir contre les attaques de types man-in-the-middle (interception des communications entre deux parties) et contre l'utilisation de proxy(s). Le client n'a pas l'accès au système de détection (déploiement sur le serveur) et il n'est pas possible d'utiliser un proxy car le système est relié au serveur directement.
- L'étude a montré l'efficacité du mécanisme de détection d'un joueur tricheur et non tricheur. En effet, le Cheat Score d'un tricheur est au moyen de 22 fois plus grand qu'un joueur normal.

III.2.1.1.2. Inconvénients

L'utilisation modérée du *Wall-hacking* rend le comportement du tricheur moins suspect, ce qui augmente la difficulté de le détecter par l'analyse des comportements des joueurs. Ceci augmente les chances du tricheur à ce qu'il ne soit pas détecté.

III.2.1.2. Approche bayésienne

Dans cette approche, les auteurs proposent une méthode pour la détection de la tricherie en utilisant les réseaux bayésiens dynamiques [Yeung et Lui, 2008]. Le cadre de détection de tricherie se base seulement sur les *Etats de jeu* et peut résider complètement du côté du

serveur sans nécessiter aucune modification du code client. Cela simplifie le processus de déploiement et améliore le cadre de détection qui se fait d'une manière transparente aux joueurs en cas où une nouvelle forme de tricherie est suspectée.

Un réseau bayésien *BN* est un graphe acyclique qui se compose d'un ensemble de nœuds et un ensemble d'arêtes dirigées. Chaque nœud représente une variable, et chaque arête orientée représente une influence causale à partir d'un nœud parent Xp à son enfant Xc. La probabilité d'un nœud X est conditionnée uniquement sur ses nœuds parents et conditionnellement indépendantes sur d'autres variables aléatoires.

Pour expliquer le mécanisme de fonctionnement de cette approche, prenons l'exemple suivant : la figure III.2.a illustre un réseau bayésien qui modélise la relation de causalité entre trois variables aléatoires : Tricherie ($\check{C}$), Distance ($\check{D}$) et la précision de visée ($\check{A}$). Ici les variables aléatoires de la tricherie et la distance sont les nœuds parents de précision de visée. Par conséquent, pour tout joueur (tricheur ou non), sa distance de la cible visée peut influencer le résultat de la précision de visée. Dans ce cas, la probabilité de la variable de précision de visée dépend des valeurs de la tricherie et de la distance, tandis que les variables de la tricherie et de la distance sont indépendantes de toute variable aléatoire.

La distribution de probabilité conjointe de ce réseau bayésien est donné par :

$$\mathbf{P}(\check{\mathbf{C}}, \check{\mathbf{D}}, \check{\mathbf{A}}) = \mathbf{P}\left(\tilde{\mathbf{A}} \,\middle|\, \mathbf{parents}(\check{\mathbf{A}})\right) \mathbf{P}\left(\tilde{\mathbf{C}} \,\middle|\, \mathbf{parents}(\check{\mathbf{C}})\right) \mathbf{P}\left(\tilde{\mathbf{D}} \,\middle|\, \mathbf{parents}(\check{\mathbf{D}})\right)$$

$$= \mathbf{P}(\tilde{\mathbf{A}} \,|\check{\mathbf{C}},\, \check{\mathbf{D}}) \mathbf{P}(\tilde{\mathbf{C}}) \mathbf{P}(\check{\mathbf{D}}) \qquad \textbf{Équation III. 1}$$

Dans certaines situations, les valeurs des deux variables aléatoires de la précision et de la distance sont observables et on peut déterminer la probabilité de la tricherie d'une variable donnée des valeurs connues, c'est-à-dire $\mathrm{P}(\check{\mathrm{C}}, \check{\mathrm{D}}, \check{\mathrm{A}})$. Cela peut se faire par inférence : inférer un réseau bayésien est la tâche de calculer la probabilité d'un nœud qui donne la valeur de tous autres nœuds. Pour inférer le réseau bayésien, il faut connaître les probabilités a priori de tous les nœuds du réseau. Ces probabilités peuvent être obtenues via le processus de formation qui consiste à prélever les valeurs des variables pour un ensemble de cas d'essai.

Par exemple, supposons que :

- La tricherie est une variable aléatoire discrète qui a deux résultats possibles, vrai ou faux, c.-à-d. $\check{C} \in$ [true, false].
- La variable de précision de visée est une variable aléatoire discrète avec un résultat de 1 (précise) ou 0 (imprécise), c.-à-d. $(\check{A}) \in [0, 1]$.
- La distance est une variable aléatoire discrète avec un résultat de 0 (c.-à-d., très proche de la cible) ou 1 (c.-à-d. loin de la cible), c.-à-d., $(\check{D}) \in [0, 1]$.

L'enregistrement des valeurs de la précision, la tricherie et la distance d'un joueur en particulier dans une session de jeu, donne un ensemble de données tel que celui montré dans le tableau III.1. Chaque ligne du tableau correspond à un échantillon de donné unique. Les résultats des distributions de probabilités sont les suivantes :

$$\mathrm{P}(\check{\mathrm{C}}) = \begin{cases} 0.4\ \check{\mathrm{C}} = \mathrm{true} \\ 0.6\ \check{\mathrm{C}} = \mathrm{false}, \end{cases}$$

$$\mathrm{P}(\check{\mathrm{A}}\,|\check{\mathrm{C}},\ \check{\mathrm{D}}) = \begin{cases} 0.25\ \check{\mathrm{A}} = 1, \check{\mathrm{C}} = \mathrm{false}, \check{\mathrm{D}} = 0 \\ 0.75\ \check{\mathrm{A}} = 0, \check{\mathrm{C}} = \mathrm{false}, \check{\mathrm{D}} = 0 \\ 0.50\ \check{\mathrm{A}} = 1, \check{\mathrm{C}} = \mathrm{false}, \check{\mathrm{D}} = 1 \\ 0.50\ \check{\mathrm{A}} = 0, \check{\mathrm{C}} = \mathrm{false}, \check{\mathrm{D}} = 1 \\ 0.50\ \check{\mathrm{A}} = 1, \check{\mathrm{C}} = \mathrm{true}, \check{\mathrm{D}} = 0 \\ 0.50\ \check{\mathrm{A}} = 0, \check{\mathrm{C}} = \mathrm{true}, \check{\mathrm{D}} = 0 \\ 1.00\ \check{\mathrm{A}} = 1, \check{\mathrm{C}} = \mathrm{true}, \check{\mathrm{D}} = 1 \\ 0.00\ \check{\mathrm{A}} = 0, \check{\mathrm{C}} = \mathrm{true}, \check{\mathrm{D}} = 1 \end{cases}$$

Tricherie ($\check{C}$)	Distance $\check{D}$	Précision de visée ($\check{A}$)
True	1	1
True	1	1
True	0	1
True	0	0
False	1	1
False	1	0
False	0	1
False	0	0
False	0	0
False	0	0

Tableau III. 1 : Données obtenues lors de l'étape de formation pour le réseau Bayesian de la figure III.2.a [Yeung et Lui, 2008]

Avec ces probabilités a priori, le réseau bayésien est inféré.

Supposons qu'à un moment donné, le tir est précis (c.-à-d., $\breve{A}$= 1) et le joueur est loin de la cible (c.-à-d., $\breve{D}$= 1). Maintenant, la probabilité de savoir si le joueur triche ou non est déduite à cet instant par la règle de Bayes, la probabilité est donnée par la formule suivante :

$$P(\breve{C} = T | \breve{A} = 1, \breve{D} = 1)$$

$$= \frac{P(\breve{A} = 1 | \breve{C} = T, \breve{D} = 1) P(\breve{C} = T) P(\breve{D} = 1)}{\sum_{\breve{C}} P(\breve{A} = 1 | \breve{C}, \breve{D} = 1)\, P(C) P(\breve{D} = 1)}$$

$$= \frac{P(\breve{A} = 1 | \breve{C} = T, \breve{D} = 1) P(\breve{C} = T)}{P(\breve{A} = 1 | \breve{C} = T, \breve{D} = 1) P(\breve{C} = T) + P(\breve{A} = 1 | \breve{C} = E, \breve{D} = 1) P(\breve{C} = F)}$$

$$= \frac{(0.5)(0.4)}{(0.5)(0.4) + (0.5)(0.6)}$$

$$= 0.4$$

Par conséquent, à tout instant, l'approche des réseaux bayésiens permet de déterminer la distribution de probabilité de($\breve{\boldsymbol{C}}$). Donc on peut savoir si le joueur triche ou non, sur la base des valeurs observées des autres variables aléatoires.

Bien que l'approche BN puisse aider à estimer la probabilité de savoir si le joueur triche ou non, l'approche a une limitation majeure car elle peut seulement modéliser le comportement statique d'un joueur. Etant donné que les résultats des variables aléatoires($\breve{\boldsymbol{C}}$), ($\breve{\boldsymbol{A}}$) et ($\breve{\boldsymbol{D}}$) peuvent être dépendantes du temps, une représentation temporelle est nécessaire pour prédire avec certitude si un joueur est un tricheur ou non.

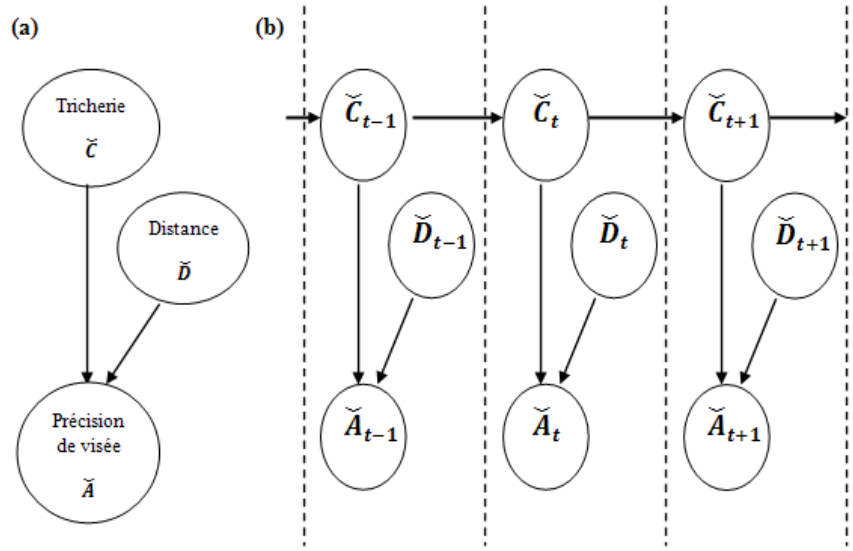


Figure III. 2 : Un exemple de réseau bayésien et de réseau bayésien dynamique [Yeung et Lui, 2008]

Lorsque des variables aléatoires prennent des valeurs différentes au cours du temps alors que la structure du réseau d'inférence reste inchangé, le modèle temporel résultant est appelé un réseau bayésien dynamique (DBN). Typiquement, un DBN est composé en reproduisant le même réseau bayésien pour les différents intervalles ou tranches de temps. Les dépendances existent non seulement entre les nœuds à l'intérieur de la même tranche de temps, mais un nœud dans une tranche de temps peut également dépendre du même nœud et/ou d'autres nœuds dans les tranches de temps précédentes. Par exemple la Figure III.2.b illustre un DBN formé en reproduisant le réseau bayésien dans la figure III.2.a à chaque tranche de temps.

La détection de la tricherie est présentée avec un moteur de jeu en ligne multi-joueurs. Pour démontrer l'efficacité de la méthode proposée, les auteurs de l'approche ont construit un prototype d'un système de jeu multi-joueurs en utilisant un jeu open source «tireur de premier-personne». Ils ont choisi un type spécifique de tricherie appelé *viser le robot* (ou aimbot). Pour réaliser ce système de détection dynamique, ils ont également mis en place une routine d'inférence bayésienne dynamique sur le serveur de jeu de sorte qu'ils pourront inférer la probabilité de tricherie pour chaque joueur individuellement au cours d'une session de jeu.

Dans un jeu en ligne multi-joueurs en temps réel, tous les joueurs doivent maintenir une vision cohérente des Etats de jeu. La perception de l'interaction en temps réel est obtenue en mettant à jour les états de jeu entre tous les joueurs fréquemment.

Avantages

- Le taux de faux positifs est faible.
- Elle est transparente aux joueurs ce qui facilite l'amélioration de cette approche si une nouvelle forme de tricherie est suspectée.
- Un nœud dans une tranche de temps peut également dépendre du même nœud et/ou d'autres nœuds dans les tranches de temps précédentes. Cela montre bien le comportement du joueur durant les différents tranche du temps, d'où la facilité de la détection du tricheur.

Inconvénient

- Les tricheurs détournent parfois le système en jouant sans utiliser exclusivement les robots (alternations entre les robots et le jeu humain) ce qui donne des résultats normaux de la probabilité, d'où l'augmentation de la difficulté de les détecter.

III.2.1.3. Approche automatique de détection de Bot

Un *bot informatique* est un agent logiciel automatique ou semi-automatique qui interagit avec des serveurs informatiques. Un bot se connecte et interagit avec le serveur comme un programme client utilisé par un humain, d'où le terme « *bot* », qui est la contraction (par aphérèse) de « *robot* ». On les utilise principalement pour réaliser des tâches répétitives que l'automatisation permet d'effectuer rapidement. Ils sont également utiles lorsque la rapidité d'action est un critère important [Chen et al., 2006].

L'approche automatique de détection de bot [Mitterhofer et al., 2009] est réalisée du côté serveur et se base seulement sur l'analyse du comportement de l'avatar. Ce dernier est contrôlé par un script qui automatise une séquence spécifique d'actions constamment répétées. Cette approche est complètement transparente aux utilisateurs. Elle se concentre spécifiquement sur le mouvement des avatars par l'extraction des waypoint (point de passage) qui décrivent le *path* (une séquence des endroits visités par l'avatar) parcouru et en trouvant les séquences répétées dans la *route* prise (le parcours du mouvement qu'un avatar effectue). Quand un avatar se déplace, le client de jeu envoie régulièrement des paquets avec de

nouvelles coordonnées au serveur. Les mouvements de l'avatar sont alors enregistrés au niveau du serveur et cette trace est utilisée comme base pour l'approche de détection.

Cette approche a été implémentée et évaluée pour le jeu *WOW*. Elle permet de détecter si un joueur utilise le bot, en commençant d'abord par traiter et transformer les données du mouvement d'un avatar puis interpréter les résultats pour détecter le bot. La figure III.3 représente les différentes étapes pour le traitement et la transformation du mouvement. Ces étapes sont :

- La collecte de données et la construction d'une route
- L'Extraction de Waypoints
- L'Abstraction de la représentation de la route et la découverte des répétitions

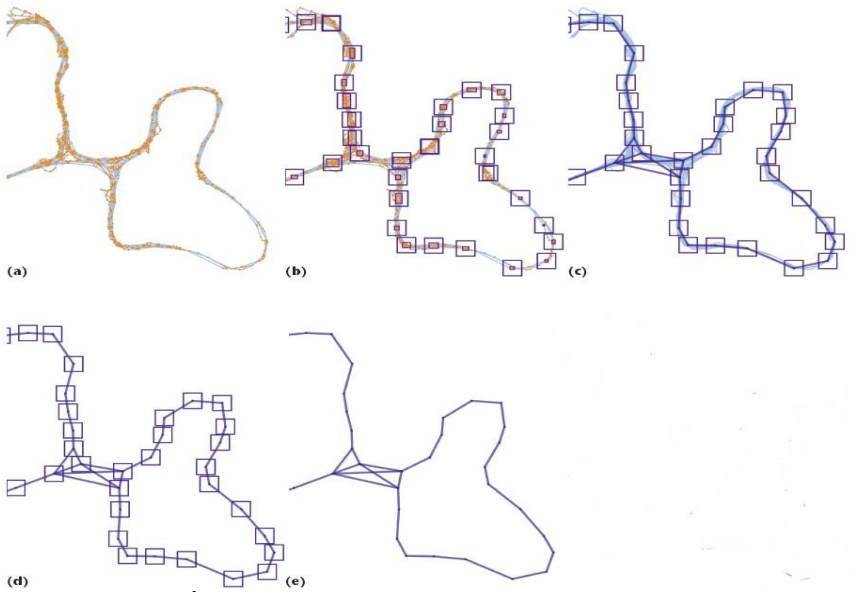


Figure III. 3 : Étapes de traitement de mouvement [Mitterhofer et al., 2009]

La Figure III.3.a montre les routes et points, établis après la suppression de points qui débordent. Cette figure représente la phase *de la collecte de données et de la construction de route*. Pendant cette étape, les points de déplacement de l'avatar envoyés au serveur, sont dessinés (points oranges) et les routes de cet avatar (courbes bleu) sont reconstituées en reliant ses différents points. Ce traitement est fait par l'intermédiaire de l'algorithme de simplification de ligne de Douglas-Peucker [Douglas et Peucker, 1973], qui a comme résultat une courbe simplifiée qui ressemble à la courbe originale dans certains niveaux de tolérance mais se compose de moins de sommets.

La Figure III.3.b, représentant la phase *d'extraction de Waypoints*, montre les *waypoints* extraits des accumulations de points (les petits rectangles violets marquent les clusters extraits)**.** Quand le Bot prend le même chemin plusieurs fois, les points de différentes exécutions s'accumulent dans des clusters avec haute densité. Pendant cette étape, ces différents clusters sont extraits et des *waypoints* sont créés où un *waypoint* est une zone d'un diamètre fixe centrée dans le cluster. Le diamètre doit être aussi petit que possible mais suffisamment grand pour que si un bot passe par un *waypoint* plusieurs fois dans le *path*, la zone traversée dans chaque exécution sera prise en charge. Cela est réalisé en utilisant l'algorithme de clustering personnalisé basé sur l'algorithme k-means [Eckey et al., 2002], où la taille de cluster est limitée à la taille de *waypoint*. Le résultat est un ensemble de *waypoints* répartis sur toute la longueur du *path* que l'avatar a pris.

La Figure III.3.c, représentant la phase *d'Abstraction de la représentation de route et la découverte des répétitions*, montre un *path* simplifié de l'avatar créé en connectant les waypoints avec des lignes droites. Dans cette étape, l'ensemble des waypoints est utilisé pour la représentation abstraite et simplifiée de la route. Cela est fait en faisant une itération sur la séquence initiale de points, et en enregistrant le waypoint correspondant pour chaque point. Si le point n'a pas de waypoint, il faut le supprimer. Comme chaque waypoint est unique, une séquence de waypoints peut décrire une route. Cette description est appelée «Séquence de mouvement». Pour découvrir les répétitions dans les séquences de mouvements, les auteurs utilisent un tableau étendu de suffixe [Gusfield, 1997] et ce à l'aide de la table LCP (Longuest Common Prefix). Cette table montre les sous-séquences répétées, en particulier, une grande table LCP signifie qu'une grande partie de la route a été répétée.

La Figure III.3.d montre la suppression des routes initiales (courbes bleu clair supprimées) et la Figure III.3.e monte le path simplifié avec les waypoints supprimés ;

Après avoir traité et transformé ces données du mouvement d'avatar, deux métriques permettent d'interpréter les résultats intermédiaires et de décider si le joueur est un BOT :

1. Passages moyens de segment de path **:** cette métrique prend en compte la façon dont un avatar se déplace souvent à travers les lieux qu'il visite sur sa route.

2. Répétitions dans la séquence waypoint : cette métrique prend en compte la quantité et la longueur des séquences répétées dans sa route.

Inconvénient

Les Bots suivent habituellement des paths plus ou moins exacts, mais ils les laissent parfois, ce qui augmente la difficulté de les détecter par l'analyse de mouvement.

III.2.2. Mécanisme de détection pour l'architecture Peer to Peer

III.2.2.1. Adressage de la tricherie dans MMOGs distribué

Cette approche, présentée dans [Kabus et al., 2005], montre les moyens d'appliquer les techniques de P2P pour réduire les coûts et fournir une protection contre la tricherie. La prévention des tricheries est par consensus, elle permet au pair la médiation pour décider si un pair est un tricheur ou non. Elle se base sur les quatre points suivants. La *diffusion distribuée d'état* : qui effectue le traitement des requêtes d'action. Le *contrôle mutuel* : l'état global de jeu est maintenu d'une manière distribuée sur les clients. Le *journal d'audit* **:** distribue également le calcul global d'état de jeu parmi les clients. Le *calcul de confiance* : agit sur l'hypothèse que les joueurs ne peuvent pas manipuler leur logiciel de client de quelque façon. Cela permettrait à la fois la prévention et la détection des fraudes provoquées par les clients piratés inutiles, en fournissant un environnement idéal pour les jeux en ligne distribués.

Diffusion distribuée d'état : Si le coût primaire d'une exécution d'un jeu est la bande passante, alors une des approches est de distribuer ce coût aux clients. Si plusieurs clients sont dans la même région et doivent être informés sur un même événement de notification d'état du jeu, ce message est envoyé à un seul client qui l'envoie ensuite aux autre clients de la même région. Cette approche permet d'économiser la bande passante du serveur là où il ya beaucoup de joueurs regroupés dans la même zone.

Contrôle mutuel : L'idée fondamentale derrière cette approche est " vous ne pouvez pas faire confiance à un seul client, mais faites confiance au consensus des multiples clients non affiliés ". La raison pour laquelle un client ne devrait pas participer dans le calcul de l'état global de jeu est que le client s'exécute sur l'ordinateur du joueur qui pourrait être altéré.

Le Journal d'Audit : Plutôt que d'essayer d'empêcher la tricherie en temps réel (coût en synchronisation), on peut plutôt essayer de détecter les tricheurs. L'idée est que, une fois le tricheur est désigné, son compte sera fermé et une restauration même partielle est envisagée.

Calcul de confiance : Le calcul de confiance (TC : Trusted Computing) peut apporter deux grands intérêts aux fournisseurs des jeux en ligne. En premier lieu, la possibilité que seuls les logiciels qui sont signés par le producteur (et donc de confiance) peuvent fonctionner sur un ordinateur qui appartient aux ordinateurs de confiance (TC : Enabled computer) ou dans un environnement spécialement sécurisé par le système d'exploitation. En deuxième lieu, le deuxième intérêt est la possibilité qu'un ordinateur appartenant aux ordinateurs de confiance (TC : Enabled computer), puissent prouver qu'il est digne de confiance aux autres systèmes.

Le premier intérêt peut garantir que le logiciel du client ne peut pas être manipulé, et le second permet au fournisseur de jeu d'identifier les clients dignes de confiance.

III.2.2.2. Approche de résiliation des coordinateurs compromis

Un coordinateur est un type de nœud P2P qui est chargé de recevoir et de diffuser les mises à jour des joueurs. Le coordinateur peut être soit un participant au jeu ou bien comme étant un dispositif dédié [Norden et Guo, 2007].

L'approche de résiliation des coordinateurs compromis [Norden et Guo, 2007] a pour but de sécuriser l'architecture P2P basée coordinateur, connue pour sa vulnérabilité aux actions de tricherie passive et active. Un attaquant peut être n'importe quel client ou coordinateur d'une zone ou un coordinateur d'une autre zone ou n'importe quelle entité du réseau qui s'intéresse à la perturbation du jeu.

L'architecture ACORN, que nous allons présenter, utilise cette approche et permet de détecter et diminue les attaques DoS.

Deux scénarios résultent de l'interaction entre les attaquants et les coordinateurs :

1) L'attaquant induit le coordinateur à la tricherie.

2) L'attaquant lance une attaque DoS contre le coordinateur.

Pour permettre de décrire objectivement l'architecture d'ACORN et d'une manière quantitative, les notations suivantes sont introduites :

- T_{cheat} : c'est le temps durant lequel le coordinateur peut mener une tricherie passive ou active.
- T_{act} : c'est le temps durant lequel le coordinateur produit une tricherie active.

L'architecture ACORN amène à la formalisation des deux problèmes suivants :

Formalisation du problème 1 : étant donné un coordinateur qui a été compromis (induit) à tricher passivement ou activement. L'objectif est de minimiser le T_{cheat} en déplaçant la fonctionnalité de coordinateur à travers les différents clients pour que le temps durant lequel le coordinateur triche (T_{cheat}) soit limité dans la durée entre le début de la tricherie et l'instant où la fonctionnalité est déplacée. Elle est applicable aussi quand le coordinateur est sous attaque DoS.

Formalisation du problème 2 : étant donné un coordinateur qui triche de manière active, l'objectif est de minimiser le T_{act}.

Pour la première formalisation ACORN propose de déplacer les fonctionnalités du coordinateur au coordinateur candidat pour minimiser T_{cheat}, son but est beaucoup plus la prévention. Son fonctionnement est comme suit :

L'architecture des coordinateurs dans ACORN introduit un nouveau type de nœud appelé *Coordinator Access Points* (CAP), et montre une seule région avec un coordinateur actif dans un ensemble de coordinateurs candidats. Les candidats peuvent être choisis au hasard ou en fonction des performances de leur connexion réseau. Le CAP joue le rôle d'intermédiaire de confiance entre les clients et les coordinateurs, et aussi il permet la communication client/coordinateur. Le bénéfice d'ACORN est analogue à l'expression : «tirer sur une cible mobile», ce qui résulte à un T_{cheat} réduit.

En général le changement de coordinateur survient dans deux cas : soit le coordinateur ne remplit pas son rôle, soit le coordinateur quitte la région. L'ajout dans ACORN est le changement planifié du déplacement de coordinateur. Donc le T_{cheat} est réduit par rapport au cas d'un «coordinateur statique».

ACORN a deux instances :

- *CAP-choice* : la prise de décision se fait par le CAP, protégeant le Coordinateur des clients.
- *Coordinator-choice* : se repose sur l'ensemble de la communauté P2P pour prendre la décision de déplacement du Coordinateur indépendamment.

Pour la deuxième formalisation, ACORN se base sur les tricheries actives menées par les coordinateurs. Deux nouveaux mécanismes de détection de tricherie sont proposés

- *K-Window* : est utile dans les scénarios où on ne peut pas faire confiance aux clients.
- *Checkpointing* : est une approche distribuée où les clients contribuent conjointement à la détection de tricherie.

Avantage

- Déplacement transparent (sans connaissances des clients).
- Mouvement sensible à la charge (changement du coordinateur basé sur la congestion imminente du coordinateur).
- N'importe quel hôte pourrait devenir un coordinateur avec un impact négligeable sur les performances du système, augmentant les avantages d'ACORN en fournissant un grand ensemble de coordinateurs candidats.
- Concernant le choix du coordinateur, il n'y a aucun point de défaillance unique évident, et n'importe quelle vulnérabilité dure seulement la durée d'intervalle de mouvement "$\mathbf{t_{move}}$".

Inconvénient

- Le CAP pourrait devenir un point de défaillance unique, car il est le seul gestionnaire des coordinateurs et donc s'il tombe en panne et comme la solution n'envisage pas de

plan de secours dans le cas où le CAP tombe en panne, ceci entraine une défaillance dans le fonctionnement du système.

- Le $\mathbf{t_{move}}$ étant constant, les événements de changement se produisent d'une manière régulière. Un attaquant peut prévoir quand agir, s'il peut prévoir quand il ya passation d'un coordinateur à un autre.

III.2.2.3. Approche basée sur la cohérence

Cette approche [Kabus et Buchmann 2007] s'intéresse aux attaques de type: Exploitation de Confiance mal placée, Exploitation du Manque de Confidentialité et la Collusion. Elle traite les points suivants :

- ❖ identification des attaques appropriées et leur impact possible.
- ❖ présentation d'une conception de système basée sur la réplique sélective qui adresse ces attaques.
- ❖ analyse des scénarios et les contre-mesures concrètes d'attaque.

Cette approche divise l'univers de jeu en un ensemble de régions de taille réduite afin de distribuer la charge des calculs. Chaque région est assignée à un groupe de nœuds qui maintient l'état de cette région. Les nœuds responsables qui contrôlent l'état d'une région s'appellent les *contrôleurs de région* (*Region Controllers*) (RCs). Chaque *RC* maintient l'état entier d'une région.

Le nœud d'un joueur se connecte à tous les membres du groupe qui est responsable de la région dans laquelle le joueur avatar est situé. Le nœud exécute un logiciel de client qui prend l'entrée du joueur et l'envoie comme une demande au groupe entier. Après le traitement de la demande, chaque RC du groupe envoie sa mise à jour d'état de nouveau au nœud du joueur. Le nœud prend comme réponse correcte celle qui a été envoyé par la majorité des RCs. La réponse contient des informations sur des changements de la région de jeu et le client la montre au joueur. Tant que les RCs défectueux ou malveillants sont dépassés en nombre par les corrects, l'état correct de jeu sera montré.

Concernant le problème de *manque de confidentialité*, le système proposé peut prévenir le stockage des données sur des clients non autorisés si nécessaire, mais le cas d'un nœud autorisé révélant des données confidentielles au nœud non autorisé ne peut pas être empêché par des moyens techniques. Ce système exige que chaque message soit signé par l'expéditeur.

Cette approche de réplique soulève trois problèmes principaux : le *placement de réplique* qui sera géré par des super-pairs contrôlés par le fournisseur de jeu, la *cohérence* qui considère l'état de jeu correct comme étant l'état déterminé par un vote de majorité et ne doit pas laisser des répliques dans des états différents et la *propagation de mise à jour*.

Afin de créer un système entièrement fonctionnel, le système nécessite deux services supplémentaires : Le service de gestion et le service de persistance.

Le **service de gestion :** c'est le premier point de contact pour que les joueurs joignent le jeu. Il maintient une liste de tous les RCs qui sont actuellement dans le jeu et il tout nouveau client au sujet des RCs responsables de la région. Sans la permission du service de gestion, aucun client ne peut joindre le jeu.

Le service de persistance : tous les RCs de cette région envoient les modifications depuis la dernière sauvegarde au service de persistance. Ce dernier détermine l'état correct en choisissant celui qui tient la majorité. Il est nécessaire pour créer une sauvegarde périodique de son état, ce qui est important en cas de plantage du système ou d'arrêt pour des raisons de maintenance.

Afin de comprendre comment le système proposé détecte et empêche la tricherie des attaques, prenons différents scénarios pouvant se produire lors du fonctionnement. Considérons d'abord le cas idéal où seulement les nœuds bénins sont présents. Par la suite, nous allons examiner les attaques possibles des nœuds malveillants qui peuvent être effectuées en fonction du rôle qu'ils jouent, soit un client d'interface ou un contrôleur de région.

Premier cas où aucun tricheur n'est présent : Le système fonctionne comme suit. A chaque fois qu'un joueur veut joindre une session de jeu, le logiciel client se connecte au service de gestion qui est responsable de l'authentification du joueur. Si l'authentification est

réussie, l'état de l'avatar du joueur est récupéré à partir du Service de persistance et est communiqué au groupe responsable de RC. Tous les RCs communiquent l'état de l'avatar ainsi que les informations nécessaires sur l'environnement au client du joueur. Considérant que tous les RCs sont honnêtes, le joueur devrait recevoir les mêmes données de tout le RCs. Maintenant, les RCs et le client du joueur, possèdent toute l'information nécessaire pour permettre au joueur de commencer à jouer. Dès que le client affiche l'information reçue, le joueur peut commencer à agir avec l'univers de jeu. Pour ce faire, le logiciel du client traduit ses entrées en requêtes d'action et les envoie à chaque RC sur la liste qui a été reçue pendant la connexion. Tous les RCs reçoivent la même requête d'action, les traitent et mettent à jour leur état de jeu local en conséquence. Après que la transition d'état de jeu ait été exécutée, le nouvel état doit être communiqué à tous les clients intéressés. Tous les RCs envoient un paquet de mise à jour à chaque client qui contient seulement les changements qui sont pertinents pour ce client. Puisque l'état sur tout le RCs est identique, toutes les mises à jour reçues par un client simple sont égales. Le client incorpore la mise à jour à son état local de jeu et après cela il est prêt à émettre la prochaine requête d'action. Lorsque le joueur quitte le jeu, l'état de l'avatar doit être écrit dans un stockage persistant pour qu'il puisse être rétabli dès que le joueur décide de rejoindre à nouveau. Le client émet une commande de déconnexion du système à tout le RCs. A la réception de cette commande, les RCs envoient l'état actuel au service de persistance. Puisque l'état sur tout le RCs est le même, le service reçoit le même état d'avatar de tout le RCs et l'écrit dans un stockage persistant.

Deuxième cas où des différents scénarios de fraude peuvent être rencontrés : le système fonctionne comme suit. Tous les messages échangés dans le système doivent être signés par l'expéditeur. Le système n'accepte que les messages qui sont correctement signés. Puisque n'importe quelle communication est faite via des messages de réseau, la tricherie est faite en envoyant des messages falsifiés, des messages anormaux à différents récepteurs ou des messages omettant. Toutes les contre-mesures supposent que la majorité de RCs se comporte bien.

Cas des messages falsifiés : un message est dit falsifié quand son contenu est dévié de celui d'origine qu'aurait été envoyé par un expéditeur honnête. Il peut être falsifié directement ou bien indirectement suite à une manipulation d'état. Si l'état local d'un RC a été manipulé, il peut envoyer une mise à jour à ses clients qui diffère d'une mise à jour envoyée par un RC

honnête. Notons que l'incitation pour envoyer des messages falsifiés est assez faible puisque l'émetteur peut toujours être identifié par la signature.

Cas des messages anormaux : ce sont des messages anormaux utilisés seulement pour perturber le jeu et non pas pour avantager un quelconque joueur. Les expéditeurs de tels messages peuvent être identifiés à travers la signature.

Cas des messages omis : Par le non envoi d'un message, il n'est pas possible de changer l'état de jeu et donc un joueur ne peut gagner aucun avantage. Ainsi l'incitation pour cette attaque peut être faible. Les RCs qui n'envoient pas des messages de mise à jour ou de sauvegarde n'ont aucun effet. Puisque la majorité envoie les messages corrects, le procédé de vote donne le résultat correct.

Apres avoir présenté les différents cas possible de tricheries, le système les traite comme suit :

Les RCs qui émettant des mises à jour de déviation doivent être corrigés ou remplacés. Ceux qui continuent à envoyer des mises à jour de déviation sont probablement malveillants et doivent être remplacés.

Le processus de remplacement d'un RC peut être lancé par les clients qui peuvent facilement détecter des différences dans les mises à jour reçues. Si une majorité de clients conteste un RC, le service de gestion peut le remplacer par un autre. Alternativement, le service de persistance peut détecter la déviation des RCs et initier le remplacement.

Avantages

- Réduction de la charge sur le réseau du moment que l'univers est divisé en plusieurs régions.
- Réduction du risque de la tricherie grâce au principe de cohérence.

Critique

- L'assertion que la majorité de RCs se comportent bien n'est pas une certitude, donc ces contre-mesures restent à prouver.

III.2.2.4. Approche basée sur la signature des événements

Cette approche [Chan et al., 2008] propose un protocole de signature d'événements appelé **EASES** (*Efficient and Secure Event Signature*) pour signer les messages des événements discrets, d'une manière efficace et sécurisée. Elle se base sur le concept de la signature jetable par la génération d'une série de clés de chaînes de hachage. Cette approche prend en considération la réalisation de l'authentification et de la non-répudiation des signatures digitales sans le risque de coûts élevés induits par la signature des messages.

Le protocole EASES s'inspire principalement des deux protocoles NEO [Dickey et al., 2004] et SEA [Corman et al., 2006] qui utilisent les signatures digitales pour signer chaque mise à jour d'événement afin d'éviter, avec une faible latence, la tricherie dans les MMOGs P2P. EASES comporte quatre phases :

1. La phase d'initialisation : après l'établissement de la connexion, chaque joueur produit une série de clés pour la signature des mises à jour d'événement.
2. la phase de signature : chaque joueur signe sa mise à jour d'événement.
3. la phase de vérification : après que les mises à jour d'événement des autres joueurs soient reçues, chaque joueur peut vérifier ces dernières.
4. la phase de réinitialisation : les joueurs peuvent régénérer de nouvelles clés de signature lorsque les clés ont été utilisées.

Cette approche décrit également une version dynamique d'EASES qui n'exige pas de la pré-production des clés de hachage des chaînes afin de réduire le temps de préparation de clé et l'utilisation de mémoire. Cette version nécessite que deux phase, la phase de signature et la phase de vérification.

Avantages

- EASES a le calcul et le coût de la bande passante faibles,
- Puisque les deux phases d'initialisation ou de réinitialisation des chaînes de hachage ne sont pas nécessaires, un adversaire ne peut pas s'imposer puisqu'il n'obtient aucune connaissance au sujet de la clé de validation.

III.2.2.5. Approche basée sur la sélection des arbitres sécurisés

Cette approche [Daniel et Soh, 2008] traite du problème de sélection d'arbitres (RSP: *Referee Selection Problem*) de confiance de simulation et validation du jeu pour fournir une sécurité équivalente à la solution C/S. En effet, la sélection d'arbitres de confiance est un problème critique puisqu'ils sont sélectionnés de l'ensemble des pairs du réseau et généralement, ces derniers ne sont pas dignes de confiances.

La sélection d'arbitres a aussi pour but d'essayer de maximiser la réactivité, correspondant à la minimisation du temps de réponse, et maximiser l'équité, correspondant à la minimisation de l'écart de temps de réponse entre arbitres et joueurs. Ce problème est traité par les deux algorithmes de sélection d'arbitre sécurisés (SRS: *Secure Referee Selection*), SRS-1 et SRS-2. Ces deux algorithmes assurent que la probabilité des arbitres corrompus contrôlant une zone reste au-dessous d'une limite prédéfinie, tout en essayant de maximiser la réactivité (temps de réponse minimal) et l'équité (l'écart des temps de réponse entre les joueurs est minimal.

L'objectif de cette approche est le choix des arbitres multiples pour chaque zone. Ces derniers sont utilisés pour empêcher un seul fauteur (devenir un arbitre) de perturber le jeu, et empêcher aussi les nœuds corrompus qui complotent ensemble pour perturber le jeu (un arbitre corrompus avec d'autres joueurs (les joueurs de collusion)).

Le serveur d'authentification est chargé d'authentifier et de valider les joueurs joignant et de choisir des pairs pour agir comme arbitres.

Le modèle de ce système définit les facteurs suivants :

- **P**= $\{P_i \mid$ i est l'identifiant unique (ID) de chaque joueur$\}$ est l'ensemble des pairs/joueurs dans le jeu.
- **R= {**$R_f \mid$ f est l'identifiant unique (ID) de chaque arbitre**} ⊆ P** est l'ensemble des arbitres dans le jeu.
- $\mathbf{Z_P \subseteq P}$ est l'ensemble des joueurs situés dans la zone Z.
- $\mathbf{Z_R \subseteq R}$ **et P** est l'ensemble des arbitres contrôlant la zone Z.
- **C** = $\{C_i \mid C_i$ est un ensemble de joueurs de collusion$\}$

- La taille du plus grand groupe de pairs de collusion, maxC ≈ max ($|C_i|$).
- $\mathbf{Z_R \cap Z_P = \emptyset}$ c.-à-d. qu'un arbitre ne doit pas contrôler la zone dans laquelle est situé son avatar. En d'autres termes, chaque R_f dans Z_R n'est pas dans Z_P.
- Le nombre d'arbitres par zone $|Z_R|$, est définit par le développeur. Chaque joueur dans Z_P reçoit l'état de jeu de tous les R_f dans Z_R et prend le résultat de majoritaire ; donc, il exige au moins $[|Z_R|/2]$ arbitres de collusion contrôlant une zone pour trifouiller l'état de jeu.
- $\mathbf{D_{i,f}}$ est le temps de réponse d'un joueur P_i à un arbitre R_f. Le temps de réponse est symétrique i.e., $D_{i,f}= D_{f,i}$.

Le problème de choix d'arbitre est de sélectionner un ensemble arbitre Z_R, en prennent en compte les critères suivants :

1. la probabilité S_Z qu'il y ait $[|Z_R|/2]$ ou plus d'arbitres de collusion n'est pas plus grand que S_{max}, avec S_{max} est l'éditeur prédéfini $0 \leq S_{max} \leq 1$.
2. le temps de réponse moyen de pair à arbitre $\overline{d_{i,Z_R}}$ est minimisé par l'équation suivante :

$$\overline{\mathbf{d_{i,Z_R}}} = (\sum_{\mathbf{P_i \in Z_P . R_f \in Z_R}} \mathbf{d_{i,f}})/(|\mathbf{Z_P}| \times |\mathbf{Z_R}|), \qquad \textbf{Équation III. 2}$$

3. la différence, Δ, entre le maximum et le minimum de temps de réponse joueur-arbitre pour tous les pairs dans Z_P est minimisée

III.2.2.5.1. Algorithme de sélection d'arbitre

Pour traiter ces critères il faut :

- Connaitre tous les temps de réponse de pair-à-pair. Pour cela l'approche propose l'utilisation des coordonnées du réseau [Dabek et al., 2004] pour estimer le temps de réponse de pair-à-pair. Chaque pair calcule ses propres coordonnées de réseau et les transmet au serveur d'authentification. Ces coordonnées fournissent une bonne évaluation de temps de réponse entre deux pairs quelconques i et j, même si aucune mesure directe entre i et j n'a été faite.
- Calculer la taille de l'ensemble d'arbitre candidat : cette taille, dénotée par Ψ, est calculée en sorte qu'elle soit minimale et permettant que la sélection aléatoire

des pairs de $|Z_R|$ à partir de Z_{CR} limite la probabilité S_Z de sélection d'au moins $\alpha = [|Z_R|/2]$ des arbitres de collusion à pas plus que S_{max}. La valeur S_Z est donnée par la formule suivante :

$$\mathbf{S_Z} = \frac{\sum_{\mathbf{i=\alpha}}^{\mathbf{|Z_R|}} \binom{\mathbf{maxC}}{\mathbf{i}} \binom{\mathbf{\Psi - maxC}}{\mathbf{|Z_R|}}}{\binom{\mathbf{\Psi}}{\mathbf{|Z_R|}}} \qquad \textbf{Équation III. 3}$$

Algorithme SRS-1 :

L'algorithme SRS-1 insiste sur la réactivité (critère (2)) tout en satisfaisant la sécurité (critère (1)). Etant donné la taille minimum pré-calculées $\Psi = |Z_{CR}|$, SRS-1 exécute trois étapes:

(i) Sélectionner l'ensemble d'arbitre candidat $Z_{CR} \subseteq Z_{RP}$, telles que la sélection de n'importe quel sous-ensemble $Z_R \subseteq Z_{CR}$ donnera une petite valeur de $\overline{D_{i,Z_R}}$,

(ii) Sélectionner aléatoirement des arbitres $|Z_R|$ à partir de Z_{CR} ;

(iii) Augmenter artificiellement le temps de réponse de pair à $d_{F\%}$ (défini plus tard) tels que F% des pairs ont de temps de réponse égal.

Pour l'étape (i), SRS-1 sélectionne des arbitres proches de la majorité de joueurs. Soit (x_i, y_i) les coordonné de P_i. Il définit le *major* comme le point dans l'espace de coordonnées qui est le plus proche de la majorité de joueurs dans Z_P. La distance totale, TD(x, y), d'une coordonnée (x, y) à tous les joueurs est :

$$\mathbf{TD(x,y)} = \sum_{\mathbf{P_i \in Z_P}} \mathbf{dist}((\mathbf{x,y}),(\mathbf{x_i}, \mathbf{y_i})) \qquad \textbf{Équation III. 4}$$

où dist(A, B) est la distance entre les points A et B. Formellement, le *major* est le point qui minimise l'Équation III.4. SRS-1 formera Z_{CR} dans l'étape (i) par la sélection Ψ pairs pas dans Z_P qui sont les plus proches de la coordonnée major. Ainsi, la sélection des arbitres proches du major adresses le critère(2). La sélection arbitre au hasard dans l'étape (ii) les adresses critère (1). La sélection aléatoire d'arbitre dans l'étape (ii) adresse le critère (1). Ainsi l'étape (iii) adresse le critère (3).

Le mécanisme pour diminuer l'écart de temps de réponse est qu'un arbitre peut envoyer des mises à jour en retard à un pair avec un temps de réponse réduit de sorte qu'il reçoive des mises à jour en même temps qu'un pair un avec un temps de réponse grand, donnant artificiellement ces pairs le même temps de réponse. Nous définissons le poids d'équité $0 \leq F \leq 100\%$ dans l'étape (iii) comme le pourcentage minimum des pairs qui doivent avoir un temps de réponse égal moyen aux arbitres. Spécifiquement, du moment que pour chaque pair, nous calculons le temps de réponse moyen avec tous les arbitres puis nous calculons dF% (le temps de réponse maximum parmi le plus rapide F% de joueurs), et augmentons le temps de réponse du F% le plus rapide de pairs à dF%.

Algorithme SRS-2 :

L'algorithme SRS-2 souligne l'équité (critère (3)) tout en satisfaisant sécurité (critère (1)). Comme pour SRS-1, SRS-2 comprend trois étapes principales :

(i) Sélectionner un ensemble d'arbitre $Z_{CR} \subseteq Z_{RP}$ tels que la sélection de sous-ensemble $Z_R \subseteq Z_{CR}$, encourra une petite valeur d'inflation $d_{F\%}$;

(ii) Sélectionner aléatoirement des arbitres $|Z_R|$ à partir de Z_{CR} ;

(iii) Augmenter artificiellement le temps de réponse de pair à $d_{F\%}$ tels que F% des pairs ont de temps de réponse égal.

Notez que les étapes (ii) et (iii) de SRS-1 et SRS-2 sont identiques.

Cet algorithme adresse le critère (3) par le choix des arbitres qui minimise l'écart de temps de réponse du pair.

Par conséquent, SRS-2 choisit des arbitres près du centre de Z_P, choisissant des arbitres tels que le temps de réponse moyen de tous les arbitres aux autres joueurs est minimisé. Soit le *centro* le point dans l'espace de la coordonnée telle que la distance maximale à tous les joueurs dans Z_P est minimisé. Soit la distance maximale, MD(x, y), d'une coordonnée (x,y) à tous les joueurs soit :

$$\mathbf{MD(x, y) = max\ (\forall P_i \in Z_P, dist\big((x, y), (x_i, y_i)\big))} \qquad \textbf{Équation III. 5}$$

Le Centro est le point qui minimise l'Équation III.5.

Avantages

- Elle empêche la tricherie par les joueurs de collusion (les joueurs qui complote avec des arbitres corrompus).
- Elle assure un temps de réponse minimal, d'où la réactivité du jeu est assurée.
- Elle assure l'équité.

III.2.2.6. Approche basée sur une architecture d'anti-tricherie dynamique (DACA)

L'approche DACA [Liu et Tang, 2009] s'intéresse principalement à la tricherie artificielle, et les programmes de type BOT ne sont pas pris en compte. L'architecture proposée est hybride : une partie est gérée par un serveur central qui dirige l'exploitation du système global, et une autre réside au niveau de certains nœuds P2P pour effectuer le calcul de jeu et les communications des joueurs.

Cette approche divise l'environnement de jeu virtuel en plusieurs zones de différentes tailles et la taille de chaque zone dépend de sa population. Chaque zone forme un groupe de pair-à-pair (P2P) et s'autogouverne pour réduire la charge. La carte de jeu peut être divisée en forme de grille statique et les joueurs dans la même grille établissent un groupe de P2P.

DACA propose une division simple pour constituer un groupe P2P, appelée division **WRD** (*Weighted Responsible Division*), qui prend en considération les changements dynamiques et les différents niveaux de popularité des hotspots, réduisant ainsi le taux de changement de zone et offrant un meilleur équilibrage de la charge.

III.2.2.6.1. Architecture et fonctionnalités générales de DACA

- **L'architecture du système :** c'est une architecture hybride composée de trois couches **:** La couche la plus basse est la carte du jeu, elle est divisée en grille de tailles différentes avec des *hotspots* (point chaud, concept que nous avons défini dans I.2.3.4) au centre. La deuxième couche, est la couche d'action, c'est la couche où les joueurs peuvent agir les uns avec les autres et jouer. La couche supérieure est la couche de contrôle**,** elle contient

le data holder pour superviser le fonctionnement du groupe associé à P2P et trois types de serveurs : authentification, jeu, et base de données.

- **La division d'univers de jeu :** Une carte de jeu est divisée en une grille formée de zones en prenant en considération la distribution des joueurs, la propriété de changement dynamique de chaque zone, le comportement d'itinérance des joueurs, la charge de déséquilibrement et le changement de zone fréquent. La division de l'univers de jeu en DACA a les caractéristiques suivantes : équilibrage de charge, taux de transfert faible, adaptation et la diversité.
- **Fonctionnement du système** : Un groupe de P2P est constitué de l'ensemble des joueurs d'une zone, incluant un data holder sélectionné aléatoirement par DACA d'une autre zone. Le fonctionnement du système repose sur les trois procédures suivantes : connexion, le jeu, et le transfert.

III.2.2.6.2. Mécanisme d'anti-tricherie

Au cours du jeu, chaque joueur diffuse toutes ses actions aux membres de son groupe, calcule les actions de ces membres et envoie les résultats à son data holder. Chaque zone est contrôlée par trois entités : **data holder**, **data verifier** et **computation verifier**.

- **Data holder :** Il contrôle toutes les informations requises d'une zone. DACA attribue à chaque zone un data holder qui joue dans une autre zone pour éviter la tricherie basée sur la considération égoïste.
- **Data verifier :** Pour éviter la possibilité que le data holder fournisse à ses joueurs des informations de zone fausses, le serveur de base de données sélectionne de la même zone considérée un joueur au hasard, désigné comme *data verifier* (vérificateur de données), pour fournir des informations de cette zone. Ce dernier est responsable pour faire la vérification de la cohérence des données à partir du serveur de base de données et du data holder. En cas d'incohérence, le *data verifier* informe le serveur de jeu pour revérifier la légalité du data holder.
- **Computation verifier** : Pour éviter la tricherie en groupe, pour chaque zone, DACA sélectionne au hasard un joueur d'une autre zone, désignée comme *computation verifier* (vérificateur de calcul), pour joindre le calcul de jeux. Ainsi, chaque

computation verifier joint deux groupes P2P, calcule toutes les actions des joueurs dans ces deux groupes, et les synchronise avec les deux data holders correspondants.

Trois scénarios de tricherie peuvent se produire :

1. **Pas de tricherie :** Dans une zone sans joueurs tricheurs, les résultats de calcul que le data holder reçoit sont tous cohérents. Le data holder met à jour ces résultats avec le serveur de base de données et ses membres de groupe. Puisque tous les résultats sont cohérents, il n'existe aucune anomalie et le jeu se déroule correctement.
2. **Tricherie minoritaire :** Dans une zone avec peu de joueurs tricheurs, les résultats que le data holder reçoit sont incohérents. Le data holder enregistre les résultats de la majorité et met à jour les résultats principaux avec le serveur de base de données et ses membres. Ceci est motivé par le fait que la plupart des joueurs sont légitimes et les résultats de la majorité sont corrects et cohérents.
3. **Tricherie majoritaire :** Si la majorité des joueurs d'une zone trichent ensemble, la plupart des résultats que le data holder reçoit sont cohérents et enregistrés comme résultats de cette période de temps. Le data holder met à jour les résultats avec le serveur de base de données et les diffuse à ses membres de zone. Ces résultats enregistrés sont incorrects ; cependant, comme il existe quelques ou au moins un des joueurs qui sont légitimes - ceux choisis aléatoirement par DACA -, ces derniers signalent l'incohérence des résultats. De ce fait, le serveur de jeu restaure l'état de la zone à l'enregistrement précédent et rejoint le calcul de cette région. Tout joueur ayant communiqué des résultats de calcul signalés comme incohérents avec ceux du serveur de jeu seront considérés comme tricheurs.

III.3. Conclusion

Dans ce chapitre nous avons montré les différentes approches de détections de la tricherie dans les MMOGs, nous avons constaté que :

Pour les solutions d'anti-tricherie basées sur les MMOGs à architecture C/S, l'approche bayésienne semble être efficace car elle peut efficacement détecter les tricheurs et que le taux de faux positifs est faible, de plus elle est transparente aux joueurs ce qui facilite l'amélioration de cette approche si une nouvelle forme de tricherie est suspectée.

Pour les solutions d'anti-tricherie basées sur les MMOGs à architecture P/P, nous remarquons que n'existe pas une solution P2P pure, ce sont des solutions hybrides avec un serveur central pour l'authentification et l'abonnement pour diriger l'exploitation du système global, et une partie avec certains nœuds P2P pour effectuer le calcul de jeu et les communications des joueurs. Ces approches ont respecté les considérations générales dans un MMOG tel que la cohérence, le temps réel, la scalabilité et la sécurité, et elles ont aussi réglé le problème de la charge et la saturation de bande passante en laissant une certaine liberté de gestion aux nœuds.

Dans le chapitre prochain, nous allons faire une synthèse sur ces différentes approches et nous discuterons l'approche que nous allons proposer.

CHAPITRE IV : Proposition : DACA-Extended

IV.1. Introduction

Aujourd'hui, les jeux en ligne massivement multi-joueurs (MMOGs) connaissent une forte expansion. Nous trouvons des millions d'utilisateurs enregistrés et des centaines de milliers de joueurs concourants actifs.

À ce jour, l'architecture dominante pour l'implémentation d'un MMOG est l'architecture Client/serveur (C/S), car elle est relativement facile à mettre en œuvre et sécurisée. Mais elle reste vulnérable aux tricheurs qui profitent des failles des jeux et du système. Dans cette architecture classique, un serveur de jeu centralisé est nécessaire pour héberger un monde virtuel persistant. Vu l'augmentation explosive des joueurs, la scalabilité du serveur de jeu est devenue un défi crucial. De plus l'architecture C/S a rencontré beaucoup de problèmes qui restent non résolus, telles que la fiabilité, la surcharge du réseau (problème de bande passante) et du serveur de jeu, la tolérance aux pannes et le coût.

L'architecture Pair à Pair (P2P) se présente comme une alternative intéressante à l'architecture C/S pour les MMOGs, car elle exploite les ressources informatiques des joueurs, et peut effectivement organiser la charge de travail imposée aux serveurs de jeu. Récemment, la puissance de calcul disponible sur les ordinateurs personnels a augmenté de façon spectaculaire avec une bande passante réseau réduite. L'architecture P2P a non seulement le potentiel de réduire le coût d'un MMOG, mais peut aussi éviter les problèmes des architectures C/S comme la scalabilité, la fiabilité, la tolérance aux pannes et la redondance.

Dans l'architecture P2P, une partie de l'application est installée sur la machine des joueurs. Comme le joueur a un accès illimité sur sa propre machine, ce dernier est capable de manipuler et de faire des modifications sur sa machine, ce qui lui permet de contourner la protection de son PC. Par conséquent, les MMOGs basés sur les architectures P2P sont vulnérables à la tricherie.

Différentes approches ont été présentées pour la détection de la tricherie dans les MMOGs, certaines proposent des solutions qui traitent les systèmes C/S et d'autres les

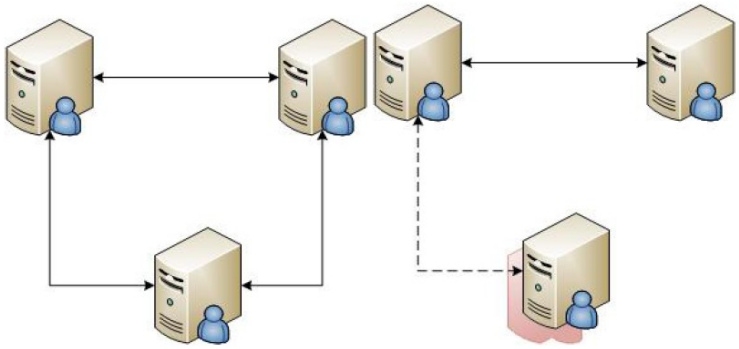
systèmes P2P. Dans notre travail, nous nous intéressons aux solutions qui traitent les systèmes P2P vu que cette architecture est de plus en plus préconisée dans les MMOGs.

IV.2. Synthèse des solutions d'anti tricherie existantes

D'après les approches que nous avons étudié, nous avons constaté que :

- Les solutions d'anti-tricherie basées sur les MMOGs à architecture C/S se basent sur l'analyse des comportements des joueurs, suivant l'hypothèse qu'un tricheur présente un comportement distinguable d'un joueur normal. Ces approches se basent essentiellement sur des types de tricheries comme le Wall-hacking et l'Aimbot. L'inconvénient de ces approches est que parfois les tricheurs se passent de l'utilisation des bots pour rendre leur comportement normal, et ainsi augmenter leur chance d'éviter d'être détectés par l'analyse de comportement. Les approches proposées dans [Laurens et al., 2007] et [Mitterhofer et al., 2009] sont des solutions à ces types de tricheries. Par exemple : dans les jeux de stratégie en temps réel, le Wall-hacking (crack de mur) est considéré comme de la tricherie car il permet aux joueurs (tricheurs) de voir des objets secrets situés derrière un mur qui influent sur les résultats du jeu.
- Les solutions d'anti-tricherie existantes pour les MMOGs à architecture P2P se basent sur au moins un des trois critères suivants :
    1. **La Cohérence et le consensus de jeu** : les solutions utilisant ce critère s'intéressent à l'état de jeu suivant le principe que chaque reproduction commençant avec les mêmes états initiaux et les mêmes événements de processus dans le même ordre finiront avec le même état pour tous les joueurs. En prenant en compte l'hypothèse que la majorité des joueurs se comportent bien, les joueurs qui envoient des mises à jour différentes que celles de la majorité sont forcément des tricheurs. Les approches [Kabus et al., 2005], [Kabus et Buchmann 2007] et [Liu et Tang, 2009] sont des solutions qui adoptent cette approche pour détecter la tricherie. Par exemple : Le tricheur provoque des problèmes de timing d'événements et des perturbations. Dans la figure IV.1, le joueur a modifié son jeu de telle sorte qu'il puisse ignorer un joueur ou un message reçu. Par exemple : s'il y a une équipe de 2 joueurs

honnêtes, le tricheur peut vouloir avoir un avantage sur eux en ignorant un joueur dans une situation désavantageuse. De cette façon, le tricheur est en mesure d'avoir le champ libre au cours de cet échange alors que les victimes n'ont pas de moyens disponibles pour se défendre. Dans la figure IV.1.b, le tricheur est représenter par la couleur rouge, ce dernier ignore tous les messages émis d'un joueur et sélectionne les messages à ignorer d'un autre joueur.



a) Echange Honnête b) Echange Triché

Figure IV. 1 : Un Echange entre 3 joueurs

2. Les superviseurs : les approches qui se basent sur les superviseurs (parfois appelés aussi data holders, coordinateurs, contrôleurs de région, …) assurent que ces superviseurs ne sont pas corrompus ou ne sont pas un objet d'attaque, car les superviseurs transmettent périodiquement les changements d'état de jeu à tous les joueurs de sa tutelle, les mises à jour aux clients et les sauvegardes au serveur de base de données. Par contre la manipulation de l'état local du jeu d'un joueur ordinaire est inutile, puisqu'il ne transmettra jamais son état à quelqu'un d'autre. Un joueur ordinaire ne peut communiquer directement avec un autre joueur que par le biais du superviseur, jamais directement. Les solutions proposées par [Norden et Guo, 2007] et [Daniel et Soh, 2008] adoptent cette approche.

3. Les signatures : Ces types d'approches se basent sur la signature d'événement pour signer d'une manière efficace les messages des événements discrets. Ils prennent en considération l'intégration du mécanisme d'authentification pour authentifier les joueurs connectés et s'assurer de leurs identités et aussi la réalisation de la non-répudiation des signatures digitales

pour que le client ne puisse pas nier un message qu'il a envoyé. La solution proposée par [Chan et al., 2008] est basée sur cette approche.

Les approches qui se basent sur les signatures ou sur les superviseurs peuvent compléter les approches qui se basent sur la cohérence. En effet, ces dernières nécessitent des signatures entre les superviseurs et les joueurs ainsi qu'une protection du superviseur pour éviter que les mises à jour envoyées par le superviseur aux joueurs ne soient fausses.

Notons que les solutions étudiées se focalisent sur un certain type de tricherie à savoir :

- Le Wall-hacking et l'Aimbot pour les MMOGs à architecture C/S et
- La collusion, l'exploitation du manque de confidentialité et d'authentification ainsi que la confiance mal placée pour les MMOGs à architecture P2P.

La proposition d'une solution anti-tricherie doit tenir compte des performances du jeu. En effet, un compromis entre la sécurité et la performance est nécessaire pour assurer que le jeu reste sécurisé et praticable. Pour cela, cinq facteurs qui caractérisent les performances d'un MMOG sont définit dans la littérature, à savoir :

- **Equilibrage de charge [Liu et Tang, 2009] :** Equilibrer le nombre de joueurs dans chaque zone afin de répartir équitablement les charges à chaque joueur.
- **Taux de transfert [Liu et Tang, 2009] :** Réduire le taux de transfert.
- **Adaptation [Liu et Tang, 2009] :** Le nombre de joueurs en ligne n'est pas fixe, de ce fait fournir la fonction d'adaptation (fusion/division) est nécessaire dans une division de l'univers de jeu. Le redimensionnement dynamique d'une zone pour s'adapter au nombre des joueurs est nécessaire.
- **Réactivité [Daniel et Soh, 2008] :** Le temps de réponse entre le superviseur et les clients doit être minimisé.
- **Equité [Daniel et Soh, 2008] :** L'écart entre les différents temps de réponse entre le superviseur et les clients doit être limité.

Afin de comparer les différentes solutions d'anti-tricherie proposées dans la littérature, nous avons réalisé un tableau comparatif (Tableau IV.1) qui récapitule les différentes solutions et les classifie selon les critères cités précédemment.

Approches anti- tricherie	C/S				P2P												
	Comportement	Etat du Jeu	Type Tricherie		Etat		Cohérence	Authentification	Type Tricherie				Performance				
			Wall-Hacking	Bot	jeu	Superviseur			Collusion	Exploitation de la confiance mal placée	Exploitation du manque d'authentification	Exploitation du manque confidentialité	Equilibrage de charge	Taux de transfert	Adaptation	Temps réponse	Equité
Laurens et al., 2007	X		X														
Yeung et Lui, 2008		X		X													
Mitterhofer et al., 2009	X			X													
Kabus et al., 2005					X		X	X	X	X	X						
Norden et Guo, 2007						X		X	X		X	X					
Kabus et Buchmann 2007					X		X	X	X	X	X	X					
Chan et al., 2008								X			X						
Daniel et Soh, 2008						X		X	X		X					X	X
Liu et Tang, 2009					X		X	X			X		X	X	X		

Tableau IV. 1 : Synthèse des solutions d'anti tricherie existantes

D'après notre étude comparative des approches existantes citées dans le tableau IV.1, nous avons conclu que DACA est la plus efficace car elle prend en compte les plus de facteurs de sécurité et aussi celle de performances du jeu tels que l'équilibrage de charge, le taux de transfert (faible) et l'adaptation qui sont négligés par les autres approches.

Nous allons à présent présenter en détaille l'approche DACA et étudier les critiques de cette dernière afin de pouvoir l'augmenter et améliorer ses performances.

IV.3. Approche DACA [Liu et Tang, 2009]

Nous allons étudier l'approche DACA suivant deux points de vue :

- **L'architecture et les fonctionnalités générales de l'approche.**
- **Le mécanisme d'anti-tricherie proposé (**Le mécanisme d'anti tricherie a été présenté dans la **section III.2.2.6.2).**

IV.3.1. Architecture et fonctionnalités générales du DACA

Elle est divisée en trois étapes essentielles :

- L'architecture du système
- La division de l'univers de jeu
- Fonctionnement du système

IV.3.1.1. Architecture du système

C'est une architecture hybride étant donné qu'une partie est gérée par un serveur central alors qu'une autre réside au niveau de certains nœuds P2P. Elle comporte trois couches :

1. **La couche la plus basse est la carte du jeu :** Elle est divisée en plusieurs grilles avec de tailles différentes, chaque grille contient un hotspot (point chaud, concept que nous avons défini dans I.2.3.4) dans le centre.
2. **La deuxième couche**, **couche d'action** : Les joueurs dans une grille forment un réseau de recouvrement de P2P pour interagir les uns avec les autres.

3. **La couche supérieure est la couche de contrôle :** Elle contient le *data holder* qui supervise le fonctionnement du groupe P2P auquel il est associé et trois types de serveurs : *authentification*, *jeu*, et *base de données*.

- **Le serveur d'authentification** : effectue principalement le contrôle de version, la connexion et l'authentification des joueurs.
- **Le serveur de jeu :** supervise le jeu global avec les responsabilités suivantes : diviser et fusionner dynamiquement des zones, joindre les calculs de jeu des zones suspectes, et de traiter les joueurs malveillants.
- **Serveur de base de données :** enregistre toutes les informations des joueurs, les personnages non-joueurs, et des objets dans l'univers de jeu. Le serveur de base de données a également pour rôle de se synchroniser avec le data holder de chaque zone afin de maintenir l'information de mise à jour.

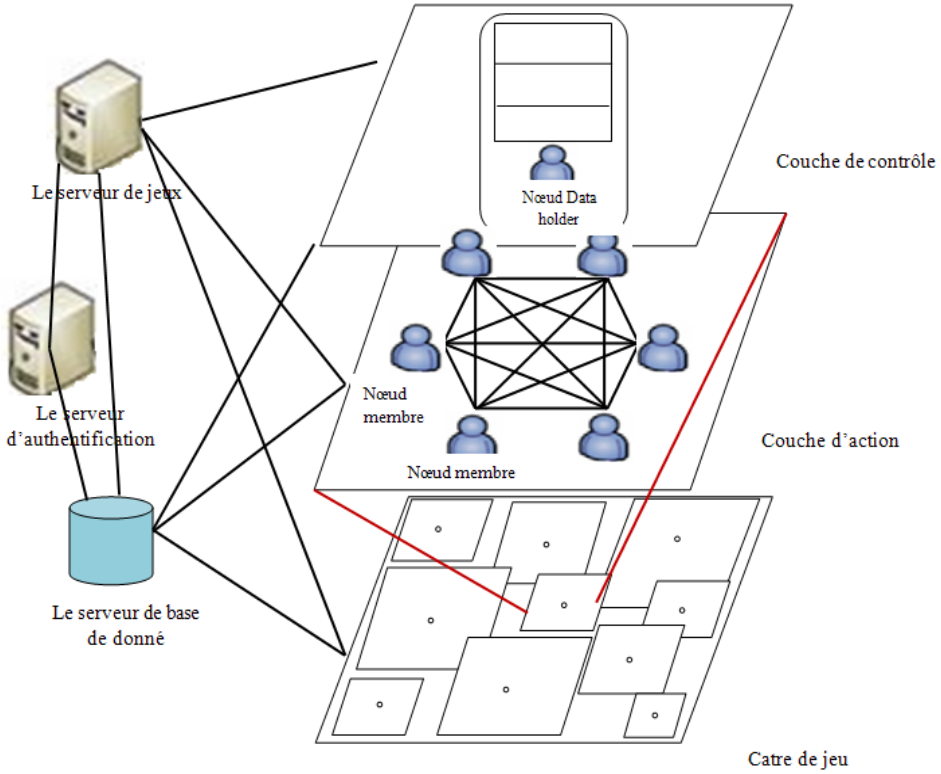


Figure IV. 2: Architecture du système DACA [Liu et Tang, 2009]

IV.3.1.2. La division d'univers de jeu

Une carte de jeu est divisée en une grille formée de zones, comme montré sur la figure IV.3.

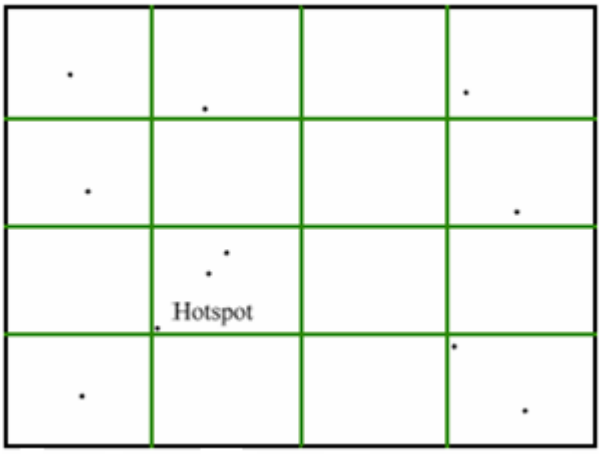


Figure IV. 3 : Exemple de la division de la grille [Liu et Tang, 2009]

La division d'univers de jeu a les caractéristiques suivantes :

- **Equilibrage de charge :** équilibrer le nombre de joueurs dans chaque zone afin de répartir équitablement les charges à chaque joueur.
- **Taux de transfert faible :** Puisque les joueurs massifs errent habituellement autour des hotspots, localiser un hotspot comme un centre d'une zone est fortement recommandé afin de réduire le taux de transfert.
- **Adaptation :** Le nombre de joueurs en ligne n'est pas fixe, de ce fait fournir la fonction d'adaptation (fusion/division) est nécessaire dans une division de l'univers de jeu. Le redimensionnement dynamique d'une zone pour s'adapter au nombre des joueurs est nécessaire.
- **Diversité :** Il est normal et pratique de donner un hotspot différent avec une popularité différente. Ainsi, la division uniforme de l'univers de jeu n'est pas pratique. Considérer la diversité de chaque hotspot est fortement recommandé.

DACA propose une division simple pour constituer un groupe P2P, appelée division de responsabilité pondéré WRD (*Weighted Responsible Division*). Cette dernière divise l'univers de jeu en un certain nombre de carrés redimensionnables avec le hotspot comme le point d'intersection des deux diagonales. La taille initiale de chaque carré est déterminée par la popularité du hotspot associé.

La région chargée est rattaché au plus petit carré. Puisque le découpage des carrés est très simplifié, il est très simple de redimensionner et déplacer.

Selon le changement dynamique de distributions des joueurs, la popularité et le centre d'un hotspot peuvent changer. Le serveur de jeu observe le changement de chaque hotspot et détermine s'il faut redimensionner la zone ou déplacer le centre d'un hotspot ou non. Après le redimensionnement ou le déplacement, le serveur de jeu informera les data holders correspondants sur la mise à jour des listes des membres.

Pour une zone lourdement chargé, WRD définit une limite supérieure pour la division et une limite inférieure pour la fusion sur le nombre de joueurs dans une zone. Si une zone avec des joueurs dépasse la limite supérieure, la procédure de division est lancée par le serveur de jeu.

La zone de grande charge est divisée en deux carrés concentriques comme montrée sur la Figure IV.4. Le carré intérieur est le carré principal et le carré extérieur est le carré secondaire. Le data holder principal gère le carré principal. Le serveur de jeu sélectionne au hasard un joueur comme un data holder pour le carré secondaire. Ces deux data holders mettront à jour les nouvelles listes de membre avec leurs membres.

D'autre part, si le nombre de joueurs dans une zone de division qui est défini comme une limite inférieur par le système, le serveur de jeu les fusionnera dans un carré. Le data holder du carré principal devient le data holder du carré fusionné.

Notons que le fusionnement est seulement produit dans les zones divisées de zones originales surfacées.

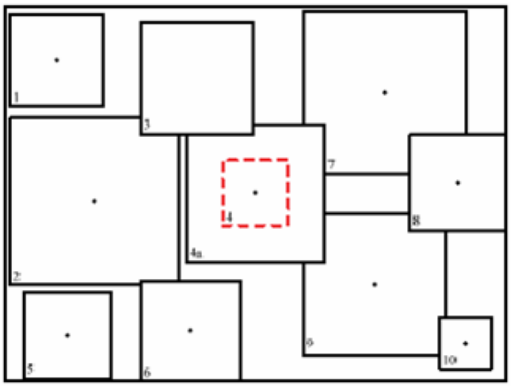


Figure IV. 4 : Exemple de redimensionnement dynamique [Liu et Tang, 2009]

L'ajustement individuel est facile à appliquer à WRD parce que WRD divise l'univers de jeu en plusieurs carrés individuels avec des tailles différentes basées sur la propriété du hotspot.

IV.3.1.3. Fonctionnement du système

Tous les joueurs dans une zone forment un groupe de P2P. Le data holder, attribué par DACA, fait partie d'un groupe de P2P. Le data holder d'une zone est aléatoirement choisi d'une autre zone. Les responsabilités du data holder incluent :

1. Enregistrement de la liste des membres.
2. Synchronisation avec le serveur de base de données pour obtenir les informations de sa zone, puis diffusion à ses membres.
3. Collection et vérification des informations de ses membres.
4. Mise à jour périodique de l'état de la zone avec le serveur de jeux. Tous les joueurs dans la même zone diffusent leurs actions à tous les autres membres, calculent toutes les actions et les transmettent au data holder. Le data holder collecte les informations de tous les membres, et les vérifie, et diffuse les résultats de la majorité de tous les membres.

Le fonctionnement du système repose sur les trois procédures suivantes : connexion, le jeu, et le transfert.

IV.3.1.3.1. Connexion/Déconnexion

Pour la connexion au système, le joueur se connecte d'abord au serveur d'authentification pour s'assurer de son identité, vérifier sa version du jeu, et mettre à jour si nécessaire. Le serveur d'authentification passera le joueur vers le serveur de jeu. Le serveur de jeu informera le joueur de sa zone (zone ID) et son data holder associé (IP ADDRESS) et informera également le data holder correspondant l'arrivé un nouveau joueur (IP ADDRESS). Le joueur s'enregistre au data holder de sa zone et obtient toute l'information de mise à jour y compris l'information de la zone et la liste des membres. Le data holder informera tous les autres joueurs du nouveau joueur rejoint.

Pour la déconnexion du système, le joueur informe simplement son data holder et se déconnecte. Le data holder synchronise la mise à jour de l'information avec le serveur de système en conséquence.

IV.3.1.3.2. Jeu

Le système définit cinq fréquences synchrones $f_{DB \rightarrow DH}$, $f_{DH \rightarrow DB}$, $f_{DH \rightarrow P}$, $f_{P \rightarrow DH}$ et $f_{P \rightarrow P}$, pour les joueurs, les data holders, et le serveur de base de données qui synchronise les uns avec les autres. Tous les data holders mettent à jour les états des joueurs de leur zone avec le serveur de base de données dans la fréquence $f_{DH \rightarrow DB}$.

Le serveur de base de données synchronise l'état de NPC (*Non-Player Character*) dans chaque zone et les adresses IP des data holders des zones voisins avec le data holder associé dans la fréquence $f_{DB \rightarrow DH}$.

Un joueur diffuse ses actions à tous les joueurs dans son groupe de P2P dans la fréquence $f_{P \rightarrow P}$.

Après avoir collecté les actions de tous les joueurs, un joueur calcule l'état de mis à jour pour chaque joueur dans sa zone et ensuite mettre à jour avec son data holder dans la fréquence $f_{P \rightarrow DH}$.

Après la collection des états de mise à jour de tous les joueurs dans sa zone, le data holder les vérifie. Si les états recueillies sont cohérents, le data holder diffusera l'état définitif de la mise à jour de chaque joueur à tous les joueurs dans la fréquence $f_{DH \rightarrow P}$. Toute incohérence révèle la tricherie dans ce domaine.

IV.3.1.3.3. Transfert

Cette procédure montre le déroulement du transfert entre les zones. Par exemple, si un joueur (disons k) se déplace vers une autre zone Y, il envoie d'abord une demande de transfert avec ses coordonnées actuelles à son data holder (data holder de la zone X) comme le montre la Figure IV.5.

Selon les coordonnées de joueur, le data holder de la zone X transmet une demande de transfert au data holder correspondant (data holder de la zone Y) et informe également ses membres l'abandon de joueur k.

Le data holder de la zone Y informe ses membres du groupe de la jointure de joueur k, puis envoie une réponse de transfert avec la liste des membres de la zone Y au joueur k.

Après l'obtention de la liste des membres de la nouvelle zone, le joueur k établit des liens P2P avec tous les d'autres joueurs du groupes et complète également la procédure de transfert. Le transfert est purement accompli par les joueurs eux-mêmes, donc la charge du serveur est très réduite.

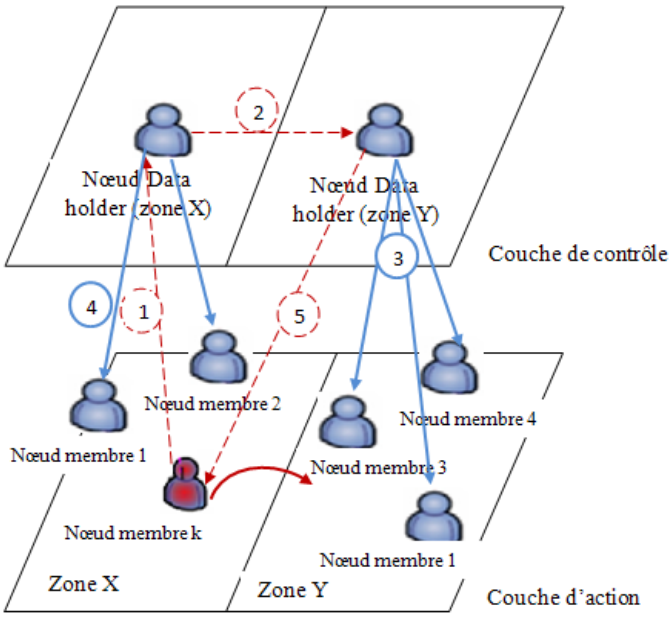


Figure IV. 5 : La procédure de transfert dans DACA

IV.3.2. Critiques du mécanisme d'anti-tricherie de DACA

Le mécanisme d'anti tricherie a été présenté dans la section III.2.2.6.2. Nous pouvons relever plusieurs insuffisances à ce mécanisme. Ces critiques concernent d'une part le système de détection et d'autre part, les performances de ce dernier.

- DACA ne prend pas en considération le temps de tricherie : dans le cas où le data holder fournit de fausses informations aux joueurs de cette zone et après la vérification qui sera menée par le data verifier, les informations fournies par le data holder seront

annulées en revenant en arrière. Le temps entre que le data holder a fourni ses information et le temps d'annulation de ses information est le temps de tricherie.

- Dans le fonctionnement du système DACA, quand un joueur veut jouer, il se connecte d'abord au serveur d'authentification pour s'identifier. Après le passage de l'étape l'authentification, le serveur d'authentification bascule le joueur vers le serveur de jeu. Le serveur de jeu informera le joueur de sa zone (zone ID) et de son data holder associé (IP ADDRESS), ce qui rend le data holder vulnérable aux attaques puisque son adresse IP est visible pour n'importe quelle joueurs.
- DACA ne prend pas en considération l'emplacement du data holder. Donc le data holder sélectionné peut être géographiquement éloigné par rapport à l'ensemble des joueurs de sa zone de contrôle. Ce qui affectera la qualité de service du jeu, vu que des durée du temps réponse du data holder vont être très longues.

IV.4. Proposition d'une solution d'anti-tricherie basée sur DACA : *DACA-Extended*

Notre approche "DACA-Extended" vise à améliorer l'approche « DACA » sur les points suivants : la minimisation du temps de tricherie, le problème de visibilité d'adresse IP et le choix du data holder.

La minimisation du temps de tricherie et le problème de visibilité d'adresse IP étant liés, nous allons donc les traiter dans la même partie.

IV.4.1. La minimisation du temps de tricherie et le problème de visibilité d'adresse IP

Le temps de tricherie est défini comme étant le temps durant lequel le data holder peut mener une tricherie ou bien être l'objet d'une tricherie (attaque) [Norden et Guo, 2007].

Les attaques possibles sur le data holder incluent [Norden et Guo, 2007] :

- L'attaquant induit le data holder à la tricherie : Pour amener le data holder à envoyer de fausses informations, un attaquant peut modifier les informations de ce dernier par l'installation d'un logiciel espion ou malveillant.
- L'attaquant peut carrément mettre le data holder hors service en lançant une attaque DoS.

La minimisation du temps de tricherie est réalisée en ajoutant la fonctionnalité du changement planifié du data holder [Norden et Guo, 2007]. Ceci consiste à changer périodiquement le data holder de telle manière que si ce dernier est cible d'une attaque, cette dernière sera brisée du moment que le data holder ne sera plus le même au bout d'un certain temps.

Dans DACA, la notion du changement planifié du data holder n'existe pas. Par conséquent, le data holder est statique et au bout d'un certain temps de recherche mené par l'attaquant, ce dernier peut l'induire à la tricherie en installant un logiciel espion ou malveillant. Par contre dans DACA-Extended avec le changement planifié du data holder, Le bénéfice est analogue à l'expression : « tirer sur une cible mobile », d'où la difficulté de trouver le data holder, donc le temps de tricherie est minimisé par rapport à l'approche DACA. Nous allons voir par la suite un exemple qui montre le temps gagné dans DACA-Extended par rapport à DACA (voir l'exemple IV.4.1.1).

Etant donné qu'à tout moment, le data holder est vulnérable, la technique de changement planifié du data holder a pour objectif de minimiser le temps de tricherie en changeant le data holder actif par un data holder candidat et propager l'information à travers les différents clients du jeu (joueurs) [Norden et Guo, 2007]. Ainsi, le temps durant lequel le data holder peut tricher est limité entre le début de la tricherie et l'instant où la fonctionnalité est déplacée.

Le changement du data holder survient dans le cas où le data verifier détecte que le data holder ne remplit plus son rôle ou lors d'un changement planifié (après un $t_{déplacement}$, voir IV.4.1.1) ; ce changement concerne toutes les fonctionnalités du data holder. Une mise à jour du nouveau data holder au niveau de tous les joueurs est effectuée.

Le temps de déplacement "$t_{déplacement}$" tel que proposé dans [Norden et Guo, 2007] est constant. Les événements de changement se produisent donc d'une manière *régulière* et *déterministe*. Par conséquent, un attaquant peut prévoir quand agir du moment qu'il peut prévoir quand il y a changement du data holder. Ceci nous amène à améliorer ce mécanisme en proposant que le choix du $t_{déplacement}$ se fait d'une manière *aléatoire* pour chaque changement du data holder. Nous verrons dans la partie IV.4.3 la réalisation de ce processus.

Pour réaliser ce mécanisme de changement planifié, nous introduisons avec DACA-Extended un nouveau type de nœud appelé *Data Holder Master* (DHM) qui s'occupe entre autres de la sélection d'un data holder actif à partir d'un ensemble de data holders candidats (EDH) [Norden et Guo, 2007]. Les clients envoient tous les messages de mise à jour au DHM et au data holder. Le DHM joue le rôle d'intermédiaire de confiance entre les clients et les data holders, et permet aussi la communication client/data holder.

Dans [Norden et Guo, 2007], le DHM est statique et est le seul gestionnaire des data holders et la solution n'envisage pas de plan de secours en cas de panne, entrainant une défaillance dans le fonctionnement du système. Pour pallier à ce problème, nous proposons une sélection aléatoire du DHM, réalisée par le serveur ; de plus, le choix du prochain DHM est réalisée d'une manière aléatoire afin que l'attaquant ne puisse pas prévoir l'identité du prochain DHM (voir la partie IV.4.3). Le mécanisme de sélection de l'ensemble des data holder (EDH) qui contient le data holder actif et les data holders candidats est abordé dans la partie IV.4.2.1.

Les deux figures IV.6 et IV.7 montrent l'architecture du data holder dans DACA et DACA-Extended respectivement.

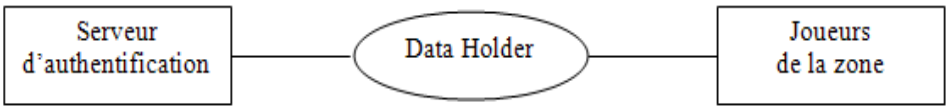


Figure IV. 6 : Architecture du Data holder dans DACA

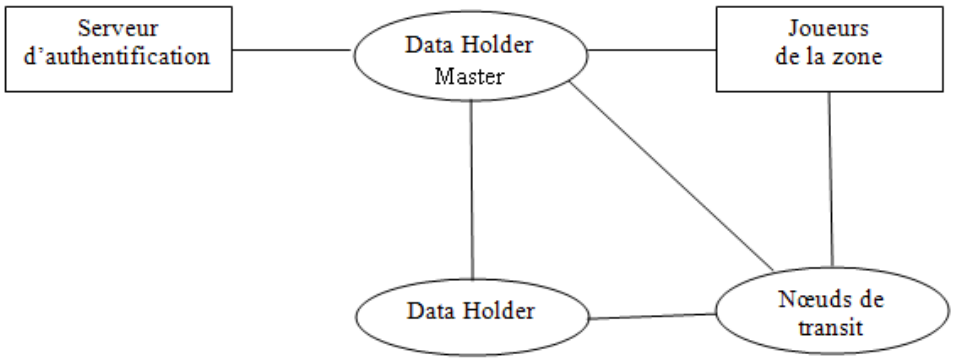


Figure IV. 7 : Architecture du Data holder dans DACA-Extended

IV.4.1.1. Exemple illustratif

Pour illustrer les avantages de DACA-Extended. On introduit les notions suivantes :

$\mathbf{t_{cherche}}$: Le temps qu'il faut pour un attaquant de trouver un nouveau data holder**.** Le nouveau data holder est le data holder candidat (à chaque changement du data holder l'attaquant doit chercher l'identité du nouveau data holder).

$\mathbf{t_{installe}}$: Le temps qu'il faut à un attaquant pour installer un logiciel espion ou malveillant et activer une tricherie (au niveau du data holder).

$\mathbf{t_{déplacement}}$: L'intervalle de temps entre deux changements consécutifs du data holder.

Le scénario suivant permet d'illustrer le mécanisme de changement du data holder qui va pousser l'attaquant à refaire une recherche pour trouver le nouveau data holder. Le gain de temps de tricherie est illustré sur la figure IV.8. Le scénario se déroule comme suit :

- ✓ A l'instant $\mathbf{t_0}$: l'attaquant commence à rechercher le data holder courant **DH.**
- ✓ A l'instant $\mathbf{t_1}$: après $\mathbf{t_{cherche}}$, l'attaquant commence à installer des logiciels espions ou malveillants sur **DH**.
- ✓ A l'instant $\mathbf{t_2}$: Après $\mathbf{t_{installe}}$, **DH** est compromis et peut commencer à tricher.
- ✓ Dans le cas statique : DH continue à tricher tout au long du jeu.
- ✓ Dans le cas dynamique : puisque le changement est planifié, $\mathbf{D_H}$ change et devient $\mathbf{DH_N}$ (le nouveau data holder) à l'instant $\mathbf{t_3}$.

- ✓ Après le $t_{cherche}$ + $t_{installe}$ de ce déplacement, à l'instant t5, le nouveau Data holder DH_N peut être compromis et induit à la tricherie.

Comme nous pouvons le constater, chaque changement du data holder permet d'économiser $t_{cherche}$+$t_{installe}$ de temps de tricherie.

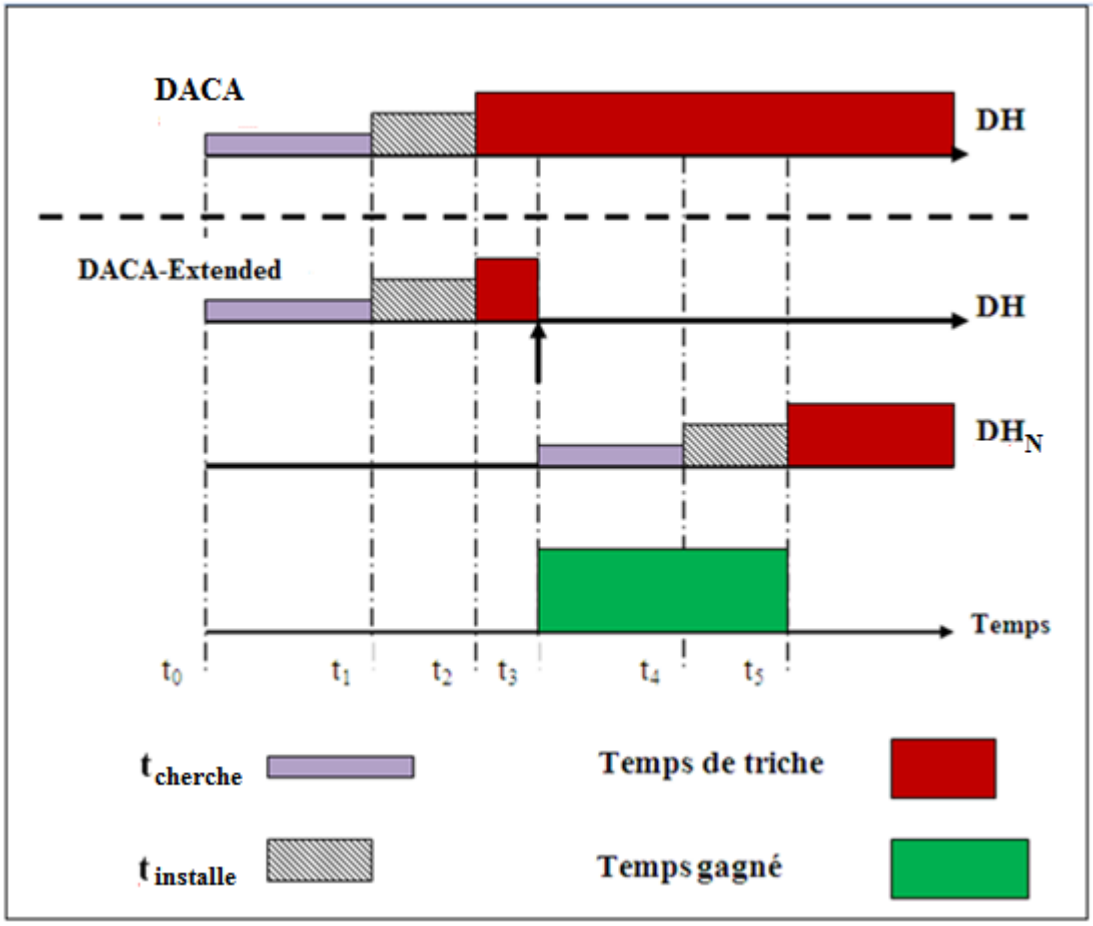


Figure IV. 8 : Scénario du changement du data holder

Visibilité de l'adresse IP

Dans notre solution DACA-Extended, le DHM gère en même temps plusieurs zones, chaque zone est constitué de plusieurs data holder. Chaque data holder contrôle les clients qui lui sont attribués.

Notre solution se base essentiellement sur une structure de donnée centrale qui est la table de hachage MMT [Norden et Guo, 2007], construite par le DHM et envoyé par ce dernier à tous les clients. Elle possède deux colonnes. Chaque entrée désigne un ensemble de

séquence d'ID et l'entrée correspondante du côté droit est la clé à laquelle ce dernier ensemble de séquence d'ID est mappé, comme illustré sur la figure IV.9. La clé est utilisée par le mécanisme standard de routage de P2P [Stoica et al., 2001, Rowstron et Druschel, 2001, Zhao et al., 2004] pour transférer les messages. Ainsi, en utilisant la table MMT de correspondance (ID-clé) les nœuds de transit effectuent le routage par l'utilisation des algorithmes de routage (par exemple le routage CHORD) [Stoica et al., 2001], pour mapper éventuellement au final la clé à une adresse IP.

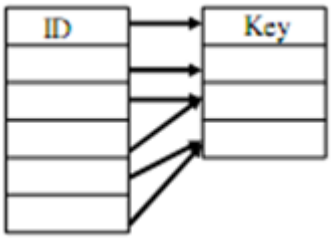


Figure IV. 9 : La table MMT

Le *seed* est utilisé par chaque client pour générer une séquence d'ID au fil du temps, en utilisant un mécanisme de *hash-chain* [Yang et al., 2004] analogue au mécanisme *RSA token-based SecureID*. Le mécanisme *hash-chain* est aussi appelée fonction de hachage à sens unique ou "*one-way hash chain*" en anglais. Ce type de fonction est très utilisé en cryptographie, principalement dans le but de réduire la taille des données à traiter par la fonction de cryptage [Yen et Zheng, 2000]. En effet, la caractéristique principale d'une fonction de hachage est de produire un haché (dans notre cas c'est l'ID) des données, c'est-à-dire un condensé de ces données. Ce condensé est de taille fixe, dont la valeur diffère suivant la fonction utilisée.

Le salage (*salting* en anglais) consiste à ajouter un secret partagé (*seed*) – généralement une chaîne de caractères – à l'information avant le hachage. Par exemple, dans un cadre cryptographique, au lieu de pratiquer le hachage uniquement sur le mot de passe, on peut le faire sur le résultat de la concaténation du mot de passe avec une autre chaîne de caractères pseudo-aléatoire, obtenue par un hachage de l'identifiant (*login*) concaténé avec le mot de passe.

Le mappage plusieurs-à-un entre le couple (numéros de séquence ID et les clés) de la MMT, combiné avec le numéro de séquence ID courant généré par le client déterminent de façon unique la clé correspondante au data holder actuel. Comme les IDs changent avec le temps, le

data holder change à l'intérieur de l'ensemble des data holders candidats périodiquement. Les clients ne connaissent pas l'adresse IP du data holder actuel, ils utilisent simplement la clé dérivée pour lui transmettre des messages en utilisant le mécanisme de routage du P2P [Stoica et al., 2001, Rowstron et Druschel, 2001, Zhao et al., 2004]. L'avantage de ce mécanisme est que, en dehors du *seed* initial et de la MMT qui est communiqués entre le DHM et un client donné, la séquence est générée d'une manière autonome (ne nécessitant aucune autre interaction entre les deux entités). De manière formelle:

$$ID_0 = h_0 = seed$$
$$ID_1 = h_1 = h(h_0) = h(seed)$$
$$ID_2 = h_2 = h(h_1) = h(h(seed))$$
$$ID_3 = h_3 = h(h_2) = h(h(h(seed)))$$
$$\vdots$$
$$ID_n = h_n = h(h_{n-1}) = h(h(h(..(seed)..))) \qquad \textbf{Équation IV. 1}$$

Où h est la fonction de hachage utilisée (comme par exemple MD5 ou SHA). La séquence ID généré par chaque client est donc : ID_0, $ID_1, ID_2, ID_3, \ldots ID_n$.

La clé présentée dans MMT est utilisé par les nœuds de transit qui utilisent l'algorithme de routage pour éventuellement mapper à l'adresse correspondante. Nous définissons et utilisons la structure de données la Table «Master Key Table» (**MKT**) [Norden et Guo, 2007] qui représente la décision de routage à partir d'une clé vers une adresse IP, voir la figure IV.10. Chaque client ou nœud de transit ne possède que l'adresse IP du prochain nœud voisin à partir de la clé et pas l'adresse IP du data holder actuel.

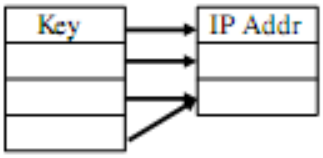


Figure IV. 10 : La table MKT

Avec l'algorithme de routage P2P [Stoica et al., 2001, Rowstron et Druschel, 2001, Zhao et al., 2004], un mappage de plusieurs-à-un a été conçu entre la clé et chaque data holder de l'ensemble EDH. Ceci consiste à assigner des IDs de nœud à tous les nœuds dans le réseau de recouvrement P2P, y compris les data holders et les nœuds de transit, donc les messages avec des clés multiples sont routés à un même ID nœud. Cette relation plusieurs-à-un rend

plus difficile pour une entité malveillante de connaître le nombre de data holders actuellement présents dans le réseau à partir de la MMT. La figure IV.11 montre la Structure de la MMT et la MKT et le mappage du numéro de séquence ID vers l'adresse IP du data holder.

Puisqu'aucun nœud P2P ne maintient la table MKT complète, une entité malveillante doit compromettre tous les nœuds de transit P2P sur le chemin vers le data holder pour pouvoir trouver le data holder actuel. Même dans ce cas, compromettre tout ces nœuds de transit sera inutile puisque le data holder aura changé au moment où l'attaquant pourra lancer une attaque directe.

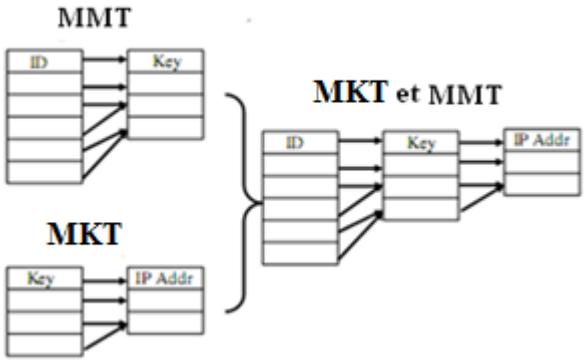


Figure IV. 11 : La Structure de la MMT et la MKT et le mappage du numéro de séquence ID vers l'adresse IP du data holder [Norden et Guo, 2007]

Pour assurer la communication client/data holder, le DHM suit les étapes suivantes :

1. DHM construit une table d'hachage Master Mapping Table **(MMT)** et l'envoie à tous les clients (joueur) par multicast.
2. DHM génère un secret partagé **(seed)** et l'envoie a tous les clients d'une manière sécurisé par multicast.
3. le DHM remplit le réseau P2P (spécifiquement les nœuds de transit) avec le mappage plusieurs-à-un stocké dans la table Master Key Table **(MKT)** entre les clés et les adresses IP des data holders dans l'ensemble des candidats.

Donc le problème de visibilité d'adresse IP est réglé. De plus il est très difficile de trouver le data holder puisque son adresse IP est invisible et même dans le cas où on peut trouver le data holder son changement à chaque fois rend sa détection presque impossible.

Ces deux diagrammes montrent les deux cas d'assignement du data holder dans: DACA et DACA-Extended.

Le cas DACA :

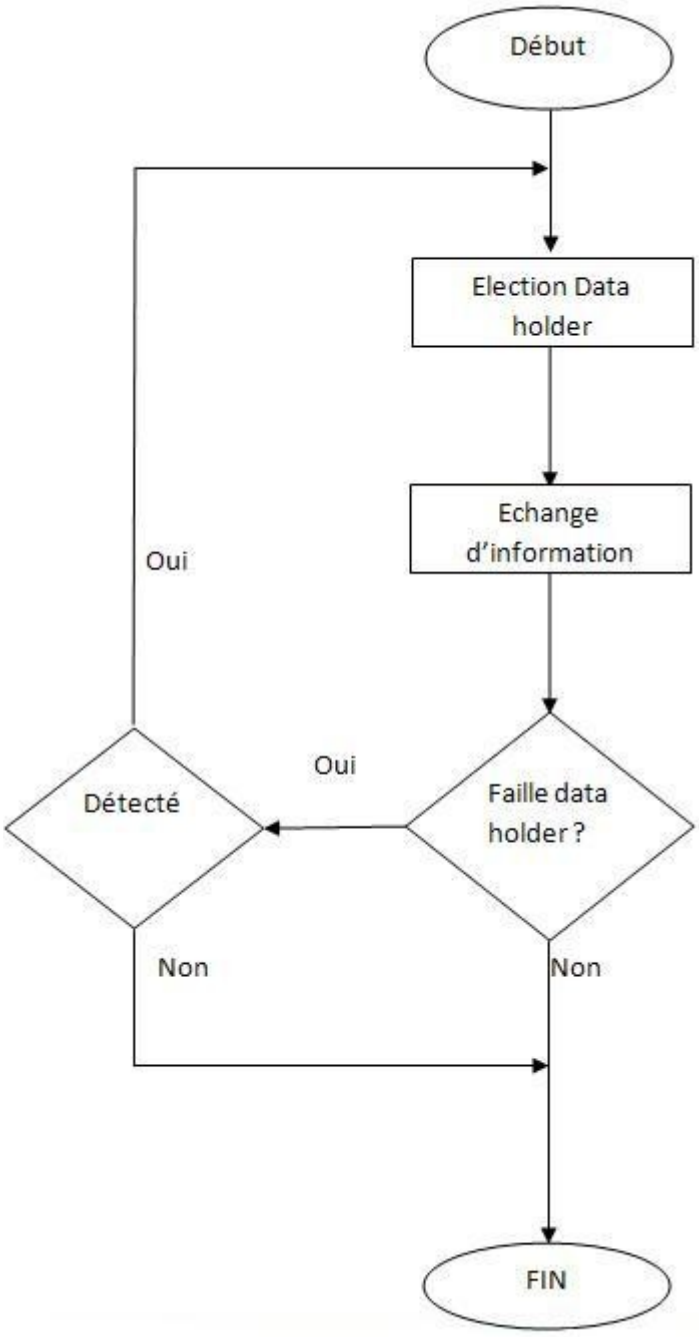


Figure IV. 12 : Organigramme d'assignement du data holder dans DACA

Le cas DACA-Extended :

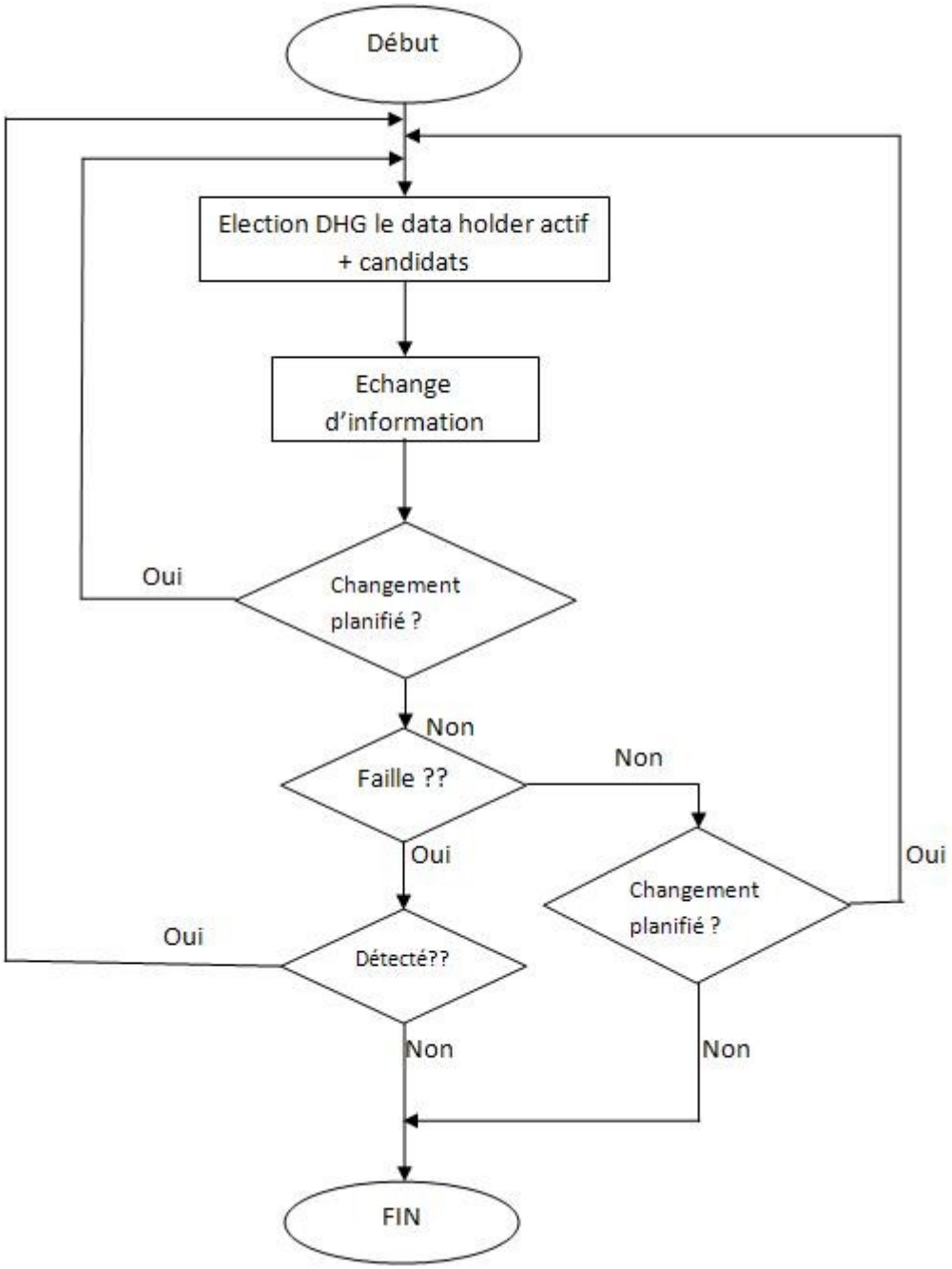


Figure IV. 13 : Organigramme d'assignement du data holder dans DACA-Extended

IV.4.1.2. Transfert dans DACA-Extended

Dans DACA-Extended, le transfert des joueurs dans les zones est réalisé d'une manière différente du DACA, car le transfert dans l'approche DACA révèle l'adresse IP du data holder aux joueurs [Liu et Tang, 2009].

L'exemple suivant montre comment le transfert se déroule dans DACA-Extended relativement au scénario classique de DACA (figure IV.14). Si un joueur (disons k) de la zone X veut se déplacer vers une autre zone Y, l'ensemble des étapes suivantes doit être effectué (ces étapes sont illustrées sur la figure) :

1. le joueur envoie d'abord une demande de transfert avec ses coordonnées actuelles à son data holder (celui de la zone X) en utilisant la clé dérivée pour transmettre le message.
2. Le data holder de la zone X transmet une demande de transfert au DHM en joignant les coordonnées actuelles du joueur et en indiquant la zone de transfert Y.
3. Le data holder de la zone X informe également ses membres le fait que le joueur k quitte la zone.
4. Le DHM transmet la demande de transfert au data holder correspondant (data holder de la zone Y).
5. Le DHM envoie un Seed au joueur K.
6. Le data holder de la zone Y informe ses membres du groupe de la jointure de joueur k.
7. Le data holder de la zone Y envoie une réponse de transfert avec la liste des membres de la zone Y au joueur k.

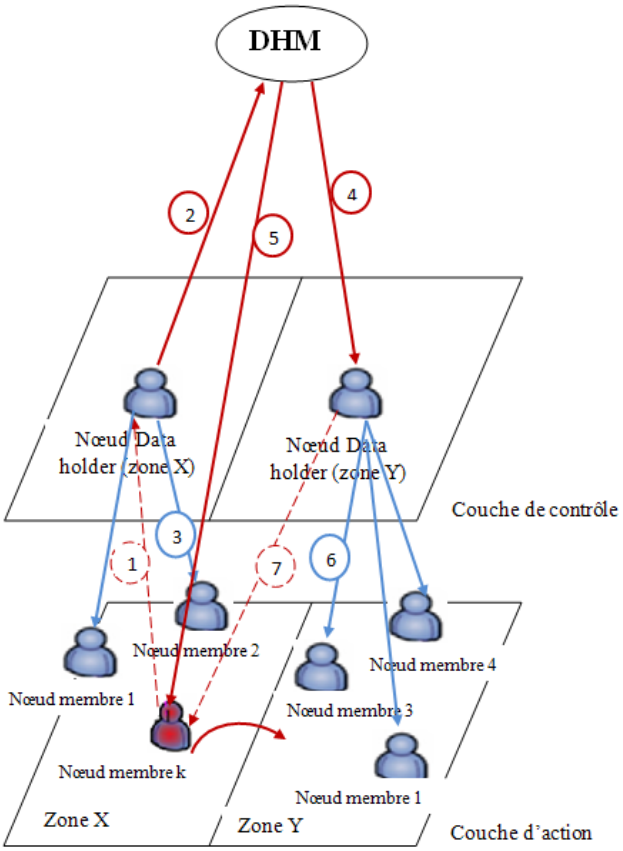


Figure IV. 14 : La procédure de transfert dans DACA-Extended

IV.4.2. Le choix du data holder

Le serveur d'authentification (SA) est responsable de la vérification et la validation de l'identité des joueurs qui joignent le jeu ainsi que de la sélection des pairs qui seront élus data holders. Ces derniers sont choisis de telle manière qu'ils disposent de ressources suffisantes (capacité de calcul "*CPU*", bande passante, capacité de stockage) pour leur permettre de recevoir et de traiter un nombre important de requêtes reçues simultanément. Pour cela les ressources de chaque pair doivent être transmises au serveur d'authentification afin de permettre la sélection des data holders.

Le data holder contrôle toutes les informations requises d'une zone. Pour une meilleure prévention, un pair ne peut être élu data holder dans la zone où figure son avatar. Si l'avatar d'un data holder se déplace dans une zone qu'il contrôle, alors le serveur d'authentification choisira un nouvel data holder comme remplaçant.

Aussi, la sélection du data holder doit assurer la cohérence temporelle dans les réponses des joueurs au data holder. Cela signifie que l'écart entre les différents temps de réponse doit être limité de telle sorte que tous les joueurs aient l'accès au jeu d'une manière équitable. Ainsi la durée des temps de réponse des joueurs doit être minimisée et donc la réactivité est améliorée.

Pour assurer la cohérence temporelle et la minimisation du temps de réponse, notre sélection des data holders doit prendre en considération la localisation géographique de ces derniers. Ainsi, la qualité de service ne sera pas affectée par la distance d'éloignement d'un data holder par rapport à sa zone de contrôle.

Pour répondre à l'ensemble des exigences de la sélection du data holder, nous nous sommes inspirés de la politique de l'approche "Secure Referee Selection" [Daniel et Soh, 2008]. Par conséquent, le choix du data holder doit répondre aux critères suivants :

1) **La réactivité de jeu :** Le temps de réponse moyen des pairs à data holder doit être minimisé. Il est calculé comme étant la somme des temps de réponse de pairs sur le nombre des pairs, comme donné par la formule suivante :

$$\overline{(\mathbf{TR_{P_i,DH}})} = (\sum_{\mathbf{P_i \in Z}} \mathbf{TR_{P_i,DH}})/|\mathbf{Z_P}| \qquad \textbf{Équation IV.2}$$

Avec : P_i est un pair dans la zone Z, DH est le data holder sélectionné, $TR_{P_i,DH}$ est le temps de réponse entre le DH et P_i et $|Z_P|$ est le nombre de pair dans la zone Z.

Pour adresser ce critère, nous devons savoir tous les temps de réponse de pair-à-pair. Pour cela, nous proposons l'utilisation des coordonnées du réseau [Dabek et al., 2004] pour estimer le temps de réponse de pair-à-pair. Lors de la connexion d'un pair, ce dernier calcule ses propres coordonnées de réseau ainsi que ses ressources, comme déjà indiqué, et les transmet au serveur d'authentification. Plusieurs méthodes peuvent être utilisées pour le calcul de ces systèmes de coordonnées. Par exemple, l'algorithme de *Vivaldi* [Dabek et al., 2004] permet le calcul de tels systèmes et l'association à chaque nœud d'un réseau deux coordonnées dans un espace euclidien, plus une hauteur. Ces coordonnées fournissent une bonne évaluation de temps de réponse entre deux pairs quelconques, même si aucune mesure directe entre ces deux pairs n'a été faite. Le temps de réponse n'est calculé qu'entre le data

holder et les joueurs (et non pas entre tous les pairs de nœuds), permettant d'économiser la bande passante nécessaire.

Pour minimiser le temps de réponse moyen, le data holder doit être choisi de telle manière qu'il soit proche de la majorité des joueurs. Nous utilisons le concept de *major* défini comme étant le point dans l'espace de coordonnées le plus proche de la majorité de joueurs de la zone [Daniel et Soh, 2008]. Nous sélectionnons comme data holder les pairs qui sont plus proches de la coordonnée major. Ainsi, la sélection des data holders proches du major adresses à ce critère.

2) **Assurer l'équité :** la différence, Δ, entre le temps de réponse maximal et minimal de joueur au data holder pour tous les pairs dans Z_P est minimisé.
   Dans ce critère nous prenons en considération les écarts des temps de réponse entre les pairs et le data holder, car si un temps de réponse entre un data holder et un des pairs est très petit, ils peuvent s'échanger plusieurs messages et mises à jour avant que le joueur avec temps de réponse élevé ne puisse recevoir sa première mise à jour, ce qui n'est pas juste. Pour diminuer l'écart de temps de réponse, le data holder peut envoyer des mises à jour en retard à un pair avec un temps de réponse petit de sorte qu'il reçoive des mises à jour en même temps qu'un pair avec un temps de réponse grand, donnant artificiellement ces pairs le même temps de réponse.

Equilibrage entre la réactivité de jeu et l'équité :

Pour minimiser le temps de réponse tout en assurant l'équité, nous proposons un mécanisme d'équilibrage qui combine entre ces deux critères en tenant compte des deux cas extrêmes suivants :

- Si nous augmentons les temps de réponse de tous les pairs à celui du pair le plus lent, la réactivité du jeu se trouve compromise (temps de réponse maximal pour tous les pairs).
- Si nous laissons tous les temps de réponse tels qu'ils sont, le jeu devient analogue à ceux commerciaux courants qui essayent de fournir le meilleur service possible à chaque joueur, en ignorant l'équité.

Dans notre approche DACA-Extended, la solution pour équilibrer entre ces deux critères est d'ignorer les pairs avec un temps de réponse très élevé par rapport à l'ensemble (c.-à-d. qu'ils sont éloignés du data holder et de l'ensemble des pairs, car la majorité des pairs auront des temps de réponse plus ou moins proches).

Pour illustrer ce mécanisme, considérons le temps de réponse (en ms) pour les deux cas suivants avec :

Δ : est la différence entre le temps de réponse maximal (*Max* $\boldsymbol{d_{i,j}}$) et minimal (*Min* $\mathbf{d_{i,j}}$) de joueur au data holder pour tous les pairs.

$\mathbf{d_{i,j}}$ **:** est le temps de réponse d'un joueur P_i à un data holder DH_j

- $d_{1,A}$=10, $d_{2,A}$=20, $d_{3,A}$=10, $d_{4,A}$=100 ; les résultats sont $\overline{(\mathbf{TR_{P_1,A}})} = \mathbf{35}$ et $\Delta = \mathbf{90}$
- $d_{1,B}$=40, $d_{2,B}$=60, $d_{3,B}$=40, $d_{4,B}$=60 ; les résultats sont $\overline{(\mathbf{TR_{P_1,B}})} = \mathbf{50}$ et $\Delta = \mathbf{20}$

Dans le cas du data holder A : le temps de réponse moyen est minimisé ce qui répond au critère 1, mais l'écart des temps de réponse Δ est élevé, ce qui ne satisfait pas le critère 2.

Dans le cas du data holder B : le temps de réponse moyen est grand ce qui mauvais par rapport au critère 1 mais l'écart entre les temps de réponse est minimisé ce qui répond au critère 2.

Dans le cas du data holder A, si nous augmentons les temps de réponse des pairs à celui du pair le plus lent donc à 100, on aura un temps de réponse très élevé, donc nous compromettrons l'objectif de la réactivité.

Nous proposons d'ignorer le *pair P_4* avec $d_{4,A}$=100 qui a un temps de réponse très élevé (le pair est éloigné de l'ensemble des pairs), et de refaire les calculs à nouveau. On aura comme résultat $\overline{(\mathbf{TR_{P_1,A}})} = \mathbf{13}$ et $\Delta = \mathbf{10.}$ Dans le nouveau cas le temps de réponse répond au critère 1, même l'augmentation du temps de réponse des pairs à celui du pair le plus lent n'affecte pas le critère 1 et assure aussi l'équité donc il répond au critère 2.

Concernant l'équilibrage entre la réactivité et l'équité : *équilibrer le temps de réponse pour chaque pair P_i,* la procédure contient deux étapes : la première étape est d'ignorer les

pairs qui ont un temps de réponse très élevé et la deuxième étape est d'augmenter artificiellement les temps de réponse des pairs au celui qui a un temps de réponse maximal.

- Pour la première étape, nous avons deux cas : le cas où Δ est inférieur ou égal au temps de réponse moyen et le cas où Δ est plus grand que ce temps de réponse moyen. Dans le deuxième cas, nous proposons d'ignorer les pairs avec des temps de réponse très élevé et refaire les calculs à nouveau. Cette étape est itérative jusqu'à ce que Δ soit inférieur ou égale au temps de réponse moyen.
- Pour la deuxième étape, l'augmentation des temps de réponse se fait comme suit : pour chaque pair P_k, nous calculons $\boldsymbol{RET_k}$ qui est la différence entre le temps de réponse maximal et celui du joueur P_k, comme donné par la formule suivante :

$$\boldsymbol{RET_k = Max\ d_{i,j} - d_{k,j}} \qquad \textbf{Équation IV.3}$$

Par la suite, le data holder envoie les mises à jour au pair P_k avec un retard qui est égale à RET_k.

IV.4.2.1. Algorithme de sélection du data holder

Nous allons maintenant expliquer et écrire l'algorithme de sélection du data holder. Cet algorithme permet de sélectionner un ensemble de data holder (EDH) qui contient le data holder actif et les data holders candidats. Pour cela nous allons prendre en considération dans le choix des data holders les deux critères réactivité et équité.

Notre algorithme est structuré comme suit :

Pour chaque zone Z :

1. sélectionner un ensemble de pairs (pour être des data holders) dans la zone Z dont les avatars jouent dans une zone autre que la zone Z. Ils doivent tout d'abord être validés par le serveur d'authentification après avoir estimé leurs ressources (capacité de calcul, bande passante, capacité de stockage) à partir des données reçues de ces pairs.
2. à partir de cet ensemble, sélectionner les pairs (data holders) qui sont les plus proches de la majorité des joueurs (être prés de *major*, voir Tableau IV.2.), et cela pour garantir un temps de réponse minimal. Le nombre des pairs (data holder actif et data holders

candidats) pourrait être statique (défini par l'administrateur) ou dynamique (selon le nombre de joueurs par zone).

3. *équilibrer le temps de réponse pour chaque pair* P_i pour que ces derniers aient un temps de réponse égal, comme discuté dans la procédure d'équilibrage entre la réactivité de jeu et l'équité. Quand de nouveaux joueurs joignent la zone, un calcul de leurs temps de réponse détermine l'action à entreprendre. Nous avons deux cas :

    - *Si le temps de réponse des nouveaux joueurs est inférieur ou égal au temps de réponse maximal*, les nouveaux joueurs joignant n'affectent pas les critères de jeu (réactivité et équité). Pour cela, nous équilibrons le temps de réponse pour chacun de ces joueur, comme décrit dans le troisième point de l'algorithme de sélection du data holder.
    - *Sinon*, une nouvelle sélection du data holder est nécessaire afin de préserver les critères de jeu. Par la suite, étant donné le changement du data holder, le temps de réponse de chaque joueur (ancien ou nouveau) doit être équilibré conformément à l'algorithme décrit.

Le tableau suivant décrit les différentes entités et fonctions utilisées dans l'algorithme de sélection du data holder et l'algorithme généralisé.

Entité	Description
Z	La zone Z
SELECT (Z)	• Le serveur d'authentification sélectionne l'ensemble des pairs de la zone Z. dont les avatars jouent dans une zone autre que la zone Z. • Ils doivent aussi avoir les ressources suffisantes pour être des data holders.
SPC	L'ensemble de pairs sélectionné par le serveur d'authentification.
EDH	Ensemble du Data holder qui contient le data holder actif et les data holders candidats.
Major (Z)	Un point dans l'espace de coordonnées (x,y) qui est le plus proche de la majorité de joueurs de la zone Z.
DH_i	Data holder
NDH	Nombre des data holders. Il peut être statique, défini par l'administrateur, ou dynamique, calculé selon le nombre de joueurs par zone.
PROCHE (SPC, (x,y))	Trouver le pair le plus proche des coordonnées (x,y).
EQUILIBRER (Z)	Augmenter artificiellement le temps de réponse des pairs pour que ces derniers aient un temps de réponse égal après avoir assuré la réactivité en ignorons les pairs avec

	un temps de réponse élevé. Voir la procédure d'équilibrage déjà présentée.
H	Nombre de zones gérées par le Data Holder Master
DH_{actif}	Data holder actif
ENVOI (seed, MMT, MKT partielle)	DHM envoie le seed et les tables MMT, MKT partielle à tous les clients en utilisant le multicast
ENVOI (message du changement)	le DHM envoie aux clients un message pour changer le data holder actif après un $t_{déplacement}$ qui est choisi aléatoirement pour chaque changement

Tableau IV. 2 : Glossaire des algorithmes

Algorithme de sélection du data holder pour chaque zone Z.

SPC= SELECT (Z) ; // un ensemble de pairs de la zone Z avec des ressources suffisantes et dont les avatars jouent dans une zone autre que Z.
NDH : integer ; // Nombre défini du Data holder dans la zone Z.
i=1;
EDH= Ø;
(x,y) = major (Z);
While i<= NDH do
DH_i= PROCHE (SPC, (x,y)) ; // Trouver le pair le plus proche des coordonnées (x,y).
EDH= EDH U DH_i;
SPC= SPC – DH_i;
i= i + 1 ;
done;
EQUILIBRER (Z) ; // Augmenter artificiellement le temps de réponse des pairs pour que ces derniers aient un temps de réponse égal sans compromettre la réactivité.

IV.4.3. Algorithme généralisé

Dans cette partie nous illustrons le fonctionnement global de DACA-Extended à savoir : la sélection du Data holder Master, la sélection de l'ensemble des data holders, la gestion du DHM, la procédure d'envoi du DHM et la partie client.

1. Le serveur d'authentification (SA) sélectionne aléatoirement le DHM à partir d'un ensemble de nœuds du réseau. Le SA peut changer à tout moment le DHM (en cas de panne, de déconnexion ou de surcharge de transfert) et sélectionner aléatoirement un nouveau DHM.
2. Le serveur d'authentification sélectionne l'ensemble des data holders (EDH) en utilisant l'algorithme de sélection du data holder que nous avons décrit dans la partie IV.4.2.1.

3. Le serveur d'authentification envoie au DHM l'ensemble des data holder (EDH). Si le canal de communication entre le serveur d'authentification et le DHM n'est pas protégé contre ce genre d'attaques comme le *middle attack* (il existe dans la littérature plusieurs mécanisme de protection), l'attaque reste toujours possible. Mais compte déjà sue cette faiblesse, nous supposons que le mécanisme proposé utilise des canaux de communications sécurisés.
4. Le DHM s'occupe du choix du data holder actif, du choix du prochain data holder, du choix du temps de changement ($t_{déplacement}$), et de la génération de la table MMT, la table MKT ainsi que le seed.

- Concernant la sélection du data holder actif et le prochain data holder actif, il existe plusieurs techniques telles que fifo, lifo et round robin [Pinedo, 2001].

    a. Par exemple round robin fonctionne de la manière suivante : après la sélection du data holder actif qui est le 1^{er} du tableau, il fait un décalage à gauche dans le tableau (le 1^{er} élément du tableau sera le dernier élément et le 2^{eme} sera le 1^{er}, et ainsi de suite) ; donc après le temps du changement, le 2^{eme} élément du tableau deviendra le 1^{er} et donc sera le nouveau data holder.
    b. Mais avec ces techniques, le choix du data holder est toujours prévisible et déterministe. Or, le choix du data holder actif et le choix du prochain data holder doit être aléatoire.
    c. Ceci nous amène à proposer la procédure suivante :

        i. nous choisissons d'abord un data holder au hasard de l'ensemble EDH qui sera élu data holder actif. Ce dernier ne sera pas supprimé de l'ensemble EDH, donc il sera un candidat comme les autres data holders candidat.
        ii. Après le $t_{déplacement}$, le data holder actif peut être l'ancien data holder actif.
        iii. L'avantage de ce choix (le data holder actif peut être un candidat) est que la probabilité du data holder reste toujours la même, car dans le cas où nous supprimons le data holder actif des candidats, la probabilité de connaitre le data holder est plus grande (a chaque fois

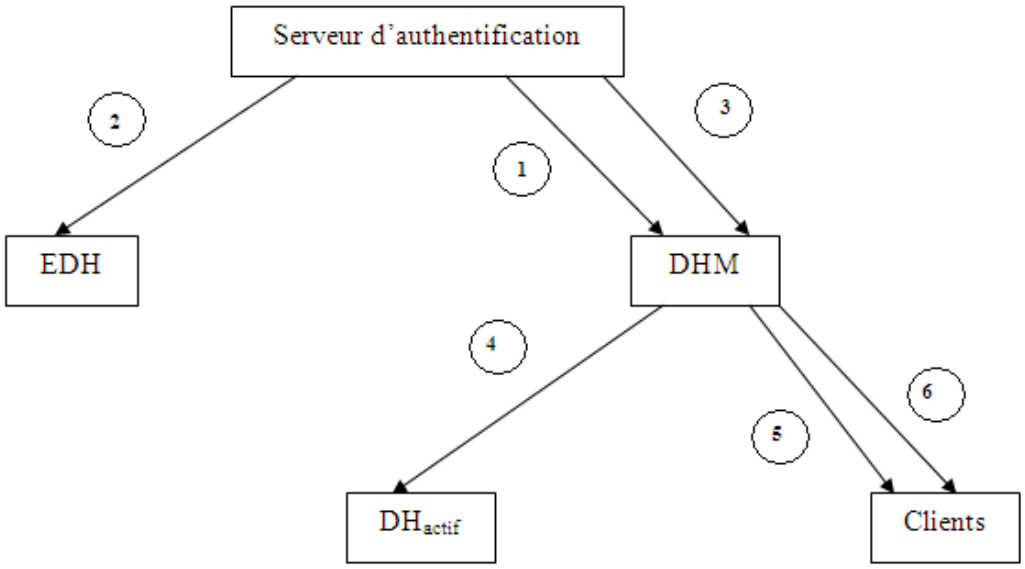
nous supprimons un data holder jusqu'il n'en reste qu'un seul et dont la probabilité de le connaitre est égale à un).

- Le DHM génère les tables MMT et MKT ainsi que le seed.
- Le DHM envoie aux clients les tables MMT et MKT partiel et le seed.
- Dans le cas où le $t_{déplacement}$ est constant, un attaquant peut prévoir quand agir. Le choix du $t_{déplacement}$ doit donc être aléatoire pour chaque changement du data holder. De plus, la valeur du $t_{déplacement}$ doit constituer un compromis entre surcharge du réseau et temps de tricherie, le but étant de diminuer le temps de tricherie sans surcharger le réseau. L'expérimentation [Norden et Guo, 2007] a montré qu'une valeur comprise entre 100 et 180 secondes permet d'avoir des résultats satisfaisants. Se basant sur ces constats, nous proposons la procédure suivante : le DHM choisit aléatoirement la valeur de $t_{déplacement}$ comprise dans l'intervalle 100s et 180s. Après chaque $t_{déplacement}$, le DHM envoie un message à tous les clients pour effectuer le changement du data holder. Puisque l'équité est assurée par le critère 2 cité dans IV.4.2, le message arrivera à tous les clients aux même temps.
- Dans le cas où un *data holder quitte le jeu* : Si ce dernier est le data holder actif, le DHM envoie aux clients un message pour le changement du data holder. Par contre, si le data holder ayant quitté est un candidat, le DHM envoie un nouveau *seed* aux clients pour générer une nouvelle séquence ID.

5. La partie client : le client a deux procédure, la première est l'envoi des messages au data holder et la seconde est le changement du data holder.
   - *Envoi de messages :* Le client envoie un message au DH en connaissant seulement son ID : il sélectionne l'entrée à partir de la table MMT qui représente une clé. Cette dernière sera utilisée pour déterminer l'adresse IP vers laquelle il va transmettre le message (prochain saut). Ceci pourra être fait en utilisant la table MKT. Le message sera acheminé a travers les nœuds jusqu'au DH actif.
   - *Changement de data holder :* cette procédure est réalisée lors de réception du message du changement du data holder par le DHM.

Algorithme principal :

Le diagramme ci-dessous illustre le fonctionnement global de l'algorithme principal :



1. le serveur d'authentification sélectionne aléatoirement DHM.
2. le serveur d'authentification sélectionne l'ensemble EDH.
3. le serveur d'authentification envoie au DHM l'ensemble EDH.
4. DHM sélectionne aléatoirement DH_{actif}.
5. DHM envoie aux clients (seed, MMT, MKT partielle).
6. le DHM envoie un message de changement aux clients après un $t_{déplacement}$ choisi aléatoirement.

L'algorithme est le suivant :

***SA* sélectionne aléatoirement le DHM à partir d'un ensemble de nœud du réseau ;**
***SA* sélectionne l'ensemble des data holders (EDH) ;** // voir l'algorithme de sélection du data holder
***SA* envoie l'ensemble EDH de chaque zone à DHM ;**
H : integer ; // nombre de zones
i : integer ;
$t_{déplacement}$ Min = 100 second ;
$t_{déplacement}$ Max = 180 second ;
pour i = 1 à H faire // chaque zone est traitée indépendamment des autres
  DH_{actif} = random (EDH) ; //le DHM sélectionne aléatoirement un data holder actif de l'ensemble des data holders candidat.
  ENVOI (seed, MMT, MKT partielle) ; // le DHM envoie aux clients le seed, la table MMT et MKT partielle par multicast.
  $t_{déplacement}$= random [$t_{déplacement}$ Min, $t_{déplacement}$ Max] ;

ENVOI (message du changement) ; // le DHM envoie aux clients un message pour changer le data holder actif.
fait ;
end.

Partie client :

ENVOI (id)
Clé = MMT [id] ; // générer la clé à partir d'id.
ip = MKT [clé] ; // avoir l'adresse IP de next hop
Transmettre (message, IP)
end.
Changement data holder :
$\mathbf{DH_{actif}}$ **= next (id);**
End.

IV.5. Conclusion

Dans ce chapitre, nous avons fait une synthèse globale sur les différentes solutions d'anti-tricherie, et nous avons proposé DACA-Extended.

Dans DACA-Extended, nous avons amélioré l'approche DACA par la minimisation du temps de tricherie, la confidentialité de l'adresse IP du data holder et la sélection du data holder qui assure la réactivité et l'équité.

Le tableau IV.3 montre ces améliorations par croix en rouge par comparaison à l'approche DACA qui sont comme suit.

- La protection du data holder : elle se fait en utilisant le changement planifié du data holder.
- La confidentialité de l'adresse IP : elle est assurée par le masquage de l'adresse IP, en utilisant le seed et les tables MMT, MKT.
- La réactivité : il assure un temps de réponse minimal par la sélection du data holder près de la majorité des joueurs.
- L'équité : tous les joueurs auront un temps de réponse égale.

Approches anti- tricherie	C/S				P2P												
	Comportement	Etat du Jeu	Type Tricherie		Etat		Cohérence	Authentification	Type Tricherie				Performance				
			Wall-Hacking	Bot	jeu	Superviseur			Collusion	Exploitation de la confiance mal placée	Exploitation du manque d'authentification	Exploitation du manque confidentialité	Equilibrage de charge	Taux de transfert	Adaptation	Temps réponse	Equité
Laurens et al., 2007	X		X														
Yeung et Lui, 2008		X		X													
Mitterhofer et al., 2009	X			X													
Kabus et al., 2005					X		X	X	X	X	X						
Norden et Guo, 2007						X		X	X		X	X					
Kabus et Buchmann 2007					X		X	X	X	X	X	X					
Chan et al., 2008								X			X						
Daniel et Soh, 2008						X		X	X		X					X	X
Liu et Tang, 2009					X		X	X			X		X	X	X		
DACA-Extended					X	X	X	X			X	X	X	X	X	X	X

Tableau IV. 3 : Synthèse des solutions d'anti tricherie existantes plus DACA-Extended

Conclusion générale

L'augmentation significative du nombre d'abonnés dans les MMOGs a poussé à la migration de l'architecture C/S vers l'architecture P2P qui exploite les ressources informatiques des joueurs. Or, ces derniers ont un accès illimité sur leurs propres machines et donc ils peuvent contourner la sécurité de leurs PC, d'où la vulnérabilité de l'architecture P2P à la tricherie.

La tricherie est considérée comme une menace très importante de sécurité dans les jeux en lignes. Elle se traduit en une activité qui modifie l'expérience du jeu pour donner à un joueur un avantage sur d'autre(s) joueur(s). Beaucoup de challenges sont soulevés dans le domaine de recherche pour pallier à ces problèmes de tricherie.

Ils existent plusieurs solutions proposées dans le but de remédier aux problèmes de tricheries. Alors que certaines se basent sur les architectures client/serveur, d'autres ont pour cible les architectures P2P.

L'étude des travaux existants nous a permis de :

- Constater que les solutions d'anti-tricherie basées sur les MMOGs à architecture C/S se basent sur l'analyse des comportements des joueurs suivant l'hypothèse qu'un tricheur présente un comportement distinguable d'un joueur normal. Les solutions d'anti-tricherie basées sur les MMOGs à architecture P2P se basent sur au moins un des trois critères suivants :

    1. **La Cohérence et le consensus de jeu** : ce critère se base sur l'état de jeu et se base sur le principe que les mêmes états initiaux et les mêmes événements se produisant dans le même ordre finiront dans les mêmes états finaux pour tous les joueurs.
    2. **Les superviseurs :** Ces approches se basent sur les superviseurs (parfois appelés aussi data holders, coordinateurs, contrôleurs de région, …) pour garantir que ces derniers ne sont pas corrompus ni objets d'attaque.

3. **Les signatures :** Ces types d'approches se basent sur la signature d'événement pour signer d'une manière efficace les messages des événements discrets.

- Dégager un ensemble de critères qui caractérisent les performances d'un MMOG, à savoir :

1. **Equilibrage de charge :** Equilibrer le nombre de joueurs dans chaque zone afin de répartir équitablement les charges à chaque joueur.
2. **Taux de transfert :** Réduire le taux de transfert.
3. **Adaptation :** Le nombre de joueurs en ligne n'est pas fixe, de ce fait fournir la fonction d'adaptation (fusion/division) est nécessaire dans une division de l'univers de jeu. Le redimensionnement dynamique d'une zone pour s'adapter au nombre des joueurs est nécessaire.
4. **Réactivité :** Le temps de réponse entre le superviseur et les clients doit être minimisé.
5. **Equité :** L'écart entre les différents temps de réponse entre le superviseur et les clients doit être limité.

Nous avons proposé l'approche DACA-Extended comme solution au problème de la tricherie dans les MMOGs à architecture P2P. L'idée principale de notre proposition est d'améliorer l'approche DACA en considérant les critères suivants : minimisation du temps de tricherie, traitement du problème de visibilité d'adresse IP, et le choix du Data-holder.

1. La minimisation du temps de tricherie est réalisé par l'ajout de la fonctionnalité du changement planifié du data holder. La solution clé est de changer le data holder actif par un data holder candidat et propager l'information à travers les différents clients du jeu (joueurs). Le changement du data holder survient dans le cas où le data holder ne remplit pas son rôle ou lors d'un changement planifié (après un $t_{déplacement}$).

   Nous avons choisi un $t_{déplacement}$ aléatoire pour empêcher un attaquant de prévoir quand agir quand il y a changement du data holder. Ce $t_{déplacement}$ a une valeur minimale de 100 secondes et une valeur maximale de 180 secondes.

Nous avons introduit un nouveau type de nœud appelé Data Holder Master (DHM) qui représente un data holder actif dans un ensemble de data holder candidats (EDH).

Nous avons sélectionné l'ensemble de data holders (EDH) en prenant en compte les deux critères (réactivité et équité). De plus le choix du prochain data holder est fait aléatoirement pour que le data holder ne soit pas déterministe et prévisible**.**

2. La visibilité d'adresse IP est réalisé par la construction de la table de hachage appelé Master Mapping Table (MMT), la table Master Key Table (MKT) et génération du Seed.
3. Le choix du data holder ; nous avons pris en considération les deux facteurs de performance suivants : la minimisation de temps de réponse et l'équité. En plus, nous avons proposé un mécanisme qui équilibre entre ces deux facteurs.

    - La minimisation de temps de tricherie : Pour cela nous avons choisi le data holder le plus proche de la majorité des joueurs.
    - L'équité : le data holder peut envoyer des mises à jour en retard à un pair avec un temps de réponse petit de sorte qu'il reçoive des mises à jour en même temps qu'un pair un avec un temps de réponse grand.
    - Mécanisme d'équilibrage : ce dernier a pour objectif d'assurer l'équité tout en minimisant le temps de réponse.

DACA-Extended apporte un ensemble d'améliorations à l'approche DACA :

- Elle permet de mieux protéger le data holder grâce au choix aléatoire de ce dernier ainsi que le temps de déplacement aléatoire.
- D'autre part, notre approche permet une meilleure confidentialité de l'adresse IP du data holder qui n'est plus envoyée en clair comme proposée dans DACA.
- Le mécanisme d'équilibrage entre la minimisation du temps de réponse et l'assurance de l'équité permet d'avoir de meilleures performances de jeu.

Le domaine de la tricherie dans les jeux en lignes massivement multi-joueurs (MMOG) basé sur les architectures P2P est un domaine récent qui suscite de plus en plus l'intérêt de la

communauté des chercheurs. Nous avons essayé à travers ce travail de contribuer dans ce domaine ; cependant, nous envisageons plusieurs perspectives afin de l'améliorer :

- Proposer un mécanisme de sécurité qui traite les problèmes de collusions entre les data holders et les joueurs. Ce problème survient par exemple ; si deux joueurs a et b s'entendent pour devenir data holder dans deux zones différentes (a contrôle la zone de b et b contrôle la zone de a).
- Proposer un mécanisme de protection de la communication entre le serveur d'authentification et les data holders afin d'éviter les attaques telles que celles du *Middle Attack*.
- Proposer un mécanisme de vérification des fiabilités des données afin de permettre au serveur d'authentification de vérifier si les données reçues par les pairs telles que la capacité de calcul et les coordonnées sont vraies ou non.
- Proposer une implémentation de l'approche DACA-Extended.

Bibliographie

[Alexander, 2005]: Alexander, T. Massively Multiplayer Game Development. Charles River Media, ISBN: 1584503904, 2nd edition, 2005.

[Anderson et al., 2002]: Anderson, D. P., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D. SETI@home: An Experiment in Public-Resource Computing. Communications of the ACM, 45(11):56–61, November 2002.

[Bengt et Rune, 2001]: Bengt, C., Rune, G. The Rise and Fall of Napster - An Evolutionary Approach. In Proceedings of the 6th International Computer Science Conference on Active Media Technology (ICAMT), 2001.

[Brookshier et al., 2002]: Brookshier, D., Govoni, D., Krishnan, N., Soto, J. C. JXTA: Java P2P Programming. Sams Publishing, ISBN: 0672323664, March 2002.

[Chen et al., 2006]: Chen, K-T., Jiang, J-W., Huang, P., Hao-Hua Chu, H-H., Lei, C-L., Chen, W-C. Identifying MMORPG Bots: A Traffic Analysis Approach. In Proc. 2006 ACM SIGCHI Int'l Conf. Advances in Computer Entertainment Tech., ACM Press, 2006, article no. 4.

[Chan et al., 2008]: Chan, M. C., Hu, S. Y., Jiang, J. R. An efficient and secure event signature (EASES) protocol for peer-to-peer massively multiplayer online games. In Computer Networks: The International Journal of Computer and Telecommunications Networking, Volume 52 Issue 9, June, 2008

[Clarke et Dede, 2005] : Clarke, J., Dede, C. Making learning meaningful: An exploratory study of using multi-user environments (MUVEs) in middle school science. Presented at AERA, Montreal, April, 2005.

[Conn et al., 2003]: Conn, G., Doctorow, C., Henson, J. Collaboration Object Lookup Architecture. Technical report, opencola.com, 2003.

[Corman et al., 2006]: Corman, A., Douglas, S., Schachte, P., Teague, V. A secure event agreement (SEA) protocol for peer-to-peer games. In: Proceedings of the First International Conference on Availability, Reliability and Security, 2006.

[Dabek et al., 2004]: Dabek, F., Cox, R., Kaashoek, F., Morris, R. Vivaldi: A Decentralized Network Coordinate System. In SIGCOMM'04, Portland, Oregon, USA, 2004.

[Damer, 1997] : Damer, B. Avatars: Exploring and Building Virtual Worlds on the Internet. Peachpit Press, 1997. ISBN 0-201-68840-9.

[Daniel et Soh, 2008]: Daniel, S., Soh, S. Secure referee selection for fair and responsive peer-to-peer gaming. In 22nd Workshop on Principles of Advanced and Distributed Simulation (PADS '08). IEEE. 2008, 63-71. Piscataway, NJ, USA., 2008.

[Dickey et al., 2004]:Dickey, C., Zappala, D., Lo, V., J. Low latency and cheat-proof event ordering for peer-to-peer games, in: Proceedings of the ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Kinsale, County Cork, Ireland, 2004, pp. 134–139.

[Douglas et Peucker, 1973]: Douglas, D. H., Peucker, T. K. Algorithms for the Reduction of the Number of Points Required to Represent a Line or its Caricature. In The Canadian Cartographer, vol. 10, no. 2, 1973, pp. 112–122, 1973.

[Doval et O'Mahony, 2003]: Doval, D., O'Mahony, D. Overlay Networks - a Scalable Alternative for P2P. IEEE Internet Computing, July 2003.

[Eckey et al., 2002]: Eckey, H. F., Kosfeld, R., Rengers, M. Multivariate Statistics, Gabler, 2002.

[Fan, 2009]: Fan, L. Solving Key Design Issues for Massively Multiplayer Online Games on Peer-to-Peer Architectures. Thèses de Doctorat, Université Heriot-Watt, 2009.

[Fan et al., 2010]: Fan, L., Trinder, P., Taylor,H. Deadline-Driven Auctions for NPC host allocation in P2P MMOGs. In International Journal of Advanced Media and Communication. Volume 4, Issue 2, pages 140-153, 2010.

[Gusfield, 1997]: Gusfield, D. Algorithms on Strings, Trees, and Sequences. In Computer Science and Computational Biology, Cambridge Univ. Press, 1997.

[Hu et Chang, 2007]: Hu, M. M. Chang, B. Massively Multiplayer Online Game Supported Foreign Language Listening Ability Training. In Proceedings of the The First IEEE International Workshop on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL), pages 176–178. IEEE Computer Society, 2007.

[Izal et al., 2004]: Izal, M., Urvoy-Keller, G., Biersack, E., Felber, P., Hamra, A. A., Garc'es-Erice, L. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In Proceedings of the 5th Passive and Active Measurement Workshop (PAM). Springer, 2004.

[Kabus et al., 2005]: Kabus, P., Terpstra, W. W., Cilia, M., and Buchmann, A. P. Addressing Cheating in Distributed MMOGs. In NetGames '2005: Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games, pages 1-6, New York, NY, USA, 2005.

[Kabus et Buchmann 2007]: Kabus, P., Buchmann, A. P. Design of a Cheat-Resistant P2P Online Gaming System. In DIMEA '2007 Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts, ACM New York, NY, USA, 2007.

[Kant et al., 2002]: Kant, K., Iyer, R., Tewari, V. A Framework for Classifying Peer-to-Peer Technologies. In Proceedings of the 2nd International Symposium on Cluster Computing and the Grid (CCGrid), page 368. IEEE, 2002.

[Karbhari et al., 2004]: Karbhari, P., Ammar, M., Dhamdhere, A., Raj, H., Riley, G., len Zegura, E. Bootstrapping in Gnutella: A Measurement Study. In Proceedings of the 5th Passive and Active Measurement Workshop (PAM). Springer, 2004.

[Kesselman, 2005]: Kesselman, J. Server Architectures for Massively Multiplayer Online Games. In Session TS-1084 Javaone Conference. Sun, 2005.

[Knutsson et al., 2004]: Knutsson, B., Lu, H., Xu, W., Hopkins, B. Peer-to-peer support for massively multiplayer games. In IEEE INFOCOM, 2004.

[Laurens et al., 2007]: Laurens, P., Paige, R. F., Brooke, P. J., Chivers, H. A Novel Approach to the Detection of Cheating in Multiplayer Online Games. 12th IEEE international conference on engineering complex computer systems, ICECCS 2007, Auckland, 2007.

[Liu et al., 2003] : Liu, D., Vo, L., Gainsbrugh, J. Gaia Online. Technical report, Gaia Interactive, 2003.

[Liu et Tang, 2009]: Liu, H. I., Tang, B. R. DACA: Dynamic Anti-Cheating Architecture for MMOGs. In AINA '2009 Proceedings of the 2009 International Conference on Advanced Information Networking and Applications, IEEE Computer Society Washington, DC, USA, 2009.

[Matthew et al., 2005]: Matthew, Y., Brian, N. L., Arnold, L. R. On the Cost-Ineffectiveness of Redundancy in Commercial P2P Computing. In ACM, Alexandria, Virginia, USA, 2005.

[Merabti et Rhalibi, 2004]: Merabti, M. Rhalibi, A. E. Peer-to-peer Architecture and Protocol for a Massively Multiplayer Online Game. In Proceedings of the Global Telecommunications Conference Workshops (GlobeCom), pages 519–528. IEEE, 2004.

[Mitterhofer et al., 2009]: Mitterhofer, S., Kruegel, C., Kirda, E., Platzer, C. Server-Side Bot Detection in Massively Multiplayer Online Games. IEEE Security Privacy Magazine (2009) **Volume:** 7, Issue: 3, Pages: 29-36, 2009.

[Mulligan et Patrovsky, 2003]: Mulligan, J. Patrovsky, B. Developing Online Games - An Insider's Guide. New Riders Publishing, ISBN: 1592730000, 2003.

[Networks, 2008]: Blue Falcon Networks - Company Profiles & Financials. Technical report, Hoover's Company Records, 2008.

[Norden et Guo, 2007]: Norden, S., Guo, K. Support for resilient Peer-to-Peer gaming. In Computer Networks: The International Journal of Computer and Telecommunications Networking, Volume 51 Issue 14, 2007.

[Ott et Dillenbourg, 2001] : OTT, D., Dillenbourg, P. Using Proximity and View Awareness to Reduce Referential Ambiguity in a Shared 3D Virtual Environment. In Proceedings of CSCL 2001.

[Pinedo, 2001]: Pinedo, M. Scheduling: Theory, Algorithms, and Systems (2nd Edition). Prentice Hall Edition, 2001. ISBN-13: 978-0130281388.

[Ratnasamy et al., 2001]: Ratnasamy, S., Francis, P., Handly, M., Karp, R., Schenker, S. A scalable content addressable network. In Proceedings of the conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM), pages 161–172. ACM, 2001.

[Robles et al., 2008]: Robles, R. J., Yeo, S., Moon, Y., Park, G., Kim, S. Online Games and Security Issues. In: Future Generation Communication and Networking, 2008. FGCN '2008.

[Rowstron et Druschel, 2001]: Rowstron, A., Druschel, P. Pastry: Scalable, Decentralized Object Location and Routing for Large Scale Peer-to-Peer Systems. In Proceedings of 18th IFIP/ACM international conference on distributed systems platforms (Middleware), pages 329–350. ACM, 2001.

[Stoica et al., 2001]: Stoica, I., Morris, R., Karger, D., Kaashoek, F. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In Proceedings of the 1st ACM SIGCOMM Workshop on Network and System Support for Games (NetGames), pages 149–160. ACM, 2001.

[Tay, 2005]: Tay, V. Massively Multiplayer Online Game (MMOG) - a Proposed Approach for Military Application. In Proceedings of the International Conference on Cyberworlds (CW), pages 396–400. IEEE Computer Society, 2005.

[Tutschku, 2004]: Tutschku, K. A Measurement-based Traffic Profile of the eDonkey Filesharing Service. In Proceedings of the 5th Passive and Active Measurement Workshop (PAM). Springer, 2004.

[Webb, 2002]: Webb, C. Peer-to-Peer Application Development. Hungry Minds, ISBN: 0764549049, 2002.

[Yan et Randell, 2005]: A systematic classification of cheating in online games. In NetGames ’2005: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, pages 1–9, New York, NY, USA, 2005.

[Yang et al., 2004]: Yang, Z., Lang, W., Tan, Y. A new fair micropayment system based on hash chain. In Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE’04), 2004, pp. 139–145.

[Yen et Zheng, 2000]: Yen, SM., Zheng, Y. Weighted One-Way Hash Chain and Its Applications. In Proceeding of the ACM ISW '00 Proceedings of the Third International Workshop on Information Security, 2000.

[Yeung et Lui, 2008]: Yeung, S. F., Lui, J. C. S. Dynamic Bayesian approach for detecting cheats in multi-player online games. In Multimedia systems ISSN 0942-4962, vol. 14, n°4, pp. 221-236, Springer, 2008.

[Zhao et al., 2001]: Zhao, B., Kubiatowicz, J., Joseph, A. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical report, UC Berkeley, 2001.

Glossaire

MMOG	Massively Multiplayer Online Games
MUD	Multi-User Dungeon
EV	Environnement Virtuel
MMORPG	Massive Multi-player Online Role-Playing Games
MMOFPS	Massively Multiplayer Online First-Person Shooter
MMORTS	Massively Multiplayer Online Real-Time Strategy
WOW	World of Warcraft
PW	Persistent World
NPC	Non-Player Characters
PC	Player Characters
AOI	Area of Interest
DACA	Dynamic Anti-Cheating Architecture
QoS	Quality of Service
TC	Trusted Computing
SRS	Secure Referee Selection.
RSP	Referee Selection Problem
IDS	Intrusion Detection System
BN	Bayesian Networks
DBN	Dynamic Bayesian Network
DoS	Deny of Service
RC	Region Controller
WRD	Weighted Responsible Division
DH	Data Holder
DHM	Data Holder Master
MMT	Master Mapping Table
MKT	Master Key Table