UQAC
Université du Québec
à Chicoutimi

ÉTUDE DE L'IMPACT DE L'ARCHITECTURE PUB/SUB SUR L'EFFICIENCE DU TRAITEMENT DES DONNÉES DANS LE CADRE DES JEUX VIDÉO MASSIVEMENT MULTIJOUEUR.

PAR LUCAS GUICHARD - [redacted]

MÉMOIRE PRÉSENTÉ À L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI EN VUE DE L'OBTENTION DU GRADE DE MAÎTRISE ÈS SCIENCES (M. SC.) EN INFORMATIQUE

QUÉBEC, CANADA

RÉSUMÉ

Ce mémoire explore l'impact du modèle de communication « *publish/subscribe* » sur l'efficience énergétique des serveurs de jeux massivement multijoueurs en ligne. Dans un contexte où les Technologies de l'Information et de la Communication (TIC) représentent une part croissante des émissions de gaz à effet de serre mondiales, l'optimisation de la consommation énergétique des infrastructures numériques devient un enjeu majeur pour le développement durable.

Notre recherche s'articule autour de l'hypothèse suivante : l'adoption du modèle « *publish/subscribe* » pourrait permettre de réduire significativement le volume de données traitées par les serveurs de jeux, et par conséquent, leur consommation énergétique. Pour vérifier cette hypothèse, nous avons d'abord établi théoriquement et empiriquement la relation entre le volume de données traitées et la consommation énergétique des équipements réseau.

Après une analyse approfondie de l'état de l'art des architectures réseau pour les jeux massivement multijoueurs en ligne, nous avons identifié les forces et les faiblesses des solutions existantes. Nous avons ensuite proposé une expérimentation comparative entre une solution de réplication standard basée sur *Unreal Engine* [1] et une implémentation personnalisée utilisant le modèle « *publish/subscribe* ». Les résultats confirment que ce modèle permet de réduire considérablement la consommation de bande passante tout en maintenant un niveau de synchronisation adéquat entre les clients, démontrant ainsi son potentiel pour améliorer l'efficience énergétique des serveurs de jeux vidéo massivement multijoueurs.

Cette recherche ouvre des perspectives prometteuses pour le développement d'architectures de communication plus efficientes, et donc plus respectueuses de l'environnement dans le secteur des jeux vidéo, contribuant ainsi aux objectifs plus larges de l'informatique verte.

1. https://www.unrealengine.com/fr?sessionInvalidated=true, Saisi le 7 mai 2025

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS

Pub/Sub *Publish/Subscrib*
MOG *Multiplayer Online Game*
MMOG *Massively Multiplayer Online Game*
C/S Client/Serveur
P2P Pair-à-Pair
DR Demande-Réponse
TR Temps Réel
SAS *Software As a Service*
PNJ Personnage Non Joueur
TIC Technologies de l'Information et de la Communication
MOM *Message Oriented Middleware*
API *Application Programming Interface*
JMS *Java Message Service*
IoT *Internet of Things*
IIoT *Industrial Internet of Things*
MQTT *Message Queuing Telemetry Transport*
EPA Agence de Protection Environnemental
RoHS *Restriction of Hazardous Substances*
TSS *Trailing State Synchronization*
BSS *Baseline State Syncronization*
RUDP *Reliable User Datagram Protocol*
TCP *Transmission Control Protocol*
UDP *User Datagram Protocol*
LAN *Local Area Network*
IM *Interest Manager*
RPC *Remote Procedure Call*

REMERCIEMENTS

Je tiens à remercier chaleureusement Yannick Francillette et Valère Plantevin, mes directeurs de recherche, pour leur soutien indispensable tout au long de ce projet. Yannick a joué un rôle clé en m'incitant à clarifier et à préciser mes idées, ce qui a considérablement enrichi mon approche théorique. Sa capacité à questionner et à approfondir les concepts a été cruciale pour la rigueur de ce mémoire. Du côté de Valère, son accompagnement technique a été tout aussi essentiel. Il n'a pas seulement guidé mes recherches, mais il a aussi pris le temps de m'expliquer des concepts complexes et de me fournir des ressources précieuses difficiles à trouver. Son expertise et sa patience m'ont permis de surmonter les défis techniques rencontrés durant cette étude.

Un grand merci également à ma famille et à mes amis. Leur aide pour la relecture et les corrections a été incroyablement précieuse. Je les remercie d'avoir pris le temps de lire et relire ce long document sur leur temps libre.

AVANT-PROPOS

La passion pour le développement logiciel m'habite depuis mon plus jeune âge, me poussant continuellement à explorer les rouages complexes des systèmes informatiques et à perfectionner mes compétences en programmation. L'algorithmie, avec ses défis logiques et sa quête d'optimisation, a toujours exercé sur moi une fascination particulière, m'incitant à rechercher constamment les solutions les plus élégantes et efficientes aux problèmes rencontrés.

Mon intérêt pour les jeux vidéo multijoueurs, et plus spécifiquement pour les jeux massivement multijoueurs en ligne, s'est imposé comme une évidence au fil de mon parcours. Ces environnements virtuels, capables d'accueillir simultanément des milliers d'utilisateurs interagissants en temps réel, constituent à mes yeux un défi technique fascinant. L'architecture sous-jacente à ces mondes persistants représente un terrain d'exploration riche en complexités algorithmiques et en problématiques d'optimisation.

Parallèlement à ces aspirations professionnelles, j'ai développé une conscience aiguë des enjeux environnementaux liés au numérique. Dans le contexte actuel, où la préservation des ressources énergétiques devient impérative, l'informatique verte s'impose comme une nécessité incontournable. Ce mémoire traduit ainsi ma volonté de concilier ma passion pour le développement de jeux massivement multijoueurs avec l'impératif écologique, en explorant des solutions qui permettraient de réduire l'empreinte énergétique de ces infrastructures tout en maintenant leur performance et leur capacité à fournir des expériences de jeu riches et immersives.

INTRODUCTION

Dans le contexte actuel de transformation numérique, le secteur des Technologies de l'Information et de la Communication (TIC) représente une composante cruciale de l'économie mondiale. L'avènement d'Internet et des réseaux de communication a non seulement révolutionné nos modes de vie, mais a également stimulé une croissance exponentielle dans divers domaines tels que le commerce, l'éducation, la santé et le divertissement. Cette intégration croissante des TIC dans presque tous les aspects de notre quotidien soulève cependant des préoccupations importantes liées à l'impact environnemental de ces technologies. En effet, alors que ces systèmes facilitent des innovations majeures, ils reposent sur une infrastructure physique imposante, comprenant des serveurs, des centres de données et des réseaux de distribution qui consomment d'importantes quantités d'énergie.

Cette consommation énergétique massive se traduit par des émissions significatives de gaz à effet de serre, plaçant ainsi le secteur des TIC parmi les contributeurs notables au réchauffement climatique. En 2021, les TIC représentaient 3,5 % des émissions de gaz à effet de serre mondiales, tout en prévoyant de doubler pour 2025 [1]. La croissance continue du trafic de données et de la demande en bande passante est un principal facteur de cette consommation d'énergie [2]. Dans ce contexte, il devient impératif d'explorer et de mettre en œuvre des pratiques plus responsables visant à réduire la consommation énergétique et à minimiser l'impact environnemental du secteur.

Parmi les plus gros générateurs et consommateurs de données des TIC, on retrouve le streaming (*YouTube* [3], *Netflix* [4], *Discord* [5], etc.) et les jeux vidéo en ligne (karmametrix.com). La popularité du jeu vidéo auprès de toutes les générations fait de lui un des supports de divertissement majeur. Le rapport de 2020 de l'association canadienne du logiciel de divertissement compte 23 millions de joueurs, ce qui représente 61 % de la population [2]. En 2023, l'industrie mondiale du jeu vidéo continue de croître, générant des revenus considérables. Selon *Newzoo*, le marché global des jeux a atteint 187,7 milliards de dollars en 2023, avec une prévision d'augmentation jusqu'à 212,4 milliards de dollars d'ici à 2026. Le segment des

2. https://karmametrix.com/blog/web-sustainability/internet-carbon-footprint-data/, Saisi le 7 juillet 2025
3. https://www.youtube.com/, Saisi le 7 juillet 2025
4. https://www.netflix.com/, Saisi le 7 juillet 2025
5. https://discord.com/, Saisi le 7 juillet 2025

jeux sur console a été particulièrement performant, contribuant à hauteur de 56,1 milliards de dollars des revenus mondiaux, ce qui en fait le second segment le plus important après les jeux mobiles [6].

1 INFORMATIQUE VERTE

L'informatique verte, ou « *green computing* », désigne les pratiques et les technologies visant à réduire l'impact environnemental des systèmes informatiques. Cette réduction passe, entre autres, par l'amélioration de l'efficience des infrastructures logicielles et matérielles du secteur numérique. C'est une démarche qui s'applique sur l'ensemble du cycle de vie d'un appareil, c'est-à-dire de sa conception jusqu'à sa destruction [3].

Au-delà des considérations matérielles, l'informatique verte s'intéresse également à l'optimisation des algorithmes et à la gestion efficiente des ressources de calcul. Des recherches récentes ont démontré que certains algorithmes peuvent être repensés pour réduire significativement leur consommation énergétique sans compromettre leurs performances fonctionnelles. L'optimisation des structures de données et des requêtes dans les applications en ligne peut, par exemple, générer des économies d'énergie substantielles. De même, l'émergence des techniques d'intelligence artificielle économes en ressources, comme la distillation de modèles ou l'élagage de réseaux neuronaux, témoigne d'une prise de conscience croissante de la nécessité d'intégrer les considérations énergétiques dans l'ensemble de la chaîne de développement logiciel.

2 QU'EST-CE QU'UN JEU

La définition du terme « jeu » est très difficile à déterminer, car le jeu tel que nous le connaissons peut prendre une multitude de formes différentes et avoir lieu dans des contextes différents. Par exemple, il y a les jeux de plateau comme le *Monopoly*[4] [7] ou les échecs[5], les jeux de cartes comme le *Uno* [8] ou la *Bataille* [9], les jeux d'arcade comme le « PinBall » [10] ou le jeu de hockey pneumatique, les jeux vidéo et bien d'autres encore. Dans « *Game Engine*

6. https://newzoo.com/resources/blog/explore-the-global-games-market-in-2023, Saisi le 7 juillet 2025
7. https://shop.hasbro.com/en-ca/monopoly, Saisi le 7 juillet 2025
8. https://service.mattel.com/instruction_sheets/FFK04-FR.pdf, Saisi le 7 juillet 2025
9. http://fr.wikipedia.org/w/index.php?title=Bataille_(jeu)&oldid=221895871, Saisi le 7 juillet 2025
10. https://sternpinball.com, Saisi le 7 juillet 2025

Architecture »[11] [6], Jason G. explique qu'un jeu est un environnement dans lequel plusieurs agents adoptent des stratégies et des tactiques pour maximiser leur gain dans un cadre régi par des règles établies. Il précise ensuite que dans le contexte de console ou d'ordinateur de jeu, le terme jeux désigne habituellement des images d'un monde en deux ou trois dimensions avec un personnage principal contrôlé par le joueur.

La notion de jeux vidéo multijoueur en ligne (« *Multiplayer Online Game (MOG)* ») réfère à la connexion de deux joueurs, ou plus, à travers le réseau internet. Lorsque le terme « multijoueur » désigne un groupe de plusieurs milliers de joueurs en simultané, on parle de jeu vidéo massivement multijoueur en ligne (« *Massively Multiplayer Online Game (MMOG)* »). Dans le cadre de notre recherche, seul l'échange d'information nous intéresse. D'un point de vue plus technique, on peut considérer qu'un jeu vidéo est une messagerie en ligne dans laquelle plusieurs milliers de personnes peuvent échanger à travers un ou plusieurs canaux.

3 CADRE DE L'ÉTUDE

Les principaux consommateurs d'énergie dans le secteur du numérique sont les serveurs. Les serveurs sont des ordinateurs ou des systèmes informatiques qui fournissent des ressources, des données, des services, ou des programmes à d'autres ordinateurs, appelés clients, via un réseau. Conçus pour être opérationnels de manière continue [7], ils assurent diverses fonctions selon les besoins des utilisateurs, comme l'hébergement de site web, la gestion des bases de données, ou le stockage de fichiers. Leur rôle central dans la gestion des échanges d'informations les rendent indispensables pour le bon fonctionnement des réseaux informatiques.

Appliqués au secteur du jeu vidéo, **les « *MMOGs* » sont particulièrement énergivores due à l'infrastructure importante nécessaire au traitement du volume massif de donnée généré par le besoin de synchronisation entre tous les joueurs.** Un moyen efficient de faire des économies d'énergie au niveau des serveurs de jeu serait de réduire le volume de données qu'ils doivent traiter. Dans ce mémoire, nous allons étudier **l'impact du modèle de communication « *publish/subscribe* » sur l'efficience des serveurs appliqués aux jeux vidéo**. Le but est de déterminer si ce modèle de communication pourrait réduire la consommation globale des serveurs de jeux, tout en conservant la qualité de leurs services.

11. http://fr.wikipedia.org/w/index.php?title=Air_hockey&oldid=223412992, Saisi le 7 juillet 2025

4 PLAN

Ce mémoire se divise en six chapitres. Tout d'abord, nous établirons les prérequis nécessaires à une compréhension approfondie de la recherche que nous présentons. Cette section posera les fondements théoriques et techniques indispensables pour aborder les concepts suivants. Dans une seconde partie, nous définirons les critères de recherche qui permettront de qualifier la pertinence des travaux déjà publiés pour notre recherche. Grâce à eux, nous effectuerons une revue de l'état de l'art sur l'usage du modèle de communication « *Publish/Subscrib (Pub/Sub)* » dans le contexte des « *MMOGs* ». Cette analyse détaillée permettra de cerner les applications actuelles et les limites des propositions utilisant ce modèle dans ce secteur spécifique. À la suite de cela, nous proposerons une nouvelle approche établie à partir de la classification de l'existant faite, qui s'appuie sur les critères de recherche définis préalablement. Notre proposition visera à combler les lacunes identifiées lors de l'analyse de l'état de l'art. Par la suite, notre proposition sera soumise à une évaluation rigoureuse avec les critères de recherche et les tests de performances présentées dans l'état de l'art. Cette évaluation permettra de mesurer l'efficacité et la pertinence de notre solution face aux défis actuels. La section cinq présentera une discussion globale du travail réalisé durant cette recherche. Nous porterons un regard critique sur les forces et les faiblesses de nos démarches tout en constatant les limites de nos contributions. Enfin, nous conclurons par une synthèse des résultats obtenus, en discutant de l'impact du modèle de communication « *publish/subscribe* » sur l'efficience des serveurs de jeux. Nous proposerons également des pistes pour de futures recherches, qui pourraient pousser plus loin les améliorations introduites par notre travail.

CHAPITRE I
PRÉREQUIS

Dans cette section, nous allons parler de toutes les notions qu'il est important de connaître pour comprendre les enjeux du secteur des « *MMOGs* » vis-à-vis de l'informatique verte. Dans un premier temps, nous allons introduire la notion d'informatique verte avec ses objectifs et ses différentes phases d'applications. Après quoi, nous présenterons les « *MMOGs* » en expliquant leur but et les défis techniques qu'ils requièrent pour fonctionner. Nous ferons le lien entre ces deux notions en introduisant le modèle de communication « *Pub/Sub* » avec son architecture, ses avantages et ses contraintes. Enfin nous introduirons l'hypothèse sur laquelle se basent les travaux de ce mémoire et nous détaillerons comment elle est vérifiée.

Tout au long du document, il est nécessaire de comprendre la nuance entre « efficacité » et « efficience ». Lorsque l'on parle d'efficacité, c'est le résultat qui est mis de l'avant. Le but est d'atteindre un résultat le plus vite possible, indépendamment des ressources nécessaires. À l'inverse, dans le cadre de l'efficience, c'est le coût en ressource que l'on cherche à minimiser tout en continuant d'atteindre le même résultat. Dans notre cas, il est très important de noter que l'on travaillera seulement dans un but d'efficience et non d'efficacité.

1.1 INFORMATIQUE VERTE

L'informatique émerge dans un contexte dans lequel l'usage intensif des technologies de l'information contribue de manière significative à la consommation énergétique mondiale et, par conséquent, à l'émission de gaz à effet de serre [8]. L'évolution et les décisions majeures qui englobent cette pratique sont visibles sur la figure 1.1.

1.1.1 HISTORIQUE

Depuis le début des années 1990, la prise de conscience de l'impact du mode de vie de l'homme sur l'équilibre de la nature conduit les générations à changer leurs habitudes. L'évolution de notre consommation de ressources naturelle passe par la conscientisation et l'éducation de la population vis-à-vis des risques liés à la surconsommation et la pollution. Dans ce but, *l'Agence de Protection Environnemental (EPA)* des États-Unis introduit en 1992

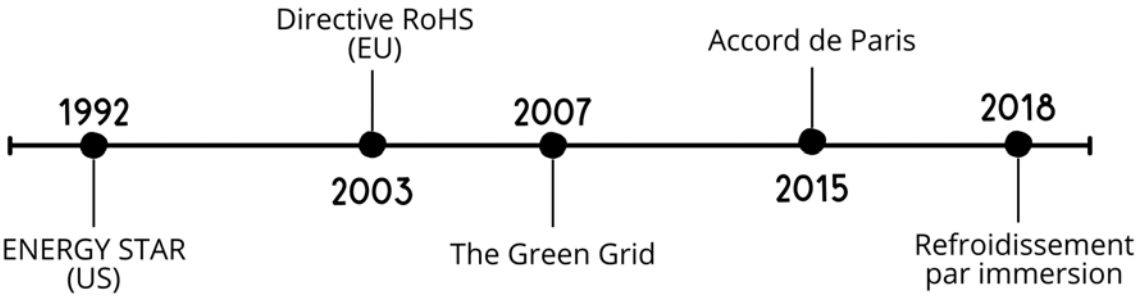

FIGURE 1.1 : Historique de l'informatique verte

le programme « *ENERGY STAR* » qui vise à réduire la pollution à travers l'amélioration de l'efficience de la consommation d'énergie [9]. Ce programme, basé sur le volontariat, donne lieu à une certification qui permet au consommateur d'identifier les produits (appareil électrique, maisons, etc.) qui respectent les critères imposés. De son côté, en 2003, l'Union européenne légifère l'utilisation des substances dangereuses dans les processus de fabrication des équipements électriques et électroniques en publiant la directive « *Restriction of Hazardous Substances (RoHS)* » [12]. En 2015, l'accord de Paris [13] engage ses 196 pays signataires à réduire considérablement leurs émissions de gaz à effet de serre.

Conscientes de l'impact de leurs centres de données sur l'environnement, de grandes entreprises du secteur telles que, « *Advanced Micro Devices* » (AMD) [14], *Dell* [15], *IBM* [16], *Intel* [17], *Microsoft* [18], décident de se regrouper pour former « *The Green Grid* » [19] en 2007. Ce consortium a pour but de concentrer les efforts des membres pour créer des outils, fournir de l'expertise technique et de promouvoir l'optimisation de l'efficience énergétique et des ressources des écosystèmes de centres de données. En 2024, ce consortium compte parmi ses

12. https://environment.ec.europa.eu/topics/waste-and-recycling/rohs-directive_en, Saisi le 7 juillet 2025
13. https://unfccc.int/fr/a-propos-des-ndcs/l-accord-de-paris, Saisi le 7 juillet 2025
14. https://www.amd.com/, Saisi le 7 juillet 2025
15. https://www.dell.com/, Saisi le 7 juillet 2025
16. https://www.ibm.com/, Saisi le 7 juillet 2025
17. https://www.intel.com/, Saisi le 7 juillet 2025
18. https://www.microsoft.com/, Saisi le 7 juillet 2025
19. https://www.thegreengrid.org/en/about-us, Saisi le 7 juillet 2025

membres les plus grandes entreprises du secteur telles que *Amazon Web Services* [20], *Cisco* [21], *Google* [22], *Nvidia* [23], *Oracle* [24], pour un total de 23 membres.

La consommation électrique des centres de données est répartie entre le service lui-même et les autres systèmes des bâtiments (refroidissement, éclairage, surveillance, etc.). On estime à 38 % la part de consommation de l'énergie totale d'un centre de données pour son refroidissement [10]. En 2018, un nouveau standard de refroidissement par immersion totale fait son entrée pour aider à réduire cette consommation avec comme bénéfice le fait de ne plus nécessiter de ventilateurs, de prolonger la durée de vie des serveurs en les protégeant de la poussière et d'avoir une capacité de dispersion thermique plus stable, rendant la température des serveurs uniforme.

La responsabilisation de la consommation du secteur informatique au fil des années s'est accompagnée d'une évolution de l'efficience des technologies logicielles tout en s'adaptant à la demande croissante. Les TIC ont particulièrement évoluée pour accompagner la croissance des communications réseaux dû à l'essor, entre autres, des réseaux sociaux et de « *l'Internet of Things (IoT)* ».

1.1.2 LE CYCLE DE VIE D'UN APPAREIL

Le cycle de vie d'un appareil informatique inclut sa conception, la fabrication, l'utilisation quotidienne et les mises à jour éventuelles. Finalement, lorsque l'appareil devient obsolète, il est dirigé vers le recyclage pour réutiliser ses composants et minimiser son impact environnemental. Lorsque l'on parle d'informatique verte, il est important de comprendre que ce terme englobe l'ensemble du cycle de vie des appareils et des logiciels que l'on utilise au quotidien [3].

Par exemple, dans le cadre de la construction d'une ferme de serveurs (regroupement de serveurs dans un bâtiment stratégique permettant de les contrôler et de les maintenir plus facilement), la première question qui survient durant la conception, est le lieu de sa

20. https://aws.amazon.com/, Saisi le 7 juillet 2025
21. https://www.cisco.com/, Saisi le 7 juillet 2025
22. https://www.google.com/, Saisi le 7 juillet 2025
23. https://www.nvidia.com/, Saisi le 7 juillet 2025
24. https://www.oracle.com/, Saisi le 7 juillet 2025

construction. En effet, placer une telle structure proche de la production d'énergie permet de réduire les pertes énergétiques liées au transport de cette dernière. Le professeur Andy Hooper a notamment affirmé que « il est toujours moins cher de déplacer de l'information que de l'énergie » [11].

Une fois la conception bien étudiée, c'est au tour de la réalisation du projet de prendre part à l'informatique verte en préférant, par exemple, l'usage de ressources renouvelables et/ou recyclées lors de sa création. Dans le cadre d'un ordinateur, il est très fréquent de le fabriquer à partir de plusieurs composants individuels issus de la récupération plutôt que du secteur du neuf [12].

La troisième étape de ce cycle de vie est aussi la plus connue, c'est celle de l'utilisation. L'utilisation optimisée pour une consommation énergétique réduite passe à travers l'amélioration de l'efficience des logiciels déployés sur ces machines, mais aussi leur utilisation raisonnée [3]. D'un point de vue logiciel, une utilisation raisonnée peut se traduire par la réduction de temps d'utilisation de ce dernier au strict nécessaire. Pour les appareils et les logiciels de grande ampleur, il est possible de les concevoir plus efficients en anticipant le contexte de leur utilisation. Par exemple, en pré-calculant toutes les issues d'un programme dont les possibilités sont finies, on permet de ne plus consommer d'énergie dans le calcul des résultats sur le reste de la durée de vie de cet appareil [13].

Après avoir disposé d'un appareil numérique, il existe deux possibilités : on peut soit décider de le recycler, comme évoqué plus haut, ou alors de l'améliorer en remplaçant seulement les composants qui sont défectueux, ou qui ne sont plus assez performants. L'ensemble de toutes ces mesures permettent de réduire durablement l'impact du secteur du numérique sur la pollution mondiale. Dans ce mémoire, nous nous concentrons sur la phase de l'utilisation responsable. Nous allons étudier les possibilités parmi les différents moyens de communications existants pour gagner en efficience sur la gestion des données des serveurs utilisés par les « *MMOGs* ».

1.2 JEUX MASSIVEMENT MULTIJOUEURS EN LIGNE

L'ensemble des joueurs, événements et états de l'environnement d'un jeu, est appelé « le monde ». Parmi tous ces éléments, les éléments dits « répliqués » sont ceux dont l'état doit être synchronisé entre les joueurs. Pour modifier l'état d'un élément répliqué, il faut avoir

l'autorité. Dans les jeux en ligne, le principe d'autorité permet de définir quelle version du jeu (quel ordinateur) peut décider ou non de l'état d'un élément du jeu [14]. C'est notamment grâce à ce mécanisme que l'on peut protéger les jeux de la triche en empêchant les joueurs malicieux de changer l'état d'un objet sans la permission de l'entité autoritaire du jeu (souvent, le serveur).

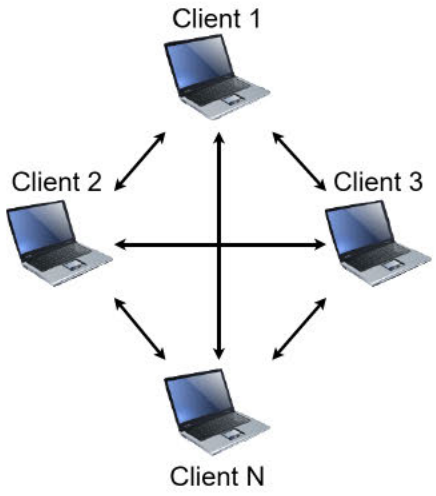

FIGURE 1.2 : Architecture « *P2P* » simple

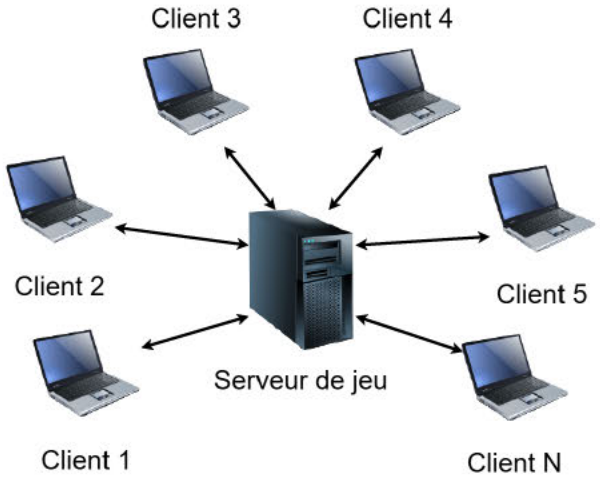

FIGURE 1.3 : Architecture client/serveur

Toutes les personnes voulant jouer au jeu doivent se connecter entre elles pour envoyer et recevoir les informations de synchronisation de jeu de la part des autres joueurs. Pour réaliser cela, il existe plusieurs types d'architectures telles que le pair à pair (« *P2P* ») (voir la figure 1.2 et le Client/Serveur (« *C/S* ») (voir la figure 1.3). Là où le « *P2P* » consiste à créer une connexion directement entre les joueurs [15], l'architecture « *Client/Serveur (C/S)* » consiste, elle, à connecter tous les joueurs (les clients) à un ordinateur central (le serveur) qui a la responsabilité de redistribuer les informations entre les clients [16]. Même si l'architecture « *P2P* » à l'avantage de ne pas nécessiter de serveur pour faire communiquer les joueurs, elle atteint vite des restrictions, car ce sont les clients qui doivent envoyer l'information à tous les autres joueurs connectés à chaque fois. Dans des situations avec beaucoup de joueurs tels que dans les « *MMOGs* », cela ne serait pas possible, car cela demanderait une quantité d'envois d'information de la part des clients beaucoup trop grande. De plus, une architecture partagée comme le « *P2P* » demande une gestion de l'autorité beaucoup plus complexe, car il n'y a pas de client avec de plus grande permission qu'un autre. Pour tenter de résoudre ce problème, une version de l'architecture « *P2P* » existe avec un arbitre [17] (voir la figure 1.4). Cet arbitre est un client choisi, ou un nouveau client non-joueur, qui est chargé de résoudre les erreurs de synchronisation entre les joueurs. Les erreurs de synchronisation proviennent du fait que tous les clients peuvent modifier le monde du jeu localement sans avoir encore reçu les informations des autres joueurs. Il est donc possible de se retrouver avec des situations litigieuses dans lesquels deux joueurs effectuent différentes actions en même temps sur un même objet. Les deux clients en situation de litige communiquent donc avec l'arbitre (qui possède sa propre copie du monde et l'historique des données de synchronisation) pour savoir qui a « raison » et qui doit annuler ses actions. Malgré ce système, les corrections apportées par l'arbitre sont encore coûteuses, car il est nécessaire à l'arbitre d'envoyer la correction à tous les clients du système, ce qui peut entrainer d'autres situations de litige.

Maintenant, du côté de l'architecture « *C/S* », les clients ne communiquent plus qu'avec le serveur, ce qui divise grandement la quantité de traitements demandés aux clients. De plus, toutes les informations transitant par un seul point, les litiges peuvent donc être traités en interne, et ainsi envoyer toujours des informations justes (ou déjà corrigées) aux clients. Enfin, la gestion de l'autorité est plus intuitive avec cette architecture. Le serveur ayant en permanence la version du jeu la plus proche de la réalité, c'est lui qui est le plus à même de décider si une action est valide en fonction de l'état du monde. Cependant, cette architecture

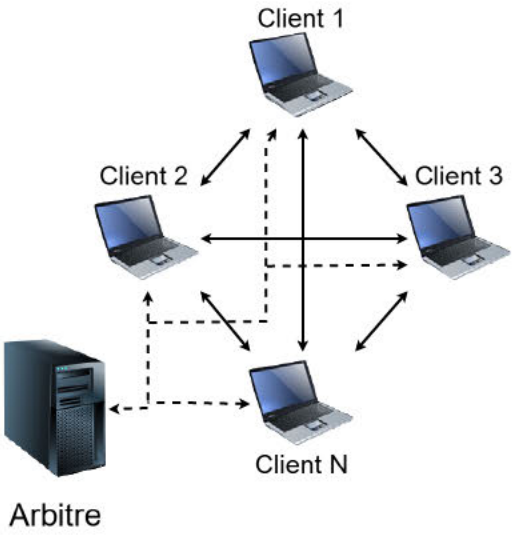

FIGURE 1.4 : Architecture « *P2P* » avec arbitre centrale

est très coûteuse en termes de performance sur le serveur, car c'est lui qui doit héberger le jeu, valider et synchroniser les actions des joueurs, ce qui implique un nombre élevé de calculs.

Dans le cadre des « *MMOGs* », l'architecture « *C/S* » est modifiée et renforcée pour pouvoir assurer la gestion efficace de vastes mondes de jeu avec de grandes populations de joueurs. Comme le montre la figure 1.5, les serveurs sont souvent structurés de manière hiérarchique et/ou distribuée pour optimiser les performances et réduire la latence. Par exemple, un serveur principal peut gérer les interactions globales, tandis que plusieurs sous-serveurs prennent en charge des régions ou des instances spécifiques du monde de jeu. Les serveurs peuvent aussi être répartis en fonction du service qu'ils fournissent (la gestion des inventaires, des statistiques, etc.)[18]. Cette architecture est accompagnée d'un système de répartition des charges qui permet de garder équilibrée la charge de travail de tous les serveurs, et ainsi avoir un jeu le plus fluide et stable possible. Cette architecture permet non seulement de gérer les états du jeu de manière centralisée, mais aussi d'assurer la sécurité et l'intégrité du jeu en contrôlant strictement les interactions des joueurs et la validation des données en interne. Enfin, la redondance de serveurs permet également de réduire les interruptions de service causées par d'éventuelles pannes, mises à jour ou redémarrages des machines.

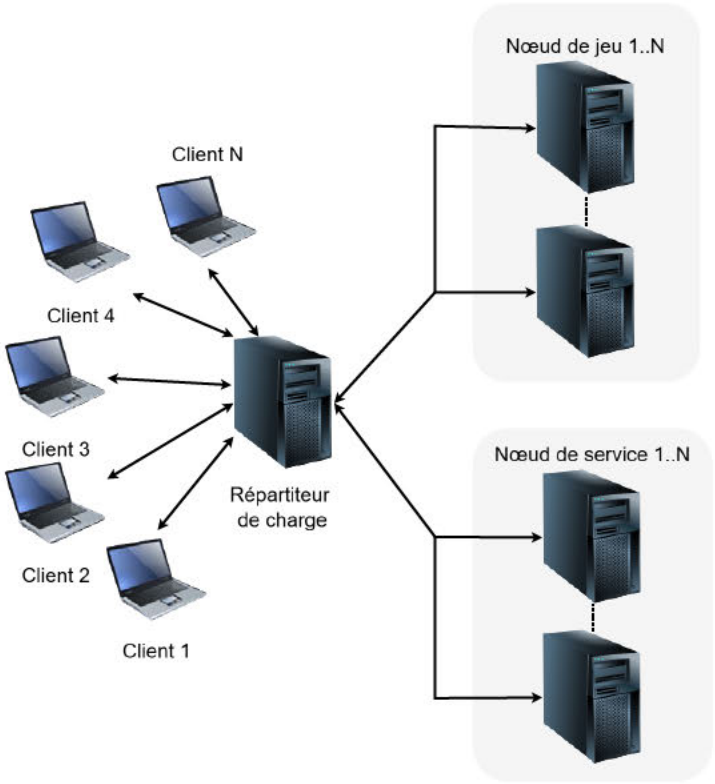

FIGURE 1.5 : Architecture client/multi-serveurs

Les « *MMOGs* », en engendrant de nombreuses interactions sociales et regroupant des milliers de joueurs en simultané dans ses mondes, nécessitent une transmission rapide et massive des données qu'ils génèrent, consommant ainsi d'importantes quantités d'énergie et de ressource de calcul. La question de l'efficience de l'acheminement de ces données se pose alors. Les joueurs ont-ils vraiment besoins de recevoir l'ensemble de l'état de la région du monde où ils se trouvent pour jouer convenablement ? Sur plusieurs « *MMOGs* » tel que *MAMMOTH* [19], il a été prouvé que non ; que le découpage des informations en groupe logique pouvait permettre de filtrer les informations envoyées aux joueurs. C'est pourquoi l'introduction de protocoles réseau plus structurés pourrait donc jouer un rôle transformateur, non seulement en améliorant l'efficience des serveurs, mais aussi en réduisant leur empreinte écologique. Dans ce paradigme, le développement de protocoles réseau, inspiré d'architecture ayant les mêmes besoins, représenterait une avancée majeure dans le cadre de l'informatique verte appliquée au « *MMOGs* ».

1.3 MODÈLE DE COMMUNICATION

Les modèles de communication jouent un rôle fondamental dans l'efficience du traitement et de la distribution des données. À travers internet, il existe trois modèles principalement utilisés : *Demande-Réponse (DR)*, *Temps Réel (TR)* et « *Pub/Sub* ». Chacun d'eux a été conçu pour répondre à une configuration de communication et est présenté plus en détail ci-dessous.

Dans un premier temps, le modèle *DR* se base sur une communication symétrique dans laquelle le client fait une demande (une requête) à un serveur ou à un service, qui répond ensuite avec les données ou les résultats souhaités. Ce modèle est omniprésent dans les interactions client-serveur sur le web et dans de nombreuses autres applications de réseau [20]. Ce modèle est la base de nombreux protocoles de communication, comme *HTTP*. Il a l'avantage de pouvoir envoyer et recevoir une donnée très précise, mais cela peut aussi devenir un inconvénient si le nombre de requêtes augmente trop. En effet, ce modèle présente des lacunes en termes d'élasticité, car chaque requête nécessite une réponse, et donc le risque de congestion au niveau du serveur est grand si trop de clients effectuent des requêtes en même temps.

Pour réduire ce problème et les requêtes répétitives, le modèle de communication en *TR* est apparu [25]. Utilisé par des technologies comme « *WebSocket* », ce modèle permet une interaction continue et bidirectionnelle entre les clients et les serveurs. Il est essentiel pour des applications nécessitant des mises à jour immédiates et fréquentes, telles que les jeux en ligne, les applications de chat en direct, et les systèmes de collaboration en temps réel. Il a l'avantage de ne pas nécessiter de faire de requête préalable, ce qui permet d'envoyer et de recevoir des informations dans des délais très brefs. Cependant, une fois encore, ce modèle atteint vite sa limite lorsque le nombre de clients augmente, car le serveur doit envoyer une copie de la même information à tous les clients connectés. De plus, ce modèle demande de maintenir la connexion entre le client et le serveur tout au long des échanges afin de garantir la bonne synchronisation des données.

À la racine des modèles de communication précédemment décrits se trouvent des protocoles réseau fondamentaux qui assurent le transport effectif des données. Parmi eux, « *Transmission Control Protocol (TCP)* » constitue l'une des fondations des communications

25. https://web.archive.org/web/20230316114234/https://www.ccontrols.com/pdf/Extv5n3.pdf, Saisi le 7 juillet 2025

réseau modernes, offrant un mécanisme de transfert fiable et orienté connexion. Contrairement aux modèles uniquement orientés vers la rapidité, « *TCP* » assure l'intégrité des données en établissant une connexion bidirectionnelle persistante, garantissant ainsi la livraison ordonnée et complète des paquets [21]. Cette approche méthodique sous-tend souvent les modèles *DR* où l'exactitude des données prime, mais engendre une latence non négligeable due aux vérifications constantes et aux retransmissions des paquets perdus, ce qui peut s'avérer problématique dans les contextes temps réel des « *MMOGs* ».

En réponse à ce compromis entre fiabilité et rapidité, « *User Datagram Protocol (UDP)* » propose une approche complémentaire qui n'est pas orientée connexion et sans garantie de livraison. Son fonctionnement léger élimine les mécanismes de contrôle inhérents à « *TCP* » pour privilégier la rapidité, faisant de ce protocole un choix adapté pour les communications *TR* où la rapidité de transmission de l'information surpasse l'importance de son intégrité absolue. « *UDP* » fournit un service minimal qui permet aux applications d'accéder directement au service de livraison de datagrammes du réseau [22]. Cette simplicité structurelle confère aux développeurs une liberté considérable pour implémenter leurs propres mécanismes de contrôle adaptés aux besoins spécifiques de leurs applications. Dans le contexte des jeux en ligne, « *UDP* » sert fréquemment de support aux données dynamiques comme les déplacements des joueurs, où une information légèrement altérée, mais rapide, reste préférable à une information parfaite, mais tardive.

Le modèle « *Pub/Sub* », lui, est basé sur un mécanisme de messagerie où les émetteurs de messages, appelés « *Publishers* » (Publieurs), ne transmettent pas directement les informations aux récepteurs, appelés « *Subscribers* » (Abonnés). Au lieu de cela, les messages des publieurs sont envoyés à un appareil tiers que l'on appelle un « *broker* » [23]. Les messages envoyés sont classés dans des catégories et les abonnés reçoivent seulement les informations qui correspondent aux catégories auxquelles ils ont souscrit. Les catégories sont appelées des sujets, ou « *Topic* » en anglais. Les sujets représentent des canaux d'échange d'informations sur lesquels les informations ont toutes un rapport avec le sujet principal. La plupart du temps, les sujets sont séparés en sous-sujets pour permettre un filtrage plus précis de l'information reçue. Le rôle du « *broker* », entre les publieurs et les abonnés, est donc de recevoir tous les messages de tous les sujets, et ainsi pouvoir les redistribuer aux clients abonnés au sujet du message en question. Le « *broker* » a également le devoir de gérer la cohérence des sujets pour éviter de manipuler un trop grand nombre de données inutilement. Dans l'utilisation du modèle « *Pub/Sub* », seul le « *broker* » échange des informations avec le serveur de jeu, ce qui

permet de réduire considérablement la charge de travail du serveur vis-à-vis de la redistribution de l'information. L'intérêt d'utiliser un « *broker* » et d'externaliser la tâche de transmission des données est de pouvoir faire varier le nombre de « *brokers* » à la volée en fonction du nombre de joueurs, mais aussi de leur localisation géographique. Dès lors, on ne multiplie plus les serveurs de jeux, mais seulement la transmission des informations qui s'y raccordent.

1.3.1 HISTORIQUE DU MODÈLE DE COMMUNICATION « *PUB/SUB* »

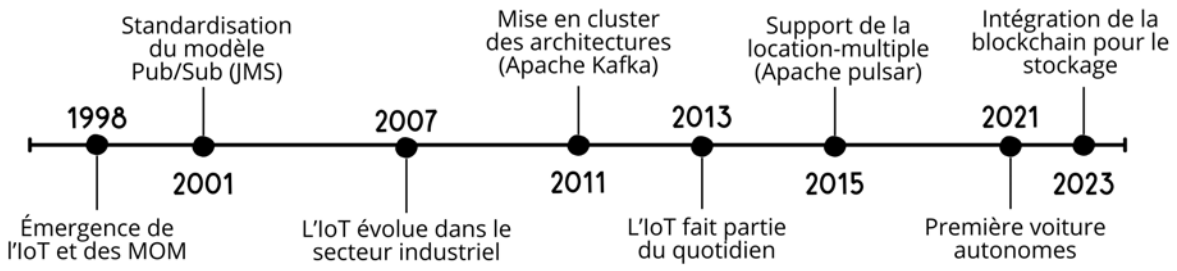

FIGURE 1.6 : L'évolution du modèle « *Pub/Sub* » au fil du temps

Depuis les années 2000, les services en ligne font face à une demande toujours plus forte, ce qui crée un besoin d'extension de leur architecture pour y répondre. En parallèle de ces changements, de nouveaux logiciels et paradigmes de communication font leur apparition pour supporter ce surplus de données. Le paradigme Intergiciels à Message (ou « *Message Oriented Middleware (MOM)* » en anglais) se présente comme une solution majeure, car il permet de résoudre le problème de duplication des messages entre les services et les clients qui les utilisent [24]. C'est avec l'API « *Java Message Service (JMS)* » publié en 2001 que les « MOM » se démocratisent (voir la figure 1.6). Le but de « *JMS* » et des « *MOM* » de manière générale est donc de répondre au besoin des entreprises de conserver leurs services tout en gagnant en élasticité. « *JMS* » définit les standards des communications en environnements distribués en offrant la possibilité de supporter les architectures « *P2P* » et « *Pub/Sub* » [25]. Parmi les fonctionnalités les plus importantes, on retrouve notamment :

Abonnement par sujet et par contenu : Les clients ont la possibilité de s'abonner à un ou plusieurs sujets directement (exemple : « Les joueurs ») ou alors de définir un type de contenu auquel s'abonner (exemple : « Les joueurs dont le niveau est 80 »). Ces deux

méthodes permettent une manipulation efficace et précise des différents sujets d'un réseau.

Abonnement temporaire Les abonnements temporaires sont retirés automatiquement à la fermeture de la connexion avec un client. Par exemple, c'est le cas lorsqu'un joueur quitte le jeu et qu'il est désabonné de tous les éléments dynamiques autour de lui (joueurs, monstres, etc.). À l'inverse, les abonnements permanents sont sauvegardés et seront restaurés à la prochaine connexion du joueur.

Rétention de message La rétention de message est un système qui permet à l'envoyeur de signaler au « *broker* » de garder une copie du message afin qu'il la renvoie aux futurs abonnés. Cela permet de gagner en vitesse et d'économiser des ressources sur les messages entre les envoyeurs et le « *broker* », car les abonnés n'ont plus besoin d'attendre un nouveau message de l'envoyeur pour se mettre à jour.

Persistance de message Les messages persistants, eux, sont sauvegardés de manière à persister aux erreurs fatales du « *broker* » pour pouvoir restaurer l'état du système après le redémarrage. Cette fonctionnalité est particulièrement utile pour, entre autres, restaurer les abonnements entre les erreurs fatales ou les redémarrages.

Péremption des messages Les messages peuvent être dotés d'une durée de validité, passée laquelle la pertinence du message n'est plus assurée et qui indique donc au « *broker* » de ne plus la distribuer. C'est une chose utilisée sur des données changeantes en fonction du temps telle que la météo, le taux de change d'une monnaie, etc.

Après la publication de « *JMS* », les technologies de communication Pub/Sub ont continué d'évoluer pour répondre aux besoins croissants d'élasticité, de performance et de flexibilité dans les systèmes distribués. En 2011, *Apache Kafka* [26] a révolutionné le traitement des flux de données en temps réel en offrant une plateforme distribuée capable de gérer des volumes massifs de données avec une latence minimale. Kafka introduit l'élasticité horizontale en supportant la mise en « *cluster* » des « *brokers* » ce qui permet de séparer les sujets en partitions qui sont elles-mêmes réparties entre les « *brokers* ». De plus, *Kafka* se veut flexible et s'adapte à beaucoup de gestionnaires de données infonuagiques [26]. *Kafka* permet donc la publication, le stockage et la consommation de flux de données, tout en assurant la durabilité et la tolérance aux pannes, ce qui le rend idéal pour les applications nécessitant une haute disponibilité et une résilience élevée.

26. https://kafka.apache.org/, Saisi le 7 juillet 2025

En parallèle, des systèmes comme *Apache Pulsar*[27] (2015) ont émergé, introduisant des fonctionnalités avancées telles que la segmentation de la charge de travail et les architecture multi-tenant. Les architectures multi-tenant suivent un principe de cloisonnement des données qui permet à un logiciel de supporter plusieurs locataires (clients) avec une seule architecture physique et logicielle. Cette fonctionnalité permet à *Pulsar* d'avoir une gestion plus efficace des ressources dans les environnements infonuagique et infonuagique distribués. Pulsar se distingue également par son architecture multi-niveaux qui sépare le traitement des données et la gestion des méta-données de chaque message, offrant ainsi une flexibilité et une élasticité accrues par rapport aux systèmes traditionnels. [27]

Le modèle de communication « *Pub/Sub* » a également assisté à l'essor de l'Internet des Objets (« *IoT* » en anglais). Déjà très implanté dans l'industrie (« *Industrial Internet of Things (IIoT)* ») et la vie professionnelle, « *l'IoT* » connaît une énorme croissance en entrant dans la vie quotidienne [28]. En 2015, avec la généralisation de l'informatique mobile (montres connectées, téléphone intelligent, voiture intelligente, tablette, etc.), le nombre d'appareils connectés s'élève à 15,9 milliards [29]. Pour accompagner ce changement, des solutions modernes, comme « *Message Queuing Telemetry Transport (MQTT)* », ont été optimisées pour des environnements contraints (faible bande passante, consommation d'énergie réduite, etc.), permettant une communication fiable et efficace entre des millions de dispositifs « *IoT* » et les services en lignes [30].

Le contexte d'évolution rapide et massive du monde numérique connecté a poussé les technologies « *Pub/Sub* » à progresser pour devenir des composants essentiels des architectures de systèmes distribués modernes, répondant aux défis d'élasticité, de résilience et de performance. En parallèle, les solutions de réplication des « *MMOGs* » ont accompagné ces changements en adaptant leur fonctionnement sur le besoin des différents « *MMOGs* » en fonction de la quantité de donnée à traiter.

1.3.2 ENVIRONNEMENTS DISTRIBUÉS ET « *PUB/SUB* »

La notion d'environnement distribué désigne un système d'information ou un réseau pour lequel l'ensemble des ressources disponibles ne se trouve pas au même endroit ou sur la même machine [31]. Ce type d'environnement a été mis particulièrement de l'avant avec

27. https://pulsar.apache.org/, Saisi le 7 juillet 2025

l'explosion des « *Software As a Service (SAS)* ». De fait, la multiplication des services en ligne proposés et le besoin de les faire communiquer entre eux a fait se distinguer le modèle « *Pub/Sub* » comme une solution efficiente pour la gestion de la transmission des données dans les environnements hautement distribués [32].

L'informatique en nuage consiste en l'exploitation de la puissance des environnements distribués pour offrir des ressources informatiques comme des serveurs, du stockage, des bases de données et des applications, le tout disponible instantanément via Internet. La dispersion des données sur plusieurs serveurs distants permet une flexibilité, une échelle et une redondance améliorées, facilitant la gestion des données et des applications à grande échelle. Le modèle distribué est essentiel pour réaliser l'élasticité et la complexité qui caractérisent l'informatique en nuage. Le modèle « *Pub/Sub* » est très populaire dans l'informatique en nuage [33] car il permet de manipuler toutes les données nécessaires à la synchronisation des différents serveurs/clients du réseau tout en permettant de définir précisément quel ordinateur a besoin de quelles informations. Ce contrôle permet une efficience accrue du traitement des données en évitant de transmettre des informations inutilisées à des machines, et ainsi éviter une surcharge de traitement.

Tout comme l'informatique en nuage, les « *MMOGs* » ont besoin d'une architecture fortement distribuée pour fonctionner (voir la figure 1.5). Le nombre d'interactions engendrées par les joueurs couplé au nombre d'entités (Personnage Non Joueur (PNJ), Objets, etc.) entraîne un besoin de traitement nécessitant des architectures serveur dites en « *cluster* » [18]. L'ensemble des données du jeu doit donc transiter à travers plusieurs ordinateurs en permanence avant d'être communiqué au client. Le modèle de communication « *Pub/Sub* » représenterait un atout sur le fonctionnement actuel des « *MMOGs* », car il permettrait d'orienter les données entre les différents nœuds de l'architecture, mais aussi d'envoyer des informations plus précises aux joueurs.

1.3.3 APPLICATION THÉORIQUE AUX JEUX VIDÉO

RÉPLICATION ET SUJETS

En informatique, un objet est un conteneur symbolique et autonome qui contient des informations et des mécanismes concernant un sujet, manipulés dans un programme [34]. Les

éléments d'un objet sont structurés, ce qui permet de manipuler directement une partie de celui-ci afin d'effectuer des tâches précises dessus. Ce n'est pas le cas, par exemple, avec les fichiers texte, qui sont constitués de séquences d'octets représentant des caractères, nécessitant une lecture séquentielle pour en déchiffrer le contenu.

Le paradigme objet, largement utilisé dans les « *MMOGs* » et les jeux vidéo en général, s'adapte aisément au modèle de communication « *Pub/Sub* ». En effet, chaque objet du monde qui doit être répliqué est représenté comme un sujet principal, subdivisé en sous-sujets pour chaque détail nécessaire. Les clients ont alors la possibilité de s'abonner soit à l'ensemble, soit à des parties spécifiques de ce groupe de sujets, afin de recevoir les informations pertinentes concernant cet objet.

Par exemple, un sujet principal pourrait être un joueur « X » et les sous-sujets représenteraient les différentes informations de ce joueur (Points de vie, Position dans le monde, etc.). Un tel découpage peut permettre de ne recevoir facilement qu'un certain type de données de la part du « *broker* » concernant un personnage.

La structuration des sujets offre également la possibilité de définir différents niveaux d'accès aux informations. Par exemple, dans les jeux par équipe, il est essentiel de connaître les informations publiques comme la position, le nom et l'apparence de tous les joueurs. Toutefois, l'accès aux informations privées telles que les points de vie, les niveaux et les types d'équipement sont réservés uniquement aux membres de notre propre équipe. Cette organisation permet une communication ciblée et sécurisée au sein des jeux.

GESTION DE L'INTÉRÊT

Un tel modèle de communication nécessite de savoir comment et quand les abonnés s'abonnent à des sujets. On appelle ça la « gestion de l'intérêt » ou « *Interest managment* » en anglais. Il existe plusieurs façons de traiter la gestion de l'intérêt. On peut soit demander aux jeux d'indiquer explicitement au fur et à mesure aux joueurs quels sont les sujets auxquels ils doivent s'abonner, soit extérioriser le processus en donnant la responsabilité à un service extérieur tel qu'un serveur dédié spécifiquement ou aux clients des joueurs directement. Ce découplage entre les émetteurs et les récepteurs permet une distribution des données plus dynamiques et adaptables, essentielles pour les serveurs de jeux qui gèrent des milliers d'interactions simultanées. De plus, le fait qu'un gestionnaire d'intérêt demande aux clients de

suivre ou non de l'information permet de réduire les envois d'information inutilisée, et donc de réduire en partie la charge de travail des serveurs de jeux.

Selon [19], il existe trois façons principales de traiter la gestion de l'intérêt :

Gestion basée sur les objets Dans cette approche, chaque joueur calcule individuellement son intérêt pour des objets basés sur des critères définis. Comme démontré par des systèmes comme *PADRES* [35], qui supportent des requêtes basées sur le contenu, l'avantage de cette méthode réside dans sa précision et sa capacité à s'adapter dynamiquement aux mouvements des joueurs dans l'espace de jeu.

Gestion basée sur les tuiles Le monde du jeu est divisé en tuiles, et les joueurs reçoivent des mises à jour sur des objets situés dans des tuiles qui chevauchent leur zone d'intérêt. Cette méthode est particulièrement efficace dans les environnements où les mouvements des joueurs sont relativement prévisibles ou confinés à des zones spécifiques. Elle est moins coûteuse en termes de calcul que la gestion basée sur les objets, car elle réduit la fréquence des évaluations de l'intérêt à des changements de tuiles plutôt qu'à des déplacements continus.

Gestion basée sur les zones Cette stratégie utilise des structures de données spatiales, comme les arbres de segmentation ou les grilles, pour déterminer rapidement quels joueurs doivent recevoir des mises à jour sur quels objets. L'intérêt est défini par des zones géographiques, et les mises à jour sont envoyées aux joueurs dont les zones d'intérêt se chevauchent avec la localisation des objets.

Bien que fréquemment utilisés, de tels systèmes demandent beaucoup de ressources et sont statiques, ce qui va à l'encontre du besoin d'élasticité des « *MMOGs* ». C'est pourquoi des systèmes conçus pour les architectures hybrides des « *MMOGs* » ont vu le jour. Ces modèles se démarquent notamment par leur approche sans découpage, c'est-à-dire sans zones fixes, contrairement aux méthodes traditionnelles qui divisent l'espace de jeu en régions statiques ou en tuiles [36]. Les zones d'intérêt sont définies dynamiquement et peuvent changer de forme et de taille sans contrainte de maintenir une forme régulière. Cette flexibilité permet une meilleure échelle et une optimisation des échanges de données entre les joueurs, réduisant ainsi la latence et améliorant l'expérience de jeu. De plus, ces solutions tirent avantage de l'aspect « *P2P* » des « *MMOGs* » pour assigner la responsabilité de la gestion de l'intérêt à un groupe de joueurs, ce qui permet au serveur d'économiser des ressources tout en se comportant comme un arbitre sur la gestion faite par les clients des joueurs en question.

1.4 RELATION ENTRE LA BANDE PASSANTE ET LA CONSOMMATION ÉNERGÉTIQUE

Ce mémoire s'articule autour d'une hypothèse fondamentale : la consommation énergétique des équipements réseau (routeurs, commutateurs, serveurs, etc.) est directement influencée par le volume de données qu'ils doivent traiter. Cette hypothèse constitue la pierre angulaire de notre approche visant à améliorer l'efficience des architectures de communication dans les « *MMOGs* ». Elle nous permet d'établir un lien causal entre la réduction du volume de trafic réseau et l'économie d'énergie réalisable, offrant ainsi une perspective concrète pour l'informatique verte appliquée au secteur des jeux vidéo en ligne.

La pertinence de cette hypothèse dans le contexte actuel est considérable. Selon l'Agence internationale de l'énergie, la consommation électrique des centres de données et des infrastructures réseau représentait environ 1 % de la consommation électrique mondiale en 2018, avec une projection de croissance significative pour les années à venir [37]. Les « *MMOGs* », par leur nature, génèrent un volume substantiel de trafic réseau pour maintenir la synchronisation entre les joueurs, contribuant ainsi à cette empreinte énergétique. Si nous parvenons à démontrer qu'une réduction du volume de données traitées entraîne une diminution proportionnelle de la consommation énergétique, nous pourrons alors justifier l'adoption de modèles de communication plus efficients comme le « *Pub/Sub* » pour réduire l'impact environnemental des jeux en ligne.

D'un point de vue théorique, cette relation peut être modélisée en analysant les opérations effectuées par les équipements réseau lors du traitement des paquets de données. Lorsqu'un paquet traverse un appareil réseau, plusieurs étapes de traitement sont nécessaires, chacune consommant des ressources de calcul, et donc de l'énergie. Ces étapes incluent la réception du paquet, l'analyse de son en-tête, la consultation des tables de routage, la détermination du chemin de sortie et la transmission du paquet [38]. Pour formaliser cette relation, considérons qu'un appareil réseau consomme E_{base} watts en état de veille et E_{proc} watts supplémentaires pour le traitement de chaque paquet. Si n paquets sont traités par seconde, la consommation totale E_{total} peut être exprimée par :

$$E_{total} = E_{base} + n \times E_{proc} \tag{1.1}$$

Cette formule, bien que simplifiée, illustre le principe fondamental : plus le nombre de paquets traités est élevé, plus la consommation énergétique augmente. En réalité, cette relation n'est pas strictement linéaire en raison de divers facteurs d'optimisation matérielle comme la mise en cache, le traitement parallèle et les mécanismes d'économie d'énergie adaptatifs. De plus, la taille de chaque paquet réseau peut également affecter la quantité de traitements nécessaires à la prise en charge du paquet par le service (serveur, routeur, etc.). Néanmoins, la tendance générale demeure valide.

En termes de cycles processeur, si un routeur utilise C cycles pour traiter un paquet de taille moyenne, traiter $(n-1)$ paquets au lieu de n permettrait théoriquement d'économiser C cycles. Sur une période étendue, cette économie se traduit par une réduction mesurable de la consommation énergétique. Par exemple, si un routeur standard consomme environ $0,002$ millijoule par cycle processeur et que le traitement d'un paquet nécessite en moyenne 1000 cycles, chaque paquet non traité permettrait d'économiser $0,2$ millijoule. À l'échelle d'un MMOG traitant des millions de paquets par seconde, cette économie devient substantielle.

Cette validation théorique, aussi appuyé par la littérature [39], [40], [41], vérifie notre hypothèse, la consommation énergétique des équipements réseau est effectivement influencée par le volume de données qu'ils traitent. Cette relation, bien que variant en intensité selon les équipements et les contextes, demeure suffisamment significative pour justifier des efforts d'optimisation des flux de données dans les architectures réseau des « *MMOGs* ».

L'optimisation de la communication réseau via des modèles comme le « *Pub/Sub* », qui permettent de filtrer et de cibler précisément les données transmises, pourrait donc représenter une approche pertinente pour réduire l'empreinte énergétique des jeux massivement multijoueurs. En limitant le traitement aux seules données essentielles pour chaque client, ces modèles pourraient permettre des économies d'énergie substantielles à l'échelle des infrastructures mondiales de jeux en ligne, contribuant ainsi aux objectifs plus larges de l'informatique verte.

Cette validation renforce la légitimité de notre travail de recherche, confirmant que l'amélioration de l'efficience du traitement des données dans le cadre des « *MMOGs* » peut effectivement conduire à une réduction mesurable de la consommation énergétique, et par conséquent, de l'impact environnemental de ces applications.

RÉSUMÉ ET TRANSITION

À travers l'analyse des architectures des jeux massivement multijoueur en ligne et leur gestion des très nombreuses interactions entre les joueurs, nous avons pu observer comment ces systèmes façonnent la communication et la synchronisation dans un environnement virtuel dense. Ce prérequis nous guide naturellement vers l'étude approfondie des technologies de « *Pub/Sub* », qui représentent un modèle de communication essentiel pour le traitement efficace des données dans ces environnements hautement distribués. L'état de l'art des modèles de communication nous permettra d'explorer comment le choix de ce modèle peut non seulement améliorer l'élasticité et l'efficience du traitement des données des « *MMOGs* » et donc leur impact environnemental.

CHAPITRE II
ÉTAT DE L'ART

2.1 DÉFINITION DES CRITÈRES

Dans cette section, nous allons introduire les différents critères qui limiteront la portée de notre recherche. Les critères de sélection permettent de définir les travaux qui doivent être étudiés dans le cadre de cette recherche. Si un travail respecte tous les critères de sélection, il est alors considéré comme pertinent et sera étudié dans l'état de l'art. Une fois sélectionné, un travail est ensuite analysé suivant les critères de recherche. Ces critères sont quantifiables et permettent de comparer les différents travaux étudiés dans ce mémoire.

2.1.1 PROTOCOLE DE SÉLECTION

Pour intégrer notre revue de littérature, un article doit répondre à deux critères fondamentaux. Tout d'abord, il doit aborder une solution de réplication conçue spécifiquement pour les « *MMOGs* ». Ensuite, l'article doit avoir été publié de 2011 à 2024. Cette période a été choisie en raison de la publication *d'Apache Kafka* en 2011, qui a marqué un tournant dans l'introduction des architectures en cluster dans les modèles de communication de type « *Pub/Sub* ».

Pour identifier les publications pertinentes, nous avons établi un processus de sélection établi sur l'utilisation de mots-clés. Les termes recherchés incluent « *network architecture* », « *replication* », ainsi que « *MMOG* » et ses variantes telles que « *MMORPG* », « *MOG* », « *MMO* », et les désignations complètes de ces acronymes. Les résultats de recherche sont d'abord filtrés par date. Nous procédons ensuite à l'examen du résumé de chaque article pour vérifier la présence d'une discussion sur les solutions de réplication. Si le résumé confirme cette orientation, l'introduction et la conclusion de l'article sont analysées afin de déterminer si le contexte de l'étude est adéquatement relié à un « *MMOG* ». Chaque travail validé selon ce processus est ensuite examiné plus en détail en fonction des critères de recherche présentés ci-dessous.

2.1.2 LES CRITÈRES DE RECHERCHES

Dans un premier temps, le **modèle de communication** est un facteur majeur dans l'étude d'un travail. En effet, il impose des besoins techniques qui vont influencer toute la structure et le fonctionnement de la solution. Nous allons constater le modèle de communication de manière naïve pour pouvoir par la suite faire des corrélations entre le modèle employé et les autres mesures effectuées.

Dans un second temps, la **bande passante** est une mesure représentant un volume de données émis ou reçu sur une période donnée. C'est un chiffre très important, car il permet de déterminer la charge de travail du serveur (ou du client). Pour la calculer, on multipliera la taille de chaque type de paquets émis par leur nombre d'émissions, puis on divisera par la durée totale de l'émission. Soit Bw la bande passante, i un type de paquet envoyé, S_i la taille en bits du paquet de type i, V_i le nombre de paquets i envoyés, et T la durée de l'émission en secondes.

$$Bw = \frac{\sum_1^i S_i \times V_i}{T} \tag{2.1}$$

En troisième lieu, nous nous intéresserons à la **latence**. Dépendamment de l'architecture, il existe plusieurs façons de la mesurer.

« *C/S* » Dans le cas d'une architecture client/serveur, la latence désigne la somme du temps d'émission de la requête du client vers le serveur (T_{envoie}), du temps de traitement de la requête sur le serveur (T_{calcul}) et du temps de retour de la requête (T_{retour}).

$$L_{C/S} = T_{envoie} + T_{calcul} + T_{retour} \tag{2.2}$$

« *Pub/Sub* » Dans le cas d'une architecture « *Pub/Sub* » la latence désigne la somme du temps d'émission de la requête du client vers le « *broker* » (T_{envoie}), du temps de traitement du « *broker* » ($T_{routing}$), du temps d'envoi entre le « *broker* » et le serveur ($T_{propagation}$), du temps de traitement de la requête sur le serveur (T_{calcul}), du temps de retour de la requête vers le « *broker* » ($T_{propagation}$), du temps de traitement du « *broker* » ($T_{routing}$) et enfin du retour du message entre le « *broker* » et le client (T_{retour}). Dans le cadre d'une architecture « *Pub/Sub* » on considère que $T_{routing}$ est négligeable, car le « *broker* » se contente de faire le travail d'un appareil de routage de paquets.

$$L_{Pub/Sub} = T_{envoie} + 2 \times (T_{routing} + T_{propagation}) + T_{calcule} + T_{retour} \tag{2.3}$$

« *P2P* » Dans le cas d'une architecture pair à pair, la latence désigne la somme des latences d'émission, de routage, de réception de chaque pair que le message doit traverser pour obtenir une réponse, plus le temps de traitement du pair ayant autorité sur la requête. Soit i le nombre de pairs que doit traverser la requête pour trouver une réponse.

$$L_{P2P} = \sum_{1}^{i} T_{envoie_i} \times 2i + T_{routage_i} \times 2(i-1) + T_{calcul} \tag{2.4}$$

Concernant le joueur, la latence pourrait également contenir la période de traitement locale de la réponse jusqu'à perception de la mise à jour du client (graphismes, sons, retours haptiques) mais ce facteur est hors de contrôle du jeu et dépend de trop de facteurs imprévisibles (configuration de la machine client, utilisation de logiciel en simultané, problèmes techniques, etc.). Afin de garantir un confort de jeu et une bonne immersion, il est primordial pour les « *MMOGs* » d'avoir une latence la plus faible possible afin de prévenir les ralentissements ainsi que les coupures de jeu [6]. Pour la suite de ce document et faciliter l'analyse des documents de l'état de l'art : est considéré comme latence acceptable, jouable toutes les latences inférieures ou égales à 100 ms. De même, on calculera la latence comme présenté dans la liste ci-dessus pour cibler uniquement la performance des architectures serveurs et des solutions de réplication. Pour calculer la latence, nous identifierons les paquets émis afin de pouvoir calculer localement la différence de temps entre l'émission d'un message et la réponse qui lui est associé.

Un facteur important lors de la réalisation des expériences est la mise en place d'une perte de paquet fictive permettant de simuler le comportement du protocole réseau utilisé. Le protocole « *TCP* » a la faculté d'être fiable à 100 %, ce qui n'est pas le cas de « d'*UDP* ». Avec l'implémentation de la retransmission des paquets perdus « ***Reliable User Datagram Protocol (RUDP)*** », la fiabilité des protocoles peut approcher les 99 % [42]. Pour la suite du document, l'expression « paquet perdu » désignera l'ensemble des paquets d'information envoyés qui n'ont n'ont pas été reçus, y compris après leur retransmission (« *TCP* »/« *RUDP* »). Ainsi, le seuil de paquet perdu jugé comme reflétant une simulation proche du réel se situe à 2 % minimum pour les solutions établi sur « *UDP* » et supérieur ou égale à 0 % pour les solutions établi sur « *TCP* ».

Enfin, notre dernier critère est **l'élasticité** de la solution. L'élasticité désigne la capacité (ou non), d'une solution à s'adapter à des volumes variables de joueurs. Elle considère la capacité à adapter de manière efficiente et dynamique les ressources à allouer pour répondre

aux variations de la demande. Par exemple, une solution qui nécessite un serveur pour 100 joueurs et quatre serveurs pour 1000 joueurs et une solution à forte élasticité. À l'inverse, une solution qui nécessite un serveur pour 100 joueurs et 15 serveurs pour 1000 joueurs est une solution à faible élasticité. On dit d'une solution qu'elle atteint sa limite élastique lorsque l'ajout de ressources ne permet plus d'augmenter la capacité de joueurs dans des conditions de jouabilité établies (exemple : 80 ms de latence et 200 ko de bande passante). Ce critère est difficilement mesurable sans devoir réimplémenter chaque prototype étudié dans le cadre de cet état de l'art. C'est pourquoi nous nous contenterons de relever les facteurs favorisant l'élasticité d'une solution établie sur les documents et les analyses déjà existantes.

2.2 PRÉSENTATION DES ARTICLES

2.2.1 *FREEMMG 2*

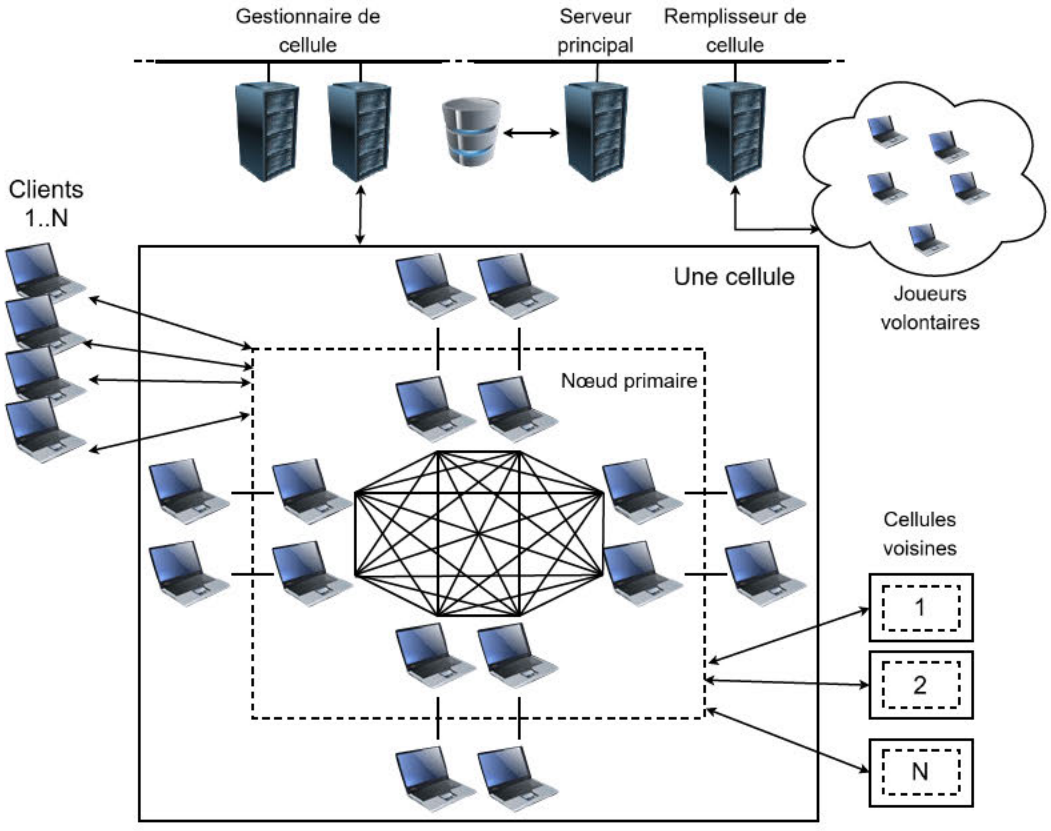

FIGURE 2.1 : Aperçu global d'un réseau *FreeMMG 2*

Les travaux de Fábio Rei Cecin présentent une architecture mêlant les modèles « *C/S* » et « *P2P* » qui vise à obtenir l'élasticité liée aux ressources des clients tout en conservant un haut niveau de sécurité contre la triche [43]. L'architecture *FreeMMG 2* suggère de créer des regroupements de joueurs en cellules de taille variable (NCELL) qui auraient la charge de synchroniser tous les joueurs connectés à celle-ci à la manière d'un réseau pair à pair « multiarbitres ». Comme le montre la figure 2.1, la cellule est composée de n pairs qui se partagent la responsabilité de la gestion du monde du jeu. On les appelle les nœuds primaires. La répétition des calculs permet de minimiser la possibilité pour un nœud primaire de falsifier les résultats d'une opération. Chaque nœud primaire est associé à un nœud secondaire qui a pour rôle de conserver en tout temps une copie des décisions du nœud auquel il est couplé afin de pouvoir prendre sa place en cas de perte de connexion (volontaire ou non). Cet ensemble, qui forme une cellule, est ensuite couplé, d'un côté, à un serveur qui veille au bon fonctionnement de celle-ci, et de l'autre, à de multiples joueurs. Dans cette solution, un client peut devenir un nœud de cellules en exécutant un processus (externe au jeu) qui contrôle toutes les actions liées au réseau. Un serveur externe est nécessaire pour recenser les volontaires pour devenir des nœuds de cellules. Enfin, un dernier serveur se charge de garder une copie de l'état du monde au complet. Ce serveur s'actualise en compilant les informations reçues de la part de tous les serveurs gérant des cellules. C'est l'autorité la plus haute sur l'état du jeu.

FreeMMG 2 présente une solution qui sépare les fonctionnalités liées au support du jeu sur le réseau et les clients de jeux eux-mêmes. Les expériences menées par Fábio Rei Cecin révèlent que la solution serait coûteuse en bande passante, en nécessitant en moyenne 43 Mb/s pour un NCELL de 12. Le rapport de *Cable.co.uk* de 2024 [28] indique une nette amélioration de la bande passante moyenne par foyer dans les Pays du Nord. C'est le cas au Canada par exemple [29]. Cependant, malgré cette évolution, la bande passante requise pour utiliser cette solution exclurait la possibilité à plus de 50 % des pays du monde de pouvoir se connecter sans ressentir des ralentissements notables. Outre le volume de données lié au support du réseau du jeu, s'ajoute les données du joueur lui-même lorsqu'il souhaite jouer tout en faisant partie du réseau de cellules. Cette forte demande de bande passante rend impossible à une grande partie de joueurs potentiels de devenir volontaires pour supporter le réseau de cellules. Une seconde faiblesse de *FreeMMG 2* est sa capacité à laisser les joueurs choisir si oui ou non, ils souhaitent

28. https://bestbroadbanddeals.co.uk/broadband/speed/worldwide-speed-league/, Saisi le 7 juillet 2025

29. https://www.canada.ca/fr/innovation-sciences-developpement-economique/nouvelles/2023/12/le-gouvernement-du-canada-est-en-voie-de-depasser-ses-objectifs-en-matiere-de-connectivite-internet-haute-vitesse.html, Saisi le 7 juillet 2025

aider le réseau de cellule. Cette possibilité, couplée à la quantité de ressources nécessaires pour devenir un nœud de cellule, renforce encore une fois la diminution du nombre de pairs potentiels sur le réseau. En cas d'un trop faible nombre de volontaires, la solution prévoit d'utiliser les ressources disponibles chez tous les joueurs, ce qui réduira considérablement l'expérience de jeux chez ceux dont la configuration est moins performante.

Le protocole expérimental mis en place pour tester *FreeMMG 2*, bien que centré sur l'usage de la bande passante, est adéquat, car il prévoit une perte de paquet de 5 %. *FreeMMG 2* semble être très efficace en procurant une latence variant de 20 à 30 ms, ce qui est bien inférieur à la latence acceptable pour une jouabilité optimale. Cependant, cette mesure a été relevée sur une période sur laquelle le nombre de joueurs ne varie pas. Cette mesure reflète donc le comportement nominal de la solution, mais ne témoigne pas de son comportement lors de la restructuration du réseau de cellules. De même, le protocole ne simule pas les situations exceptionnelles comme le plantage d'un nœud primaire d'une cellule ou la fusion de cellules entre elles.

Enfin, concernant l'élasticité, *FreeMMG 2* délègue toutes les tâches qui ne touchent pas à la sécurité aux nœuds primaires de chaque cellule. Chaque nœud réalisant une première vérification de sécurité, qui est plus ou moins coûteuse dépendamment de la taille de la cellule, cela permet d'accroître facilement le nombre de cellules de l'architecture sans pour autant augmenter la charge de travail du serveur. Le nombre de pairs par cellules est configurable, ce qui permet également de rendre plus ou moins gourmand en ressource le fonctionnement d'une cellule. Les cellules n'échangeant qu'avec un nombre limité de serveurs (ou cellules) durant leur durée de vie, cette solution a une forte élasticité.

2.2.2 *NAHPALM*

Les travaux de Christopher James Carter présentent également une architecture hybride « *P2P* » et « *C/S* » à la manière de *FreeMMG 2*, mais en ajoutant une notion de répartition des tâches au sein de chaque cellule [44] [45]. Intitulée *NAHPALM*, cette architecture est une solution hybride qui confie la sécurité à un serveur en ligne. Ce serveur est utilisé pour encadrer l'authentification des joueurs ainsi que pour aiguiller les joueurs parmi les différentes cellules « *P2P* » existantes. Lorsqu'un client se connecte au jeu, il rejoint une simulation du monde du jeu dans lequel il se trouve. Cette simulation peut être existante, auquel cas, il

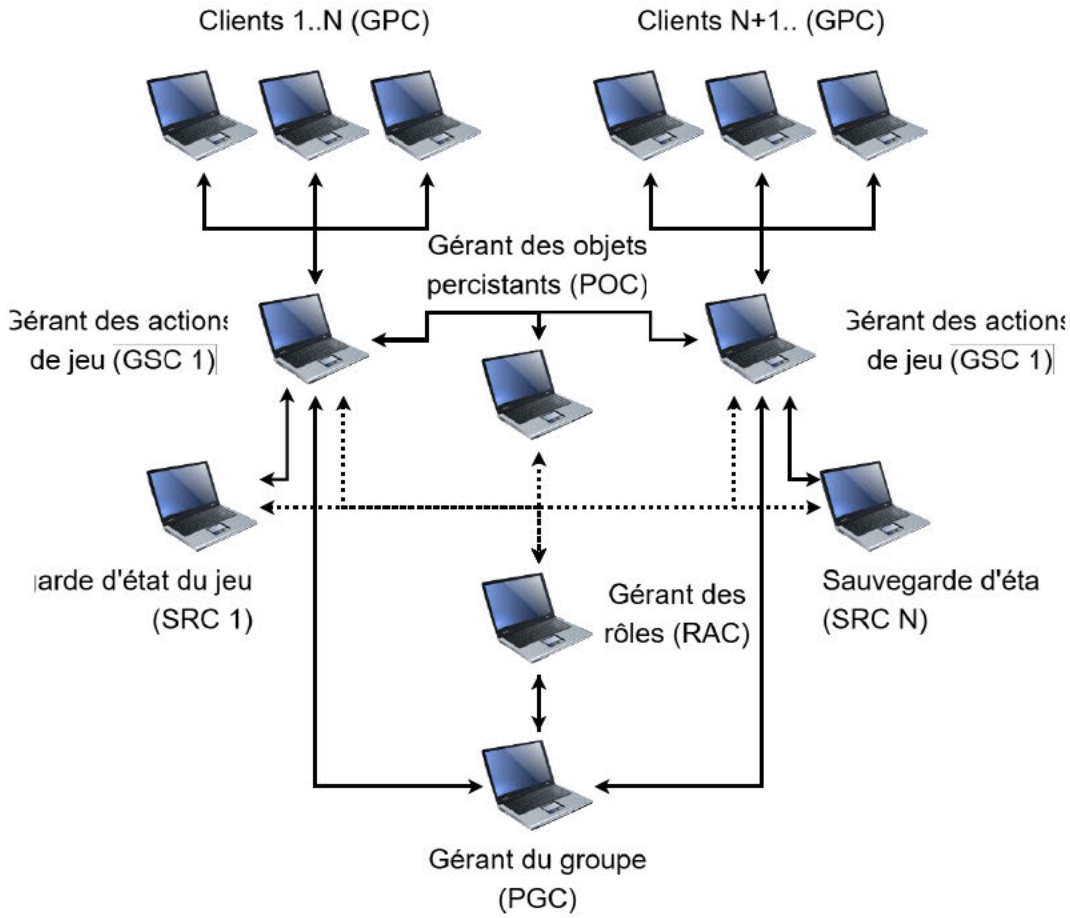

FIGURE 2.2 : Aperçu d'un groupe de pairs *NAHPALM*

est ajouté au groupe de pairs (cellule, voire la figure 2.2) qui maintiennent la simulation. Si aucune simulation n'existe ou que toutes les simulations sont complètes, alors une nouvelle est créée et le nouveau joueur obtient tous les rôles de gestion de manière provisoire (en attendant d'avoir d'autres pairs avec lui). *NAHPALM* est capable de supporter les connexions et déconnexions (volontaires ou non) des pairs avec un système de rôle dynamique qui permet à un nœud de changer de rôle de manière discrète au sein de sa cellule.

Pour encadrer les permissions et distinguer les tâches de chaque pair au sein d'une cellule, l'architecture *NAHPALM* distribue six rôles distincts :

Gérant de groupe (PGC) Le rôle PGC n'est attribué qu'une seule fois et représente l'autorité du groupe de pairs. Le pair qui détient ce rôle se doit d'imposer et de maintenir

l'architecture P2P du groupe en fonction du « *gameplay* » associé au groupe. C'est également lui qui détermine la zone d'intérêt.

Gérant des actions de jeu (GSC) Le GSC est le client responsable de propager les événements liés au jeu entre tous les autres pairs tout en spécifiant le pair à l'origine du changement. Un groupe peut contenir un ou plusieurs GSC en fonction de l'architecture du groupe.

Sauvegarde d'état (SRC) Le SRC est couplé à un GSC et sert à garder une sauvegarde de toutes ses décisions. Le SRC est responsable de détecter la disparition (volontaire ou non) de son GSC. S'il détecte l'absence de son GSC, il prend sa place et le RAC lui donne alors un nouveau SRC pour l'aider.

Gestionnaire des rôles (RAC) Le RAC vérifie que l'ensemble des rôles est distribué dans un groupe de pairs. Il est habilité à changer/transférer un rôle d'un client à l'autre en cas de besoin.

Gestionnaire des objets persistents (POC) Le POC a la responsabilité de garder une trace de toutes les modifications apportées aux objets du monde et communique avec le serveur distant pour l'informer du résultat des actions du groupe sur le monde du jeu. Le POC est soumis aux décisions du GSC concernant les modifications à approuver ou non.

Client simple (GPC) Le GPC est un client joueur. Il rend compte de toutes ses actions à un GSC qui lui répond avec le nouvel état du monde de jeu.

Dans le document d'analyse de la solution [45], le protocole annoncé introduit que la solution a été testée dans un réseau local (« *Local Area Network (LAN)* ») avec un total de huit joueurs connectés. Malgré la précision sur l'utilisation d'un logiciel de simulation de réseaux en ligne, l'ensemble des mesures de l'expérience ne sont pas représentatives d'une simulation d'un « *MMOG* ». Le matériel utilisé pour l'expérimentation ne représente pas les machines disponibles en 2014 [30]. Compte tenu du cadre de l'expérience, une implémentation complète de la solution sur du matériel cohérent serait nécessaire pour effectuer une analyse complète et juste de cette solution.

En partant du principe que la solution a une élasticité parfaite (que le nombre de ressources nécessaires est parfaitement proportionnel au nombre d'entités/joueurs), on pourrait

30. https://web.archive.org/web/20140629123257/http://store.steampowered.com/hwsurvey, Saisi le 7 juillet 2025

estimer que la bande passante de *NAHPALM* pour 100 joueurs est de 2.67 Mb/s, ce qui est largement acceptable pour la plupart de la connexion internet du monde [31].

Du côté de la bande passante, la mesure a été prise sur un réseau local. Elle ne reflète donc pas le comportement potentiel d'un réseau en ligne. En prenant $c = 1\,079\,252\,848,8 km/h$, la vitesse de la lumière dans le vide [32], et en imaginant une transmission de l'information dans une fibre optique parfaite (sans perturbation : routeur, congestion, etc.), il faudrait être à moins de 18 km du serveur pour espérer obtenir un résultat similaire. Pour rendre légitime une mesure de la latence sur cette solution, il faudrait adapter le protocole de test en décentralisant le serveur afin de pouvoir faire des mesures à travers le réseau.

Enfin, concernant l'élasticité, *NAHPALM* partitionne l'ensemble des différentes tâches du réseau dans différents rôles qui sont eux-mêmes regroupés en cellules. Cette structure permet d'encadrer facilement les différentes ressources et leur quantité en fonction du besoin. Cette possibilité garantie qu'un groupe de pairs est sûr de fonctionner dans sa meilleure configuration en offrant toujours les rôles clés aux pairs ayant les meilleures prédispositions. *NAHPALM* permet également une élasticité en termes de nombre de pairs associés à un rôle, ce qui propose une nouvelle possibilité pour contrôler au plus proche du besoin les ressources allouées dans l'architecture du jeu.

2.2.3 « *SUPERNODE ARCHITECTURE FOR SCALABLE MMOGS* »

Les travaux de Sergii Shepelenko proposent, eux aussi, une architecture hybride « *P2P* » et « *C/S* » en remplaçant la notion de cellule autoritative par un nœud unique dans le but de réduire les coûts liés à l'infrastructure [46] et simplifier l'élasticité.

L'architecture de Sergii Shepelenko est établie sur un maillage « *P2P* », dans lequel chaque joueur est appelé un nœud, ainsi qu'un serveur central (voir la figure 2.3). Afin de réduire le plus possible les tâches du serveur central, toutes les tâches liées à la jouabilité (position, interactions, etc.) sont délayées au niveau de certains joueurs répartis sur le maillage.

---

31. https://www.cable.co.uk/broadband/worldwide-speed-league/2024/worldwide_speed_league_data.xlsx, Saisi le 7 juillet 2025

32. https://web.archive.org/web/20170814094625/http://www.bipm.org/utils/common/pdf/si_brochure_8_en.pdf, Saisi le 7 juillet 2025

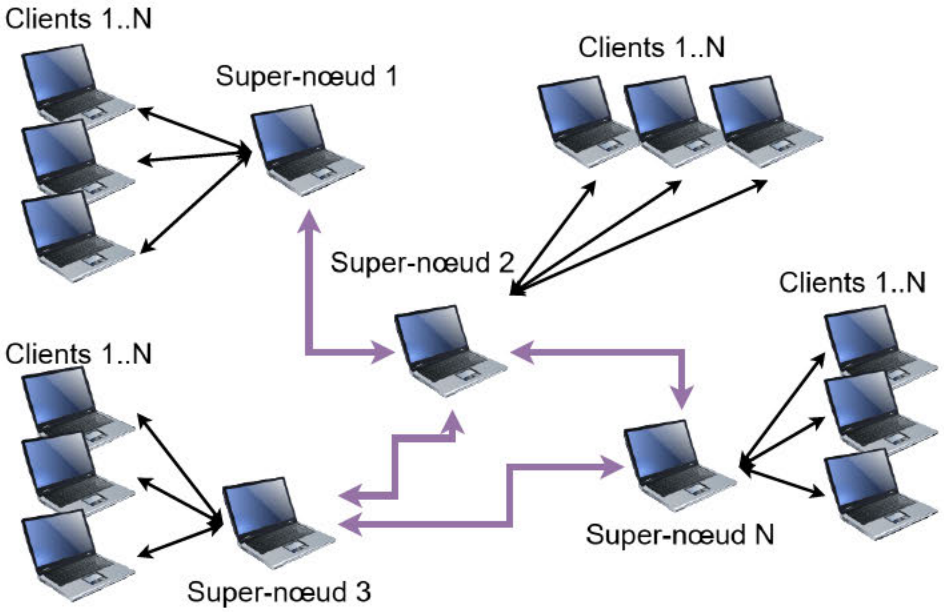

FIGURE 2.3 : Aperçu d'un maillage « *P2P* » de super-nœuds

Ces joueurs sont appelés des super-nœuds. Chaque super-nœud a en charge une région restreinte du monde dans lequel il assure la cohérence du jeu ainsi que la gestion et transmission des informations de ses joueurs aux autres super-nœuds (ou au serveur). Les super-nœuds ont également à leur charge de signaler toutes modifications du maillage au serveur pour assurer son évolution dynamique dans le temps. Par exemple, si un joueur est désigné « super-nœuds » d'une région et qu'il se déconnecte, le super-nœud déclenche un changement de rôle avant sa déconnexion pour attribuer la responsabilité de la zone de jeu à un autre joueur présent dans la zone (s'il y en a un, sinon la zone est simplement ignorée). En permettant aux super-nœuds du maillage de signaler leur état et de demander à changer la structure du maillage, la solution garantit que chaque client fonctionne de manière optimale, et ainsi propose une meilleure répartition de la charge. Pour aider les clients, le serveur veille constamment à la bonne répartition maillage et peut, en cas de déconnexion involontaire (plantage) d'un super-nœud, en attribuer un nouveau au plus vite afin de réduire la période de perturbation des joueurs de la zone.

De son côté, le serveur conserve les rôles liés à la sécurité des données telles que l'authentification, la conservation de la copie du monde (dites autoritaire), le suivi des transactions monétaires, la résolution de conflit entre super-nœuds, entre autres.

L'architecture proposée ne présente pas de données concrètes sur l'utilisation de la bande passante, ce qui constitue un point faible de la solution. Partant du principe que chaque super-nœud est responsable de la gestion de plusieurs joueurs et qu'il doit synchroniser les données de sa zone avec les autres super-nœuds. On peut supposer que l'utilisation de la bande passante pourrait devenir conséquente, en particulier lorsque le nombre de super-nœuds ou de joueurs augmente. L'absence de mesures sur ce critère empêche de déterminer si la solution est viable chez des clients chez lesquels la bande passante est limitée.

L'un des points forts de cette architecture est sa faible latence. Les tests montrent une latence de 10 ms pour un super-nœud gérant jusqu'à 25 joueurs, ce qui est excellent pour maintenir une expérience de jeu fluide. Cette faible latence s'explique par la proximité des joueurs au super-nœud et l'utilisation de communications directes pour les échanges à l'intérieur d'une même zone. Cependant, la latence pourrait varier d'une action à l'autre dans le cadre d'interactions avec des éléments hors de la zone de jeu du super-nœud du client, car ces actions nécessiteraient des synchronisations supplémentaires pour être effectuées.

Concernant l'élasticité de cette solution, elle est apportée par la gestion dynamique des super-nœuds. Lorsqu'un super-nœud quitte le réseau, il signale son changement de rôle et un nouveau super-nœud est désigné automatiquement, parmi les autres joueurs de la zone, par le serveur, assurant alors la continuité de la gestion du monde dans la zone. Cette approche permet également un suivi en temps réel de la charge du maillage grâce aux échanges entre les super-nœuds et le serveur central. Ainsi, le serveur peut décider d'allouer plus ou moins de ressources pour constituer le réseau de pairs afin de garantir une utilisation optimale des ressources tout en garantissant également une qualité de service aux super-nœuds en évitant leur surcharge. Cependant, l'élasticité de la solution dépend en grande partie de la disponibilité et de la stabilité des super-nœuds, qui doivent être capables de diriger plusieurs joueurs sans interruption et sans réduire les performances du jeu (pour lui et les joueurs de sa zone).

2.2.4 *« PEER-TO-PEER NETWORK ARCHITECTURE FOR MASSIVE ONLINE GAMING »*

Les travaux de Bongani Shongwe proposent une architecture « *P2P* » visant à résoudre les limitations élastiques des architectures « *C/S* » grâce à une architecture 100 % distribuée. L'objectif principal de la solution est de réduire la latence des messages et d'améliorer l'effica-

cité globale du traitement des interactions entre les joueurs tout en supprimant l'infrastructure serveur coûteuse habituellement employée dans les architectures « *C/S* » et hybrides [47].

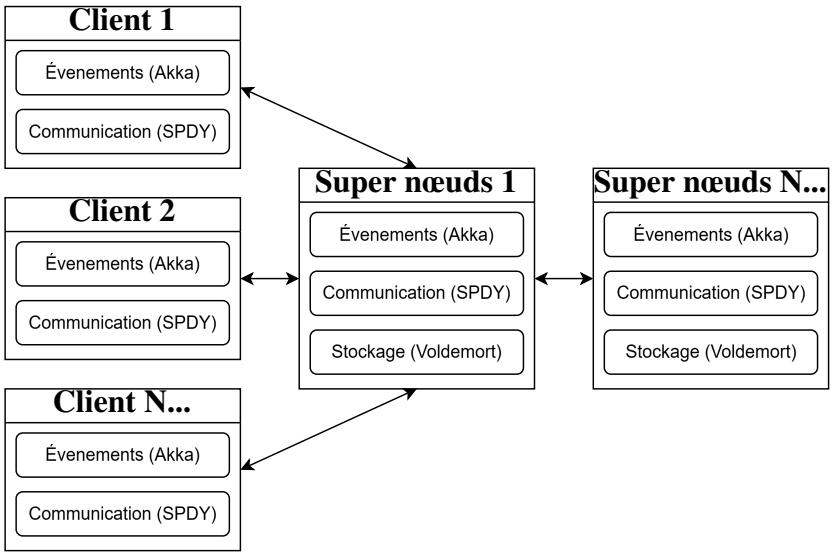

FIGURE 2.4 : Aperçu du maillage « *P2P* » de super-nœuds de Bongani Shongwe

La solution présentée par Bongani Shongwe repose sur trois composants principaux permettant d'atteindre les objectifs visés : une communication optimisée, un traitement événementiel concurrent et un stockage distribué. La couche de communication s'appuie sur le protocole expérimental *SPDY* de *Google*, fonctionnant au-dessus du « *TCP* », et permettant d'accélérer les échanges réseau en multiplexant simultanément plusieurs flux de données sur une seule connexion. Ce protocole permet ainsi une réduction significative de la latence réseau et une meilleure utilisation des ressources disponibles.

Pour le traitement des interactions entre joueurs, l'architecture implémente une couche applicative événementielle basée sur le modèle acteur grâce au cadriciel *Akka* [33]. Ce choix technique permet un traitement parallèle et asynchrone efficace des messages échangés entre les pairs, tout en assurant une excellente scalabilité horizontale lorsque le nombre de joueurs augmente.

L'architecture se base exclusivement sur un maillage « *P2P* » formé sur le modèle « super-nœuds », conçu pour s'adapter dynamiquement à la charge des joueurs et aux besoins

33. https://akka.io/what-is-akka, Saisi le 7 juillet 2025

du jeu. À l'instar de *NAHPALM*, chaque super-nœud a pour rôle de garder une portion du monde synchronisée en ayant l'autorité sur ce qui s'y passe (voir la figure 2.4). Contrairement aux autres solutions étudiées précédemment, où la réplication de l'état du monde dépend uniquement de la position du joueur, ici la gestion d'intérêt prend également en compte explicitement les besoins du joueur. Ce fonctionnement s'appuie sur le modèle « *Pub/Sub* » qui permet à chaque nœud de s'abonner aux informations dont il a besoin, réduisant ainsi la quantité de données échangées au strict nécessaire. La détermination précise et dynamique des abonnements reste cependant à la charge des développeurs du jeu, les travaux de Bongani Shongwe ne proposant pas d'algorithme générique à ce niveau.

Enfin, pour garantir l'accessibilité et la résilience des données de jeu, la solution utilise le projet *Voldemort* [34]. Ce système de stockage distribué en « clé-valeur », introduit par LinkedIn en 2009, permet de constituer une base de données non relationnelle capable de supporter plus de dix mille opérations par seconde. L'avantage principal de ce système est sa répartition et son partitionnement automatique entre les différents nœuds, assurant ainsi une haute disponibilité et une tolérance aux pannes sans nécessiter que chaque joueur possède une copie intégrale de l'état du jeu sur sa propre machine.

Bongani Shongwe présente une solution qui se base uniquement sur un maillage « *P2P* » ce qui pose un problème concernant la sécurité à plusieurs niveaux. Dans un premier temps, le pouvoir des joueurs désignés comme étant des super-nœuds est décidé par un algorithme intégré au jeu lui-même et donc est potentiellement manipulable à des fins de triche. Dans un second temps, l'absence de serveur de secours pour conserver une sauvegarde des données du jeu permet la perte de l'état du monde au cas où tous les joueurs viendraient à être déconnectés. Ce manque de serveur rend également complexe la connexion des nouveaux joueurs qui ne peuvent pas connaitre à l'avance les adresses des super-nœuds lors de la connexion au maillage « *P2P* » (au jeu de manière général).

La solution, publiée en 2014, se fonde sur *SPDY*. En 2015, *Google* annonçait la fin du support de *SPDY* dès la sortie de *Chromium 51* [35] au profit du nouveau protocole *HTTP/2* qui avait les mêmes objectifs et de meilleurs résultats. Suivie par *Cloudflare* en 2018 [36] et *Apple* en

34. www.project-voldemort.com, Saisi le 7 juillet 2025
35. https://blog.chromium.org/2015/02/hello-http2-goodbye-spdy.html, Saisi le 7 juillet 2025
36. https://blog.cloudflare.com/deprecating-spdy/, Saisi le 7 juillet 2025

2019 [37], aujourd'hui le protocole n'est plus supporté par l'immense majorité des navigateurs modernes [38]. Il faudrait donc réimplémenter la solution avec *HTTP/2* afin d'avoir des résultats plus récents, et ainsi pouvoir faire des comparaisons viables.

Pour tester sa solution, Bongani Shongwe a mis en place plusieurs tests séparés afin d'isoler la performance de chaque partie de la solution. Toutes les expériences ont été conduites dans l'objectif de déterminer le temps de transmission de l'information. L'omission du volume de donnée requis pour utiliser la solution aux niveaux des super-nœuds nous empêche de déterminer si oui ou non cette solution serait viable pour le grand public. Du point de vue de la latence, on peut la calculer en faisant la somme des latences observées. Dans la figure 6.4 du document, on observe une latence de 158 ms pour 100 joueurs et 1 route de messagerie contre 51 ms pour 100 joueurs et 100 routes de messagerie. Même s'il est certain que la condition optimale de la situation se trouve lorsque le nombre de routes de messageries est plus grand que le nombre d'utilisateurs, la figure 6.5 montre une élasticité forte de la solution lorsque le nombre de routes est élevé. Cette élasticité s'explique par la division de la charge de travail du traitement messages sur un nombre plus élevé de super-nœuds, permettant ainsi de réduire les chances de congestion de la messagerie.

2.2.5 « *PUBLISH/SUBSCRIBE NETWORK DESIGNS FOR MULTIPLAYER GAMES* »

Les travaux de César Cañas et ses collègues s'intéressent aux prédispositions des « *MMOGs* » vis-à-vis des architectures distribuées. Ils remarquent que le comportement de ce type de jeux s'adapte particulièrement bien à ce genre d'architecture, en raison du nombre élevé d'actions effectuées et de leur nature indépendante. À partir de ce constat, ils proposent trois moteurs réseau fondés sur le modèle de communication Pub/Sub. Contrairement aux autres travaux étudiés jusqu'ici dans ce mémoire, l'article ne cherche pas à améliorer les performances avec une architecture physique spécifique, mais au contraire, il s'appuie sur des solutions logicielles (logiques) pour déterminer ensuite l'architecture physique nécessaire.

L'objectif principal reste d'améliorer l'efficacité de la gestion des données afin de mieux répondre aux défis liés à la scalabilité et à la gestion des interactions entre les joueurs [19]. Leurs propositions sont fondées sur le modèle « *Pub/Sub* » décliné en trois variantes, chacune

37. https://webkit.org/blog/8569/removing-legacy-spdy-protocol-support/, Saisi le 7 juillet 2025

38. https://caniuse.com/?search=SPDY, Saisi le 7 juillet 2025

mettant en œuvre un système de communication différent. L'intérêt est de montrer les points forts du modèle de communication « *Pub/Sub* » en étudiant son comportement selon trois façons distinctes d'échanger les données. Ces trois approches sont dites « orienté objet », « orientée tuile » et « orienté contenu ».

Pour analyser les solutions proposées dans l'article, Cañas et al. définissent trois principaux cas d'utilisation, utilisés comme critères pour comparer leurs approches et identifier leurs avantages et inconvénients respectifs. Le premier critère est **la découverte des objets**. Il désigne la capacité d'une solution à permettre à un joueur de récupérer les informations d'un nouvel objet avec lequel il n'est pas encore synchronisé. Ce critère inclut également la capacité inverse, à savoir la possibilité d'arrêter la synchronisation avec un objet devenu non pertinent. Le second critère est **le transfert de copie**. Une fois qu'un objet est identifié comme pertinent par le système de gestion des intérêts pour un joueur X, ce joueur X doit pouvoir demander une copie de l'objet au propriétaire afin de récupérer son état actuel dans le jeu. Enfin, le dernier critère est **la propagation de mise à jour**. Il mesure la capacité d'une solution à diffuser les mises à jour d'un objet vers tous les joueurs concernés afin de garantir leur synchronisation.

PRÉSENTATION DE LA SOLUTION ORIENTÉE OBJET

La première variante proposée est la solution orientée objet. Dans celle-ci, chaque objet du monde est représenté par un sujet. Ce sujet est utilisé comme un canal de communication entre le propriétaire de l'objet et l'ensemble des joueurs intéressés par cet objet. Cette solution nécessite un service externe de gestion des intérêts (« *Interest Manager (IM)* »), hébergé par un ou plusieurs serveurs appelés « gestionnaires d'intérêt ». Durant une partie, ces gestionnaires d'intérêt suivent la position de tous les objets en étant automatiquement abonnés à leur sujet respectif, et disposent ainsi d'une copie locale de chaque objet du jeu. La découverte des objets s'effectue donc intégralement sur ces serveurs.

Lorsqu'un serveur « *d'IM* » détecte qu'un objet entre dans la zone d'intérêt d'un autre (et inversement), ce serveur envoie un message aux objets concernés et leur ordonne de réclamer une copie locale de l'objet nouvellement pertinent ; (1) sur la figure 2.5. Pour cela, chacun d'eux s'abonne au sujet de l'autre (2) puis envoie une demande de copie (3). Dès lors qu'un

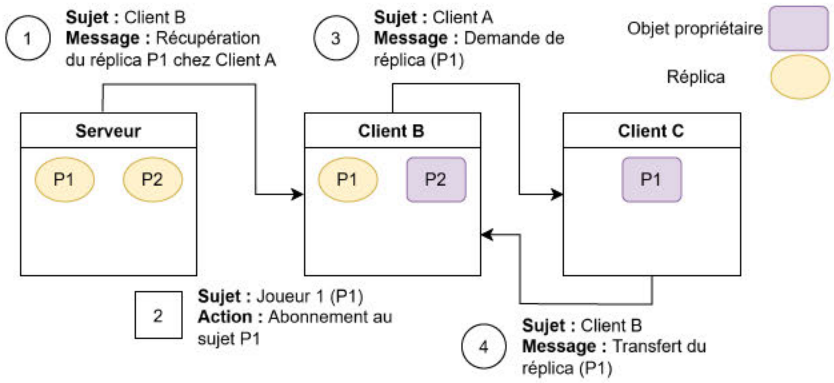

FIGURE 2.5 : Transfert de copie d'objet avec la solution orientée objet

objet reçoit une demande de copie sur son sujet, il publie toutes les informations le concernant directement sur le sujet de l'objet ayant effectué la demande (4). Ainsi, seul le demandeur reçoit le message, évitant ainsi le transfert inutile d'informations vers les autres objets déjà à jour.

Du côté des modifications, lorsqu'un joueur *A* souhaite modifier un objet possédé par un joueur *B*, il doit envoyer une demande de mise à jour au joueur propriétaire sur le sujet associé à l'objet en question ; (1) sur la figure 2.6. Le joueur *B* traite alors la demande de mise à jour puis répond sur le sujet personnel du joueur *A* en confirmant ou en rejetant la demande (2). Si la modification est approuvée, le propriétaire de l'objet publie ensuite le nouvel état de l'objet sur son propre sujet. Ainsi, tous les joueurs intéressés, et donc abonnés à ce sujet, reçoivent immédiatement les informations actualisées concernant l'objet (3).

PRÉSENTATION DE LA SOLUTION ORIENTÉE TUILE

La seconde variante proposée est la solution orientée tuile. Dans celle-ci, le monde du jeu est découpé en plusieurs zones géographiques distinctes appelées « tuiles ». Ces tuiles couvrent une zone fixe du monde du jeu. Elles sont définies lors de la conception du jeu en fonction des objets statiques présents, tels que les murs, les bâtiments, comme sur la figure 2.7 par exemple. En plus des sujets représentant chaque objet du monde, chaque tuile possède trois sujets dédiés : un sujet pour les demandes de copie, un sujet pour les réponses aux demandes de copie, et un sujet pour les notifications de mises à jour. Contrairement à la solution précédente,

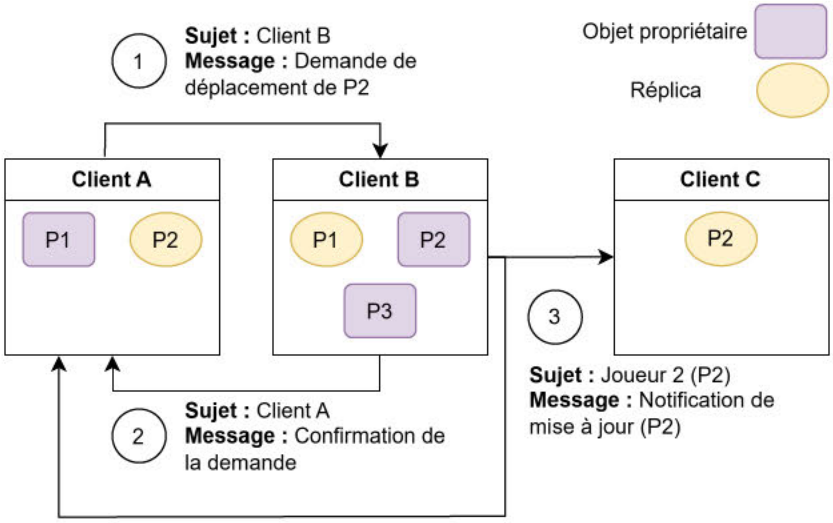

FIGURE 2.6 : Propagation d'une mise à jour d'un objet avec la solution orientée objet

la gestion des intérêts (« *IM* ») ne nécessite aucun serveur externe. En effet, le découpage statique de la zone de jeu en tuiles permet de déterminer à la conception quelles sont les tuiles incluses dans la zone d'intérêt d'un joueur en fonction de la tuile sur laquelle il se trouve actuellement. Ainsi, chaque joueur s'abonne et se désabonne automatiquement aux sujets des tuiles entrantes ou sortantes de sa zone d'intérêt selon ses déplacements.

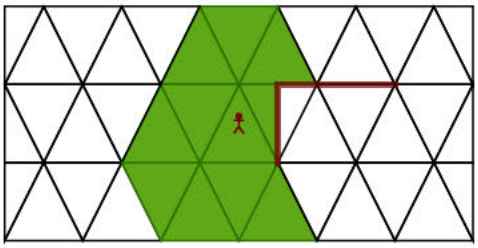

FIGURE 2.7 : Exemple de zone d'intérêt avec la solution orientée tuile

La découverte des objets s'effectue de manière décentralisée. Lorsqu'un joueur entre dans une nouvelle tuile, il s'abonne temporairement au sujet réservé aux réponses de copie de cette tuile. Il publie ensuite une demande générale sur le sujet réservé aux demandes de copie de cette même tuile afin d'obtenir les copies de tous les objets qui s'y trouvent. Tous les joueurs possédant au moins un objet dans cette tuile, et donc abonnés au sujet des demandes de copie, répondent alors en publiant directement les informations concernant leurs objets sur le sujet réservé aux réponses aux demandes de copie. Le joueur nouvellement arrivé récupère

ainsi les copies des objets présents sur la tuile, sans échanges inutiles avec les joueurs déjà synchronisés. À l'inverse, un joueur déjà présent sur une tuile découvre qu'un autre joueur est entré dans sa zone d'intérêt grâce à son abonnement au sujet de notifications de mises à jour des objets de la tuile. Lorsque le joueur A (stationnaire sur la tuile T) reçoit une notification de mise à jour de la part du joueur B (en mouvement), il s'aperçoit qu'il ne dispose pas encore d'une copie locale de l'objet B. L'objet A s'abonne alors temporairement au sujet de réponses de copie de la tuile T et envoie une demande de copie sur le sujet dédié de cette même tuile. La découverte de A et B devient alors complète et réciproque. Dans chacune de ces situations, l'abonnement au sujet des réponses de copie est temporaire. En effet, une fois les répliques reçues, il n'est plus nécessaire de les recevoir de nouveau. En effet, les futures mises à jour seront obtenues directement grâce aux messages diffusés sur le sujet dédié aux notifications de mises à jour. L'abonnement temporaire est donc suivi d'un désabonnement automatique après un délai suffisant défini lors de la conception du jeu.

Concernant les modifications des objets, lorsqu'un joueur A souhaite modifier un objet, il doit transmettre une demande de mise à jour directement au propriétaire de l'objet via le sujet dédié à cet objet. Après traitement de la demande, le propriétaire publie l'état modifié de l'objet sur le sujet de notifications de mises à jour de la tuile concernée. Tous les joueurs abonnés au sujet de cette tuile, et donc intéressés par les changements effectués dans celle-ci, reçoivent alors automatiquement la mise à jour, garantissant ainsi la synchronisation de leurs copies locales. Ce fonctionnement est très proche de celui de la solution orientée objet, à la différence qu'ici la propagation des mises à jour s'effectue par tuile et non par objet. Cette légère différence permet de garantir que seuls les joueurs situés à proximité (et donc réellement intéressés) recevront la mise à jour, tout en permettant à l'ensemble des joueurs d'être abonnés au sujet de l'objet sans nécessairement recevoir toutes les mises à jour. Cette séparation permet d'utiliser le sujet de l'objet comme un canal de contrôle, sur lequel peuvent être partagées des informations pertinentes pour tous les joueurs, indépendamment du gameplay ou de l'aspect visuel. Il pourrait s'agir, par exemple, d'informations sur la latence, ou de données publiques du joueur telles que son nom, son statut en ligne, et cetera.

PRÉSENTATION DE LA SOLUTION ORIENTÉE CONTENU

La troisième variante proposée est la solution orientée contenu. Dans celle-ci, on utilise la possibilité d'envoyer des propriétés avec les messages afin d'apporter des précisions sur le

contexte, et ainsi définir des filtres dynamiques pour le partage d'informations. Cañas et al. remarquent que cette caractéristique peut être exploitée comme une force par les MMOGs, mais que les moteurs réseau basés sur le contenu demandent une puissance de calcul beaucoup plus importante. Pour illustrer leur propos, ils présentent cette troisième solution qui accompagne l'ensemble de ses messages de deux propriétés : x et y, symbolisant les coordonnées de la source du message. Les échanges d'informations entre joueurs ont lieu sur trois sujets communs à l'ensemble du monde du jeu, comme si le monde n'était constitué que d'une seule grande tuile, ainsi qu'à l'aide des sujets dédiés à chaque objet. Les messages étant accompagnés de la position x-y des objets, il n'est plus nécessaire d'avoir des sujets séparés selon des zones distinctes. Ainsi, à l'instar de la solution orientée tuile, cette variante ne nécessite aucun serveur externe de gestion des intérêts (*IM*), la gestion étant intégrée directement dans le fonctionnement même de la propagation des messages au niveau réseau. Dans cette solution, la zone d'intérêt du joueur devient donc un rectangle calculé avec un intervalle à partir de x et y. En opposition à la solution orientée tuile, comme illustré par la figure 2.8, cette méthode ne tient pas compte des obstacles statiques dans le monde du jeu.

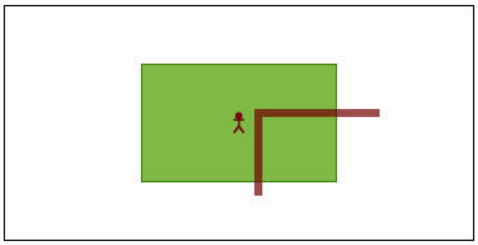

FIGURE 2.8 : Exemple de zone d'intérêt avec la solution orientée contenu

La découverte des objets fonctionne de manière entièrement décentralisée. Chaque joueur s'abonne au sujet de notifications de mises à jour et de demandes de copie en fournissant la position (x, y) de la zone d'intérêt à couvrir. Dans le cas où un joueur A entrerait dans la zone d'intérêt d'un joueur B (immobile), le joueur B reçoit automatiquement une notification de mise à jour concernant l'objet du joueur A grâce à son abonnement au sujet dédié. S'il ne possède pas encore de copie locale de cet objet, le joueur B peut alors s'abonner temporairement au sujet de réponses de copie en passant sa position en paramètre, puis envoyer une demande de copie. Le joueur A, étant déjà intéressé par la zone du joueur B, va recevoir cette demande et répondre en conséquence. Dans la situation inverse, le joueur A (mobile) cherche à découvrir de nouveaux objets dans sa nouvelle zone d'intérêt. Lors de son déplacement, il actualise les

paramètres (x, y) de son abonnement au sujet de notifications de mises à jour et de demandes de copie selon une période définie lors de la conception du jeu. Une période trop faible résulterait en un très grand nombre de synchronisations des zones d'intérêt, tandis qu'une période trop grande pourrait avoir des répercussions négatives sur la jouabilité du jeu (apparition tardive d'objets à l'écran, perte rapide d'informations, etc.). À chaque période, s'il s'est déplacé, le joueur A envoie une demande de copie sur le sujet dédié en fournissant la position de son propre avatar. En indiquant sa propre position dans la demande de copie, le joueur A s'assure que seuls les objets dont la zone d'intérêt inclut la position de A vont recevoir la demande, et ainsi y répondre. Ce mécanisme garantit des échanges efficaces tout en limitant les transmissions inutiles vers les joueurs non concernés. Dans cette solution, une position (x, y) peut correspondre à une infinité de zones d'intérêt différentes qui se superposent. C'est pourquoi, si un objet reçoit plusieurs demandes de copie simultanément, une seule réponse suffit pour répondre à toutes les requêtes reçues.

Concernant les modifications des objets, le fonctionnement est semblable à celui des solutions précédentes, avec toutefois une différence dans la propagation des mises à jour. Lorsqu'un joueur souhaite modifier un objet, il envoie une demande de mise à jour directement au propriétaire de l'objet via le sujet spécifique de cet objet. Après traitement et validation de la demande, le propriétaire diffuse la mise à jour en publiant l'état modifié de l'objet via un message sur le sujet de notifications de mises à jour, en précisant sa propre position (x, y) en paramètre. Ainsi, seuls les joueurs ayant actuellement des abonnements avec des zones d'intérêt incluant la position de l'objet modifié reçoivent ce message. Cette solution garantit une très grande flexibilité : les joueurs reçoivent toujours exactement les informations nécessaires selon les critères définis par leur abonnement dynamique, tout en évitant efficacement les échanges d'informations inutiles.

COMPARAISON DES SOLUTIONS

Les travaux de Cañas et al. [19] se démarquent des autres articles étudiés dans ce mémoire, puisqu'ils ne proposent pas directement une architecture réseau physique, mais plutôt différents moteurs réseau fondés sur le modèle de communication « *Pub/Sub* ». Cette distinction représente un avantage méthodologique, car elle permet de concentrer l'évaluation directement sur la logique d'échange des données, indépendamment des contraintes matérielles précises. Ce choix permet une certaine flexibilité dans le déploiement des solutions, mais laisse

également la question ouverte quant à l'impact concret de ces moteurs sur les infrastructures réseau réelles des jeux massivement multijoueurs.

La solution orientée objet présente une approche simple et intuitive, dans laquelle chaque objet du jeu possède son propre sujet dédié. Cependant, elle ne profite que partiellement des capacités de multidiffusion (« *multicast* ») du modèle « *Pub/Sub* ». En effet, seules les mises à jour des objets sont diffusées via le mécanisme de publication sur le sujet des objets, tandis que les autres communications, comme les demandes de copie, reposent entièrement sur des échanges point-à-point. Cela entraîne une multiplication potentielle du nombre de messages échangés à mesure que les joueurs se déplacent, chaque changement dans les zones d'intérêt entraînant de nouveaux abonnements et désabonnements individuels.

À l'opposé, la solution orientée tuile améliore sensiblement cette situation en exploitant davantage le potentiel du modèle « *Pub/Sub* ». En particulier, elle élimine les échanges point-à-point dans les mécanismes de découverte des objets et de transfert de copies. Chaque demande de copie est diffusée sur un canal dédié à la tuile, permettant ainsi de limiter le nombre de messages envoyés. En effet, un joueur n'a besoin que d'un seul message de demande par tuile nouvellement découverte, même si celle-ci contient plusieurs objets. De plus, en cas de demandes simultanées provenant de plusieurs joueurs sur une même tuile, chaque objet ne répond qu'une seule fois en diffusant sa copie sur le canal correspondant, ce qui réduit significativement le nombre global de messages échangés et améliore l'efficacité de la bande passante. Toutefois, cette solution nécessite une préparation statique préalable du découpage en tuiles, ce qui limite potentiellement son adaptabilité face à des changements fréquents de la carte du jeu.

Enfin, la solution orientée contenu reprend une structure conceptuelle semblable à la solution orientée tuile. Cependant, elle définit les zones d'intérêt de manière dynamique et continue à l'aide de propriétés géographiques directement intégrées dans chaque message. Cela permet une gestion précise des zones d'intérêt sous forme rectangulaire, évitant ainsi une multiplication importante des abonnements et désabonnements lorsque les joueurs se déplacent. Contrairement à l'approche orientée tuile, où chaque déplacement peut induire de nombreux changements d'abonnement (en fonction du nombre de tuiles couvertes par la zone d'intérêt), l'approche orientée contenu ne nécessite généralement qu'un seul abonnement à actualiser au bout d'une période définie si et seulement si le joueur a fait un déplacement significatif. Cette caractéristique conduit à une réduction notable du nombre total de messages

échangés. Cependant, cette simplicité s'accompagne d'une contrainte importante : les zones d'intérêt rectangulaires définies dans l'approche orientée contenu ne prennent pas en compte les obstacles géographiques présents sur la carte du jeu, ce qui peut occasionner la réception inutile d'informations concernant des objets non visibles ou non accessibles par le joueur.

Concernant les expériences réalisées, plusieurs aspects du protocole de test méritent d'être discutés. Tout d'abord, les joueurs simulés dans l'expérience utilisent exclusivement des machines Linux, ce qui n'est pas représentatif de la diversité réelle des joueurs en matière d'équipement et de systèmes d'exploitation. De plus, la fréquence des messages de mise à jour envoyés par les joueurs a été fixée à une fois par seconde, ce qui apparaît très faible pour des jeux exigeant une forte réactivité. Ce choix sous-estime significativement le nombre de messages réellement échangés par les joueurs dans des conditions réelles de jeu, réduisant potentiellement la pertinence comparative des résultats avec d'autres solutions. Par ailleurs, le nombre limité d'essais réalisés (seulement cinq par solution) rend les résultats sensibles à des perturbations ou des erreurs ponctuelles, limitant leur robustesse statistique. Enfin, les expériences se concentrent principalement sur la consommation de bande passante au niveau des « *brokers* », sans fournir d'informations explicites sur la bande passante utilisée au niveau des joueurs eux-mêmes, alors même que ce critère serait essentiel pour évaluer concrètement l'impact des solutions sur l'expérience utilisateur finale. La définition arbitraire d'une limite maximale de latence de 400*ms* comme seuil critique constitue également une faiblesse méthodologique, dans la mesure où cette valeur n'est pas justifiée dans l'article, et se révèle largement supérieure au seuil des 100*ms* considéré dans ce mémoire comme la limite acceptable pour une jouabilité satisfaisante présentée plus haut.

Malgré ces limites expérimentales, les résultats présentés par Cañas et ses collègues apportent un éclairage précieux sur le comportement relatif des trois solutions proposées. Les données montrent que la solution orientée contenue génère globalement moins de trafic réseau en termes d'abonnements et de désabonnements grâce à sa flexibilité dynamique, tandis que la solution orientée tuile procure un compromis intéressant entre simplicité et efficacité en exploitant pleinement le modèle « *Pub/Sub* ». La solution orientée objet, bien que conceptuellement intuitive, se révèle quant à elle la moins efficace en raison d'un recours excessif aux échanges point-à-point. Ainsi, au regard des résultats obtenus et malgré les faiblesses méthodologiques identifiées, il apparaît que les approches orientées tuile et contenu se démarquent comme des choix prometteurs, à condition d'adapter précisément leur utilisation aux contraintes matérielles et de jouabilité propres à chaque jeu.

2.3 DISCUSSION

L'analyse comparative des articles étudiés révèle plusieurs tendances intéressantes quant à l'application de différents modèles de communication ou architecture aux « *MMOGs* ». L'ensemble des résultats d'analyse est récapitulé dans le tableau ci-dessous 2.1.

Solution	Modèle de communication	Bande passante (Mb/s)	Latence (ms)	Élasticité
[43]	Hybride — P2P	$43Mb/s$ (NCELL = 12, Bande passante $\approx NCELL$)	$25ms$ ($\pm 5ms$) pour 32 pairs (20 joueurs + NCELL)	Réattribution des rôles dynamiques. Réorganisation dynamique des cellules.
[48], [45]	Hybride — P2P	$0.022Mb/s$ de téléchargement et $0.165Mb/s$ d'envoi pour 7 entités	$0,06ms$ pour 7 entités	Réallocation dynamique des ressources et de la topologie.
[46]	Hybride — P2P	N/A	$10ms$ pour 25 pairs	Répartition dynamique des super-noeuds.
[49]	Hybride — P2P — Pub/Sub	N/A	$51ms$ (100 joueurs et 100 acteurs)	Allocation dynamique des routes d'accès vers les ressources.
[19]	Hybride — Pub/Sub	$16MB/s$	1 broker (230 clients) $\approx 400ms$ (moyenne = $50ms$)	Allocation dynamique des brokers. Élasticité faible, l'évolution du nombre de joueurs n'est pas proportionnelle au nombre de brokers (un broker = 230 clients, 8 brokers = 750 clients)

TABLEAU 2.1 : Analyse de l'état de l'art

Comme le montre le tableau récapitulatif, 2.1, la majorité des solutions s'accordent sur l'utilisation d'une architecture hybride, combinant le modèle Client/Serveur (« *C/S* ») à d'autres modèles comme le Pair-à-Pair (« *P2P* ») ou le « *Pub/Sub* ». Cette tendance souligne une volonté générale de conjuguer les avantages d'un contrôle centralisé avec les bénéfices des architectures distribuées, notamment en termes d'élasticité et de gestion dynamique des ressources.

En observant les solutions hybrides axées sur l'architecture « *P2P* », telles que *FreeMMG 2*, *NAHPALM* et les modèles basés sur les super-nœuds, on remarque que celles-ci partagent un effort important dans la réduction des charges serveur en déléguant une partie des tâches à des clients volontaires. Cette décentralisation partielle permet d'améliorer la latence, avec des résultats mesurés pouvant descendre jusqu'à 0,06 ms pour *NAHPALM*. Toutefois, un défaut récurrent réside dans leur dépendance pour la capacité des pairs : en effet, ces solutions sont gourmandes en ressources locales, particulièrement en termes de bande passante, ce qui restreint leur accessibilité à une fraction des joueurs disposant d'une infrastructure réseau suffisante.

De l'autre côté, les solutions hybrides intégrant le modèle « *Pub/Sub* », telles que celles explorées par Cañas et al., présentent une autre approche intéressante. Ici, le recours à un intermédiaire – le « *broker* » – permet de centraliser les opérations critiques tout en bénéficiant d'une gestion dynamique des flux d'informations. Cependant, l'étude montre que ces solutions rencontrent encore des difficultés à atteindre un niveau satisfaisant d'élasticité : l'évolution du nombre de « *brokers* » nécessaires n'est pas proportionnelle au nombre de clients pris en charge, entraînant une perte d'efficience en cas de forte augmentation du nombre de joueurs.

Concernant l'élasticité, critère déterminant pour l'efficacité opérationnelle des « *MMOGs* », il apparaît clairement que les solutions étudiées abordent cette problématique de manière très variable. *FreeMMG 2* et *NAHPALM*, par exemple, proposent une élasticité élevée grâce à une allocation dynamique des ressources et une réorganisation flexible de leur réseau de pairs. De même, la solution « *Pub/Sub* » étudiée affiche une élasticité semblable, mais limitée par les contraintes techniques et le coût de gestion croissant des « *brokers* » au-delà d'un certain seuil critique. Cette diversité de résultats démontre que le choix du modèle de communication, ainsi que sa mise en œuvre spécifique, conditionne fortement la capacité d'une solution à s'adapter efficacement aux fluctuations du nombre de joueurs.

Enfin, il est essentiel de noter une lacune générale dans l'état de l'art concernant la mesure approfondie des performances environnementales des solutions proposées. Bien que l'optimisation des ressources locales soit souvent évoquée implicitement (réduction des charges serveur, économies de bande passante), une quantification précise de la réduction des impacts énergétiques reste absente des études. Pourtant, dans le contexte actuel où l'informatique verte prend une importance grandissante, une évaluation rigoureuse et systématique de l'empreinte énergétique des solutions proposées apparaît comme un axe incontournable pour les recherches futures.

En constatant les limites d'accessibilité des solutions reposant sur le « *P2P* » pur, les difficultés d'élasticité rencontrées par les approches « *Pub/Sub* » existantes, ainsi que l'absence généralisée de métriques précises sur l'efficience énergétique, cet état de l'art dessine un portrait contrasté des architectures actuelles. Si certaines solutions se distinguent par leur capacité à proposer une faible latence ou une élasticité théorique élevée, aucune ne parvient à concilier simultanément les trois critères de performance fondamentaux que sont la bande passante, la latence et l'élasticité, tout en s'inscrivant pleinement dans une perspective d'informatique verte. Cette synthèse met donc en lumière la nécessité d'envisager une approche alternative, capable d'exploiter les avantages des modèles hybrides tout en atténuant leurs faiblesses structurelles. Dans cette optique, la section suivante introduit notre propre proposition. Celle-ci vise à intégrer les enseignements des point forts et des points faibles tirés de l'état de l'art pour construire une solution plus efficiente, plus flexible, et plus mesurée au niveau de son impact environnemental.

CHAPITRE III
PROPOSITION

Dans le contexte de notre étude sur l'impact du modèle de communication « *Pub/Sub* » sur l'efficience du traitement des données au niveau des serveurs de jeux massivement multijoueurs, il est essentiel de valider empiriquement nos hypothèses théoriques. Cette section présente l'expérimentation que nous avons conçue pour comparer les performances de deux approches de réplication de données appliquées au jeu vidéo : d'une part, le système de réplication natif d'*Unreal Engine*, largement utilisé dans l'industrie et représentatif des standards actuels, et d'autre part, une implémentation personnalisée basée sur le modèle « *Pub/Sub* ».

L'enjeu principal de cette expérimentation est de déterminer si le modèle « *Pub/Sub* » permet effectivement de réduire la consommation de bande passante, tout en maintenant un niveau de synchronisation adéquat entre les différents clients. Cette question est fondamentale dans notre recherche, car elle touche directement à l'efficience énergétique des serveurs de jeux. En effet, comme nous l'avons établi précédemment, le traitement et la transmission des données constituent l'une des principales sources de consommation énergétique dans les infrastructures des « *MMOGs* ». Si les échanges de données demeurent indispensables au fonctionnement même des « *MMOGs* », il est néanmoins possible d'optimiser ces communications en transformant leur mode d'organisation et de distribution, améliorant ainsi significativement leur efficience énergétique.

Notre approche expérimentale vise à quantifier précisément les différences entre les deux modèles de communication, en termes de volume de données transmises et de nombre de paquets échangés, dans diverses configurations spatiales simulant différents contextes de jeu. Ces mesures permettront d'évaluer objectivement si l'adoption du modèle « *Pub/Sub* » pourrait significativement contribuer à l'amélioration de l'efficience des serveurs, conformément à notre hypothèse initiale.

3.1 ANALYSE DE LA COMPOSITION DE LA BANDE PASSANTE DANS LES JEUX VIDÉO

Pour établir une base de comparaison pertinente entre les différentes approches de réplication, il est d'abord nécessaire de comprendre précisément quels éléments constituent la

majorité du trafic réseau dans un jeu vidéo multijoueur en ligne des années 2020. À cette fin, nous avons conduit une analyse préliminaire sur un jeu développé avec *Unreal Engine*, qui figure parmi les moteurs de jeu les plus populaires en 2024 [39].

3.1.1 SUPPORTS EXPÉRIMENTAUX

Pour notre analyse, nous utilisons le projet « *Lyra Starter Game* » [40] d'*Epic Games* [41], plus précisément son mode de jeu *Convolution*. Ce choix s'explique par plusieurs facteurs. Premièrement, *Lyra* est conçu comme une référence officielle et illustre les meilleures pratiques associées au moteur *Unreal Engine*. Deuxièmement, son caractère open-source et son système de programmation visuelle Blueprint en font un outil particulièrement adapté à la recherche, permettant de reproduire facilement les expériences réalisées.

Le mode *Convolution* offre un cadre idéal pour notre analyse : il s'agit d'un FPS dans lequel deux équipes de trois joueurs s'affrontent pour capturer trois points stratégiques, accumulant ainsi des points au fil du temps. La première équipe atteignant 125 points remporte la partie. Ce mode intègre les mécaniques classiques des jeux de tir (tirer, courir, recharger, changer d'arme, se soigner, etc.), ce qui en fait un représentant pertinent des jeux actuels utilisant *Unreal Engine*.

Les tests ont été effectués sur un ordinateur Windows 11 x64 bit (*32Gb ram et Intel i7-11800H @ 2.30GHz 11th Gen*) [42], qui est le système d'exploitation le plus utilisé par les joueurs en 2024, selon le rapport annuel de *Steam* [43]. Enfin, après l'enregistrement des données réseau issues de nos parties, celles-ci ont été analysées avec le logiciel « *Network Profiler* » inclus dans *Unreal Engine* 5.5.1. Cet outil nous a permis de voir distinctement les données pertinentes pour notre étude.

---

39. https://www.perforce.com/blog/vcs/most-popular-game-engines, Saisi le 7 juillet 2025

40. https://dev.epicgames.com/documentation/en-us/unreal-engine/lyra-sample-game-in-unreal-engine, Saisi le 7 juillet 2025

41. https://www.epicgames.com/site/en-US/home, Saisi le 7 juillet 2025

42. https://www.hp.com/us-en/gaming-pc/laptops.html, Saisi le 7 juillet 2025

43. https://store.steampowered.com/hwsurvey/Steam-Hardware-Software-Survey-Welcome-to-Steam, Saisi le 7 juillet 2025

3.1.2 PROTOCOLE D'ANALYSE

Pour réaliser notre expérience, nous lançons le projet en mode développement afin d'accéder aux outils d'analyse. Nous avons au préalable réglé les options de lancement sur deux joueurs avec le « *NetMode* » sur « *Play as Listen server* », puis sélectionné la carte « *L_Convolution_Blockout* » afin d'apparaître directement dans une partie sans naviguer dans les menus. Le « *NetMode* » dans *Unreal Engine* définit le comportement réseau de l'application : en choisissant « *Play as Listen server* », notre instance joue simultanément les rôles de client et de serveur, ce qui permet d'observer en temps réel les mécanismes de réplication tout en limitant les ressources nécessaires puisqu'une seule machine héberge l'ensemble du processus d'émulation réseau, rendant ainsi possibles nos mesures des communications client-serveur dans un environnement contrôlé.

Après avoir activé l'analyseur réseau via la commande « *NetProfile Enable* », nous effectuons d'abord une première partie pour garantir que toutes les ressources du projet sont bien chargées et que le mode de jeu fonctionne correctement. Cette étape préliminaire permet également de nous assurer que l'analyseur réseau serait actif dès le lancement de la seconde partie, évitant ainsi de manquer les premiers échanges réseau lors du chargement.

Durant la seconde partie, le joueur joue de manière cohérente au mode de jeu, simulant fidèlement les échanges réseau qui aurait lieu dans une partie standard entre joueurs humains. Le joueur essaye donc de capturer les objectifs de la carte tout en tuant les ennemis sur son chemin. Lorsque cela est possible, le joueur récupère les améliorations disponibles autour de lui ainsi que les objets de soins. La partie se déroule avec deux joueurs et quatre IA, l'un des deux joueurs restant inactif pour forcer *Unreal Engine* à simuler les échanges réseau, ce qui n'aurait pas été le cas avec un seul joueur présent (les IA étant calculées localement dans ce cas).

3.1.3 RÉSULTATS ET ANALYSE

Après l'analyse des données collectées durant la session de jeu, nous avons constaté que la réplication d'un acteur repose principalement sur deux éléments : les propriétés et les appels de procédures à distance («*RPCs*») (voir le tableau 3.1). Les propriétés sont des données persistantes attachées à un acteur qui définissent son état dans le monde du jeu et sont synchronisées automatiquement selon les règles de réplication définies ; elles peuvent

être marquées pour être répliquées du serveur vers les clients, des clients vers le serveur, ou dans les deux sens selon leur fonction. Les « *RPCs* », quant à eux, sont des mécanismes de communication temporaires qui déclenchent des fonctions à distance, permettant aux clients de signaler des événements ou des intentions (comme un tir ou un saut) que seul le serveur est autorisé à valider et à propager aux autres clients selon la logique d'autorité établie.

	Propriétés	RPCs
Volume (en KBytes)	641.7	269.4
Quantité	67 226	12 257

TABLEAU 3.1 : Total des échanges réseau par catégories de données

L'analyse a révélé que la proportion des réplications de propriétés est plus de deux fois supérieure à celle des « *RPCs* », tant en volume de données qu'en nombre de paquets. Plus précisément, sur une période d'analyse complète, nous avons mesuré 641,7 Ko et 67 226 instances pour la réplication des propriétés, contre 269,4 Ko et 12 257 instances pour les « *RPCs* ». En examinant plus en détail les propriétés les plus fréquemment répliquées, nous avons identifié que les trois plus courantes sont « *ReplicatedMovement* », « *ReplicatedAcceleration* » et « *RepAnimMontageInfo* », toutes liées au déplacement des joueurs. Cette prépondérance s'explique par le fait que les joueurs sont constamment en mouvement dans le jeu et que les positions de six personnages doivent être synchronisées en permanence.

Concernant les « *RPCs* », il est notable qu'une seule « *RPC* » spécifique, « *ServerMovePacked* », représente 60,5 % de toutes les « *RPCs* » transmises. Selon la documentation et l'analyse du code source d'*Unreal Engine*, cette « *RPC* » est utilisée pour répliquer la position des joueurs humains, tandis que la propriété « *ReplicatedMovement* » est dédiée à la position des intelligences artificielles. La RPC « *ServerMovePacked* » transmet une structure contenant des propriétés liées au mouvement des joueurs, auxquelles un algorithme de compression est appliqué pour optimiser l'utilisation de la bande passante. Cet algorithme est visible dans les deux extraits de code 3.1 et 3.2 provenant du code source d'*Unreal Engine*. Le tableau 3.2 illustre la compression de données obtenur grâce à ce système de « *RPC* ». Dans celui-ci, on peut retrouver l'ensemble des propriétés concernées par la « *RPC* » ainsi que leur type [44]. Avec le calcul suivant $TauxCompression = 1 - \frac{Taille_{compress}}{Taille_{initiale}} * 100$ on peut observer que le passage de toutes ces informations de manière groupée permet à *Unreal Engine* d'économiser entre

44. https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Classes, Saisi le 7 juillet 2025

Propriété	Type	Taille (Octet)	Taille compressée (Octet)
Temps de référence	*TimeStamp*	4	4
Accélération	*FVector3*	12	$6 \sim 9$ (1 si le vecteur est de taille zéro)
Position	*FVector3*	12	$6 \sim 9$
Rotation de la caméra	*FRotator*	12	$2 \sim 6$ (1 si le rotator est égale à zéro)
Marqueurs	*Booleans*	4	1 (si au moins une valeur est à **vrais**)
Objet d'appuis	*UPrimitiveComponent**	8	$0 \sim 8$ (si non-nul)
Mode de déplacement	*MovementMode*	4	$0 \sim 1$ (si différent de MODE_Marche)
Nom de l'os de l'objet de déplacement	*FName*	8	$0 \sim 8$ (si le nom n'est pas NAME_None
TOTAL		64	$12 \sim 46$

TABLEAU 3.2 : Liste des propriétés compressée impliquées dans les « *RPCs* » liées au mouvement

$28.13\% \sim 81.25\%$ de la taille originale des données transmises. Pour arriver à ce niveau de compression, le moteur utilise plusieurs méthodes telles que : l'omission des valeurs par défaut, la détection des valeurs nulles, la compression binaire ainsi que la quantization des nombres flottants. La nécessité quasi obligatoire d'envoyer sur le réseau les données de mouvement à chaque image du jeu pour garantir un mouvement fluide en multijoueur nous permet donc de comprendre l'intérêt d'un tel choix design pour répliquer ces données.

Cette analyse nous permet de conclure que les propriétés des objets constituent indéniablement la majorité du trafic réseau dans un jeu multijoueur typique, représentant plus des deux tiers du volume total des données échangées. Cette observation est cruciale pour notre expérimentation, car elle met en lumière un levier d'optimisation significatif : une approche plus réfléchie et ciblée de la transmission de ces propriétés pourrait entraîner une réduction substantielle de la bande passante globale. En effet, l'optimisation de la réplication des propriétés, en transmettant uniquement les données pertinentes aux entités qui en ont réellement besoin, constitue un axe prioritaire pour améliorer l'efficience des communications réseau dans les jeux multijoueurs.

```
/**
 * The actual network RPCs for character movement are passed to ACharacter, which wrap to the _Implementation call here, to avoid Component RPC overhead.
 * For example:
 *  Client: UCharacterMovementComponent::ServerMovePacked_ClientSend() => Calls CharacterOwner->ServerMove() triggering RPC on the server.
 *  Server: ACharacter::ServerMovePacked_Implementation() from the RPC => Calls CharacterMovement->ServerMove_ServerReceive(), unpacked and sent to ServerMove_ServerHandleMoveData().
 *
 * ServerMove_ClientSend() and ServerMove_ServerReceive() use a bitstream created from the current FCharacterNetworkMoveData data container that contains the client move.
 * See GetNetworkMoveDataContainer()/SetNetworkMoveDataContainer() for details on setting a custom container with custom unpacking through FCharacterNetworkMoveData::Serialize().
 *
 */

/**
 * Wrapper to send packed move data to the server, through the Character.
 * @see CallServerMovePacked()
 */
    ENGINE_API void ServerMovePacked_ClientSend(const FCharacterServerMovePackedBits& PackedBits);

/**
 * On the server, receives packed move data from the Character RPC, unpacks them into the FCharacterNetworkMoveDataContainer returned from GetNetworkMoveDataContainer(),
 * and passes the data container to ServerMove_HandleMoveData().
 */
    ENGINE_API void ServerMovePacked_ServerReceive(const FCharacterServerMovePackedBits& PackedBits);

```

FIGURE 3.1 : Extrait de code provenant du fichier CharacterMovementComponent.hpp

3.2 SUPPORTS EXPÉRIMENTAUX

3.2.1 LE JEU

Pour mesurer précisément l'impact du modèle de communication « *Pub/Sub* » sur la bande passante, nous développons un jeu expérimental très simpliste qui nous permet d'isoler

```
bool FCharacterNetworkMoveData::Serialize(UCharacterMovementComponent&
    CharacterMovement, FArchive& Ar, UPackageMap* PackageMap,
    FCharacterNetworkMoveData::ENetworkMoveType MoveType)
{
    NetworkMoveType = MoveType;

    bool bLocalSuccess = true;
    const bool bIsSaving = Ar.IsSaving();

    Ar << TimeStamp;

    // TODO: better packing with single bit per component indicating
    zero/non-zero
    Acceleration.NetSerialize(Ar, PackageMap, bLocalSuccess);

    Location.NetSerialize(Ar, PackageMap, bLocalSuccess);

    // ControlRotation : FRotator handles each component zero/non-zero
    test; it uses a single signal bit for zero/non-zero, and uses 16 bits
     per component if non-zero.
    ControlRotation.NetSerialize(Ar, PackageMap, bLocalSuccess);

    SerializeOptionalValue<uint8>(bIsSaving, Ar, CompressedMoveFlags, 0)
    ;

        SerializeOptionalValue<UPrimitiveComponent*>(bIsSaving, Ar,
    MovementBase, nullptr);
        SerializeOptionalValue<FName>(bIsSaving, Ar,
    MovementBaseBoneName, NAME_None);
        SerializeOptionalValue<uint8>(bIsSaving, Ar, MovementMode,
    MOVE_Walking);

    return !Ar.IsError();
}

```

FIGURE 3.2 : Fonction de sérialisation employée par les « *RPCs* » liées au mouvement.

et de mettre en avant de manière claire la réplication des propriétés afin de pouvoir comparer la solution de réplication d'*Unreal Engine* et notre solution « *Pub/Sub* ».

Notre jeu expérimental met en scène quatre acteurs, chacun possédant 10 attributs répliqués (entier, flottant, couleur, texte, etc.). Ces attributs sont délibérément diversifiés pour représenter la variété des données typiquement répliquées dans un jeu vidéo réel, tout en maintenant une complexité maîtrisée pour l'analyse. La conception minimaliste du jeu nous permet de nous concentrer spécifiquement sur les mécanismes de réplication, sans les

interférences que pourrait introduire un gameplay plus complexe. L'ensemble des attributs et leurs types sont listés dans le tableau 4.1 et 4.2.

Afin de n'avoir aucun biais dans notre évaluation des deux solutions de réplications, nous testons les deux sur le même jeu afin de garantir que seul le code réseau diffère. La première version, utilisant le système de réplication natif d'*Unreal Engine*, représente l'approche standard de l'industrie, comme une sorte de témoin. La seconde version, utilisant notre approche « *Pub/Sub* », est quant à elle la version expérimentale étudiée dans notre analyse des résultats.

3.2.2 LOGICIELS ET MATÉRIEL

Nos expérimentations sont menées dans un environnement contrôlé, utilisant la configuration suivante :

Machine : PC portable OMEN by HP Laptop 17-ck0xxx [45]
Processeur : 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz
Mémoire RAM : 32.0 GB (31.6 GB utilisable)
Système d'exploitation : Windows 64-bit (x64-based processor)
Moteur de jeu : *Unreal Engine 5.5.1*
Outil d'analyse : *Unreal Engine 5.5.1* Network Profiler (DotNET)

Cette configuration matérielle représente un environnement de développement à jour vis-à-vis des contextes de développement et de jeux en 2024/2025, permettant d'obtenir des résultats représentatifs. Le choix d'*Unreal Engine 5.5.1* [46] s'explique par sa popularité dans l'industrie et sa robustesse en matière de fonctionnalités réseau, ce qui en fait une référence pertinente pour notre étude. De son côté, l'outil Network Profiler intégré à *Unreal Engine* nous permet de capturer et d'analyser avec précision tous les échanges réseau pendant nos sessions de test, en distinguant les différents types de paquets, leur taille, leur fréquence et leur contenu.

45. https://www.hp.com/us-en/gaming-pc/laptops.html, Saisi le 7 juillet 2025
46. https://www.unrealengine.com/en-US/blog/unreal-engine-5-5-is-now-available, Saisi le 7 juillet 2025

3.3 MÉTHODOLOGIE EXPÉRIMENTALE

Afin de reproduire le plus de situations possibles, notre expérience vise à comparer la réplication des deux jeux dans différentes situations, simulant le comportement des joueurs dans différentes configurations spatiales. Pour chaque configuration, nous avons exécuté des sessions de test identiques avec les deux versions du jeu (réplication native et réplication « *Pub/Sub* »), et collecté des données exhaustives sur les échanges réseau. Nous avons défini trois configurations spatiales principales pour nos tests.

Dans la première configuration, tous les acteurs sont placés dans un même espace ouvert et sont visibles les uns des autres. Ce scénario simule une situation dans laquelle tous les joueurs interagissent dans la même zone, comme une bataille de groupe ou un événement général dans un « *MMOG* ». La figure 3.3 montre un niveau de jeu avec un simple sol et quatre joueurs qui sont tous dans le champ de vision les uns des autres. Aucun objet ou obstacle n'est présent pour obstruer la vue entre les acteurs.

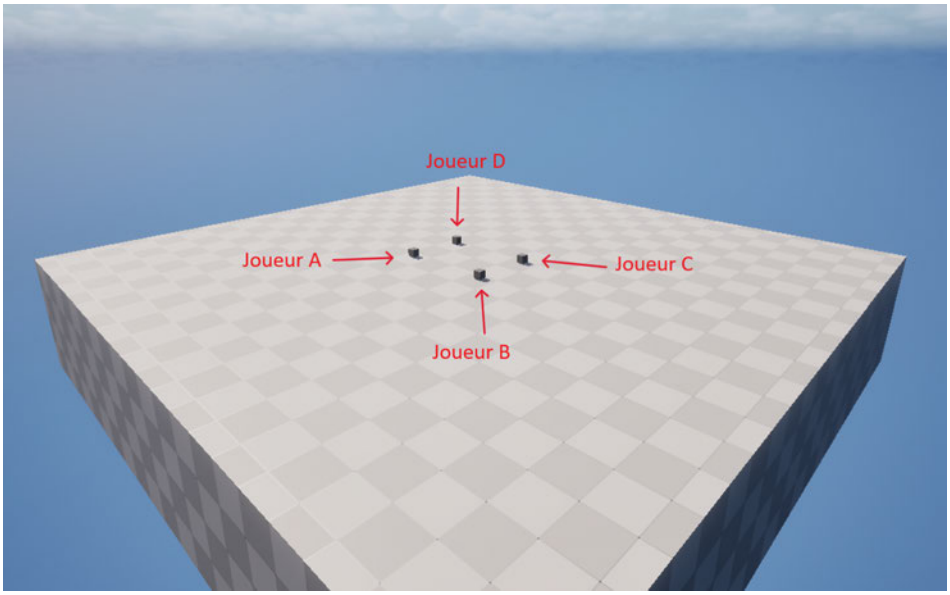

FIGURE 3.3 : Environnement de jeu multijoueur, ouvert et sans obstacle

Dans la seconde configuration, les acteurs sont répartis dans différents espaces séparés par des murs opaques, ce qui empêche toute visibilité entre eux. Cette configuration reproduit des situations dans lesquelles les joueurs évoluent dans des zones distinctes d'une carte, sans interaction visuelle directe. La figure 3.4 illustre cette situation avec un niveau semblable à

celui présenté précédemment, auquel on vient ajouter un mur opaque qui sépare les quatre joueurs en deux groupes de deux. Ce cas de figure est fréquent dans les jeux, notamment lorsque les joueurs partagent un même niveau, mais sont masqués les uns des autres par des objets statiques (murs, décors, végétations, etc.).

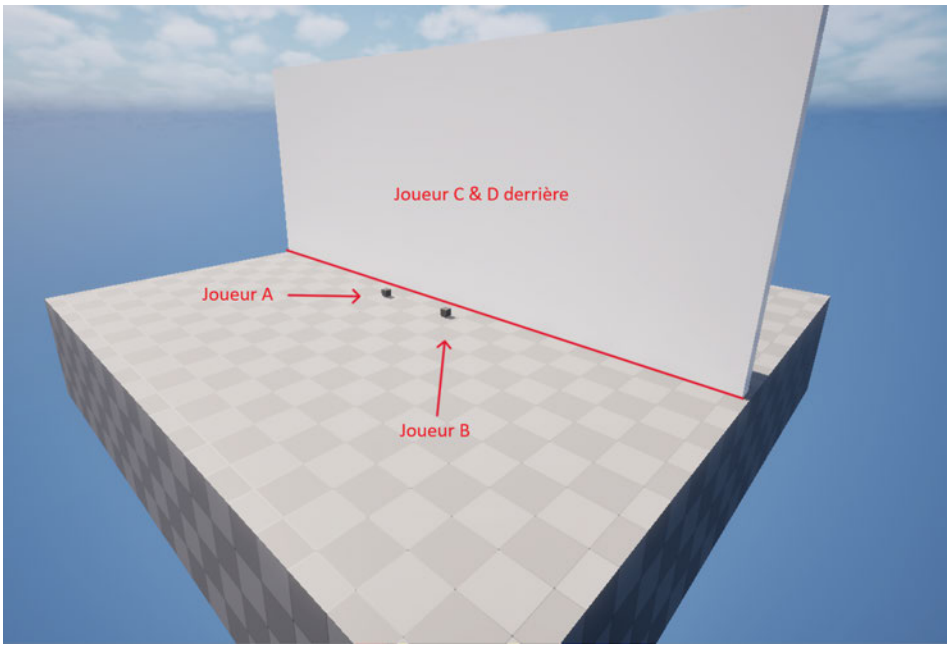

FIGURE 3.4 : Environnement de jeu multijoueur restreint par un obstacle opaque

Enfin, la troisième configuration introduit des parois vitrées, permettant une visibilité partielle entre certains groupes d'acteurs. Ce scénario simule des environnements de jeu complexes où la visibilité entre joueurs varie en fonction de leur position. Comme montré dans la figure 3.5, les joueurs sont ici répartis dans quatre espaces distincts, rendant impossible toute interaction directe. Toutefois, ce niveau met en évidence le fait que, malgré la séparation physique ou l'impossibilité d'interaction, un besoin de synchronisation visuelle demeure.

Pour chaque configuration, nous lançons des simulations sur une période de 60 secondes lors desquelles nous enregistrons les performances des solutions de réplication. Les simulations sont lancées de manière successive en alternant entre la solution standard et la solution « *Pub/Sub* » jusqu'à l'obtention de cinq séries de mesures. Cette opération est répétée pour chaque situation présentée ci-dessus. Dans le cadre de la solution standard, établie avec *Unreal Engine*, nous utilisons « *Unreal Engine 5.5.1 Network Profiler (DotNET)* » pour faire les

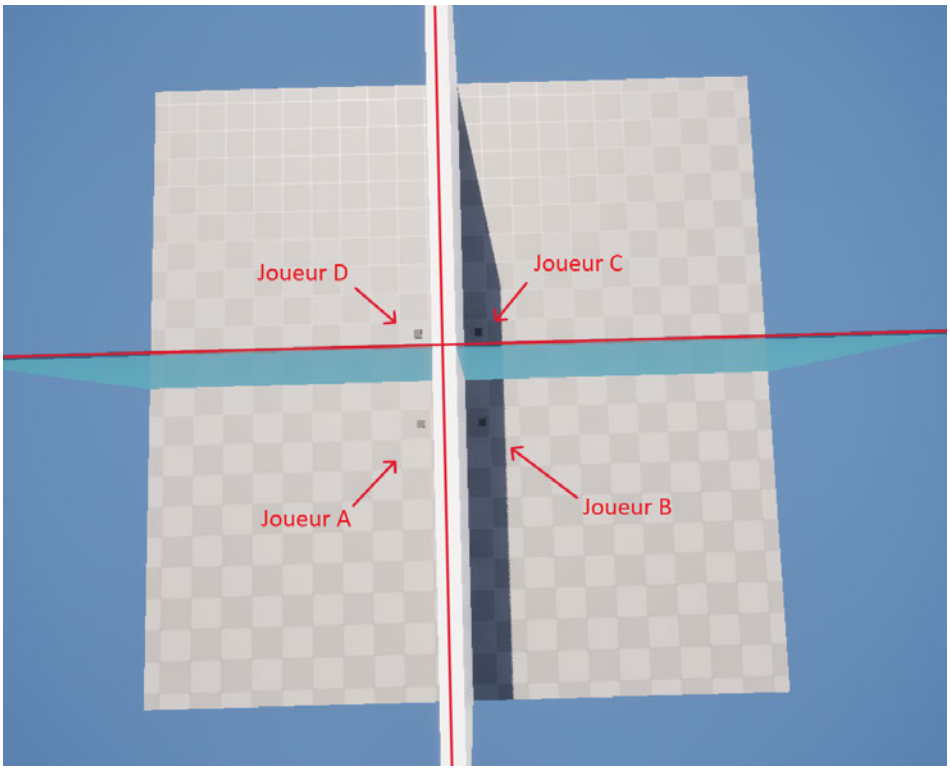

FIGURE 3.5 : Environnement de jeu multijoueur restreint par des obstacles opaques et transparents

mesures, tandis que pour la solution « *Pub/Sub* » c'est le « *broker* » qui se charge de tenir les comptes de tout ce qui transite par lui. Les données mesurées sont le volume total de données transmises (en kilooctets) et le nombre total de paquets échangés.

La solution « *Pub/Sub* » nécessite un algorithme de gestion de l'intérêt pour fonctionner. C'est pourquoi nous établissons deux groupes de données parmi les propriétés répliquées de l'expérience. Dans un premier temps, il y a le groupe « physique » qui comprend la vie, la couleur, la position et la rotation du joueur. Ce groupe représente l'ensemble des informations nécessaires au bon affichage d'un joueur à l'écran. Dans un second temps, il y a le groupe « administratif » qui comprend le score, le titre, le nom, les statistiques, la date de dernière mise à jour et le statut d'appartenance en équipe du joueur. Ce second groupe contient les informations nécessaires pour interagir avec un joueur et connaître l'état « abstrait » du joueur. Après avoir défini ces deux groupes, nous définissons pour chacun d'eux une règle qui détermine si oui ou non un joueur y est intéressé. Lorsqu'un joueur **A** a dans son champ de vision un joueur **B** alors, il est intéressé par son groupe *physique* et réciproquement. Si le

joueur **A** ou **B** est en mesure d'interagir avec l'autre, alors ils sont tous deux intéressés par le groupe *administratif* de l'autre joueur. La figure 3.6 met en scène les règles de gestion de l'intérêt de notre prototype en associant le groupe *administratif* à la carte d'identité du joueur et le groupe *physique* à son corps. Dans la situation introduite dans la figure 3.7, le joueur **A** a le joueur **B** et joueur **D** en ligne de vue, alors il est intéressé par leur groupe *physique*. Les joueurs **B** et **D** ne se voient pas, donc ils sont seulement intéressés par le groupe *physique* du joueur **A**. Seuls les joueurs **A** et **B** sont en capacité d'interagir ensemble, alors ils sont mutuellement intéressés par leur groupe *administratif* respectif. De son côté, le joueur **E** ne voit et ne peut interagir avec personne, il n'est donc intéressé par aucun autre joueur.

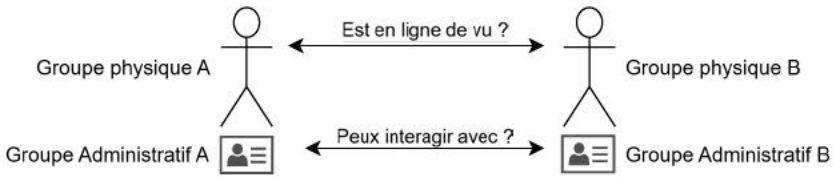

FIGURE 3.6 : Illustration des règles de gestion de l'intérêt appliqué aux groupes de données

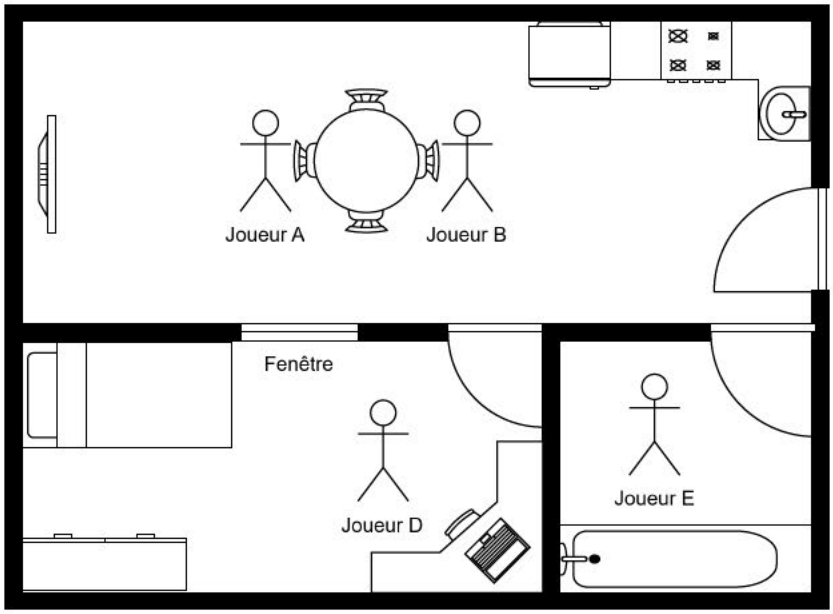

FIGURE 3.7 : Exemple d'application des règles de gestion de l'intérêt.

Cette approche méthodologique nous permet d'évaluer non seulement les performances globales des deux systèmes de réplication, mais également leur comportement spécifique

face à différentes contraintes spatiales et de visibilité, facteurs déterminants dans les jeux multijoueurs réels. En multipliant les situations d'analyse et en collectant systématiquement un large échantillon de mesures pour chaque configuration, nous visons à obtenir des résultats représentatifs qui permettront d'établir des conclusions fiables sur l'efficience comparative des deux modèles de communication. La section suivante présentera en détail l'analyse de ces résultats expérimentaux et leurs implications dans l'amélioration de l'efficience de la consommation énergétique des serveurs de jeux ou non.

CHAPITRE IV
ÉVALUATION

Cette section présente les résultats de notre expérimentation comparant la solution de réplication standard basée sur *Unreal Engine* et notre implémentation personnalisée utilisant le modèle « *Pub/Sub* ». Les données ont été collectées dans les trois configurations spatiales différentes présentées dans la section précédente.

4.1 LA SOLUTION STANDARD

L'analyse des résultats de la solution standard, présentés dans le Tableau 4.1, révèle plusieurs tendances significatives concernant la réplication des différentes propriétés des objets. Tout d'abord, nous observons que les types de données composées, en particulier la structure *Statistiques* contenant quatre entiers, génèrent un volume de s nettement supérieur aux autres types de données. En effet, cette propriété présente une moyenne de réplications variant entre 9 307,6 et 9 782,2 selon l'environnement, soit plus de trois fois le volume de réplications d'une propriété simple comme un entier ou un flottant. Ce phénomène s'explique par la nécessité de synchroniser la structure dès qu'un seul de ses membres est modifié, multipliant ainsi la fréquence de réplication. De même, la propriété « Couleur », représentée par un vecteur à quatre composantes (R,G,B,A), affiche des moyennes de réplication élevées (entre 5 900,2 et 6 113,2), confirmant cette tendance.

Propriété	Type	Réplication (Vide)	Réplication (Murs)	Réplication (Vitres)
Score	*Integer*	2 471.4	2 361.2	2 444.8
Vie	*Float*	2 491.8	2 386.4	2 465.6
Couleur	*Vector4f* (R,G,B,A)	6 113.2	6 009.8	5 900.2
Titre	*FString*	2 467.8	2 361.8	2 436.2
Nom	*FName*	2 465.9	2 362.3	2 436.2
Position	*Vector2f*	2 491.8	2 386.4	2 465.6
Rotation	*FRotator*	2 491.8	2 386.4	2 465.6
En Équipe	*Bool*	967.6	1590.4	1 463
Statistiques	*Struct (4 int)*	9 307.6	9 468.2	9 782.2
Dernière mise à jour	*FDateTime*	2 499.2	2 393	2 474.2

TABLEAU 4.1 : Moyenne de réplications des propriétés pour la solution standard

Par ailleurs, nous remarquons que la propriété booléenne « En Équipe » présente des moyennes de réplications significativement inférieures aux autres types de données (967,6 dans l'environnement vide, 1 590,4 avec des murs, et 1 463 avec des vitres). Cette disparité s'explique par la nature binaire du type booléen : lorsqu'une nouvelle valeur aléatoire est générée, la probabilité qu'elle soit identique à la précédente est de 50 %, réduisant ainsi de moitié les chances de rendre nécessaire une réplication. Les écarts importants observés entre les trois environnements pour cette propriété sont également attribuables à cette caractéristique. Il convient de noter que si l'expérience avait été répétée plus de cinq fois, la moyenne aurait vraisemblablement convergé vers une valeur plus stable, réduisant ainsi ces écarts.

Enfin, à l'exception de la propriété booléenne discutée précédemment, il est intéressant de constater que la quantité de réplication ne semble pas être significativement affectée par les différents environnements de test. En effet, pour la plupart des propriétés, les moyennes de réplications restent dans le même ordre de grandeur à travers les trois configurations spatiales, avec des variations relativement faibles. Cette observation suggère que la solution standard d'*Unreal Engine* réplique systématiquement les données à tous les clients, indépendamment de leur pertinence contextuelle ou de leur visibilité dans l'environnement de jeu, ce qui représente un axe majeur d'amélioration.

4.2 LA SOLUTION « *PUB/SUB* »

L'examen des résultats de notre solution basée sur le modèle « *Pub/Sub* », présentés dans le Tableau 4.2, met en évidence des caractéristiques distinctives par rapport à l'approche standard. Nous observons d'abord que, dans l'environnement vide, l'ensemble des propriétés ont été répliquées un nombre similaire de fois, avec des valeurs oscillant entre 3 809 et 3 990. Cette homogénéité relative indique que notre implémentation actuelle ne discrimine pas entre les différents types de données lors de la réplication. Il est important de noter que ces performances pourraient être significativement améliorées en implémentant une détection des valeurs similaires, permettant d'éviter des synchronisations inutiles lorsque les valeurs ne changent pas. Un tel mécanisme réduirait considérablement le volume global de données transmises, particulièrement pour les propriétés qui varient peu durant une session de jeu.

Propriété	Type	Réplication (Vide)	Réplication (Murs)	Réplication (Vitres)
Score	*Integer*	3 928	1 276	0
Vie	*Float*	3 990	1 276	1 302
Couleur	*Vector4f* (R,G,B,A)	3 990	1 276	1 299
Titre	*Fstring*	3 987	1 276	0
Nom	*FName*	3 981	1 274	0
Position	*Vector2f*	3 966	1 213	1 294
Rotation	*FRotator*	3 963	1 268	1 293
En Équipe	*Bool*	3 954	1 206	0
Statistiques	*Struct (4 int)*	3 948	1 264	0
Dernière mise à jour	*FDateTime*	3 809	1 261	0

TABLEAU 4.2 : Moyenne de réplications des propriétés pour la solution « *Pub/Sub* »

Le résultat le plus frappant concerne l'impact de l'environnement de test sur la quantité de réplications. Contrairement à la solution standard, notre implémentation « *Pub/Sub* » démontre une sensibilité marquée aux conditions environnementales, reflétant l'efficacité de son algorithme de gestion de l'intérêt. Dans l'environnement avec murs, nous observons une réduction drastique du nombre de réplications, qui passe d'environ 3 900 à environ 1 270 pour la plupart des propriétés, soit une division par trois du volume de données transmises. Cette

réduction significative s'explique par la capacité du système à déterminer précisément quels clients ont réellement besoin de recevoir les mises à jour en fonction de leur contexte spatial.

Plus révélateur encore, dans l'environnement avec vitres, notre solution démontre une capacité de discrimination fine : certaines propriétés (Score, Titre, Nom, En Équipe, Statistiques, Dernière mise à jour) ne sont plus du tout répliquées, tandis que d'autres jugées essentielles à la jouabilité (Vie, Couleur, Position, Rotation) continuent d'être transmises avec des moyennes autour de 1 300. Cette sélectivité témoigne de l'importance de l'algorithme de gestion de l'intérêt, capable d'adapter précisément le flux d'informations en fonction de la visibilité partielle offerte par les parois vitrées. Il est important de noter que cet algorithme représente une charge de travail supplémentaire pour les studios de développement de jeux vidéo, mais qu'il est également un outil garantissant une chance de gagne en efficience lors des synchronisations réseaux.

4.3 COMPARAISON DES RÉSULTATS

La comparaison directe des performances des deux solutions révèle des différences fondamentales dans leur approche de la réplication des données, avec des implications significatives pour l'efficience énergétique des serveurs de jeux.

Il apparaît clairement que la solution « *Pub/Sub* » présente une capacité supérieure à s'adapter au gameplay et aux besoins réels des joueurs. Cette adaptabilité se traduit par une réduction notable du volume de données transmises en s'appuyant notamment sur les règles du jeu tel que l'environnement, par exemple. Cependant, pour exploiter pleinement le potentiel de cette approche, il serait nécessaire d'intégrer à notre implémentation « *Pub/Sub* » les mécanismes d'optimisation déjà présents dans la solution standard d'*Unreal Engine*, tels que la détection des valeurs identiques, l'omission des valeurs par défaut, la compression des données vectorielles, entre autres. L'incorporation de ces techniques permettrait de réduire davantage le volume global des données transmises, maximisant ainsi les gains d'efficience.

Il est intéressant de noter que dans l'environnement de test vide, où tous les acteurs sont visibles les uns des autres, la solution « *Pub/Sub* » se révèle actuellement moins efficiente que la solution standard. En effet, les moyennes de réplications pour la solution « *Pub/Sub* » (environ 3 900) dépassent celles de la solution standard (environ 2 400) pour la plupart des propriétés simples. Cette observation s'explique par l'absence, dans notre implémentation

actuelle, des mécanismes d'optimisation mentionnés précédemment. Néanmoins, l'intégration de ces mécanismes permettrait à notre solution de tendre vers des performances similaires, voire supérieures, à celles de la solution standard dans ce scénario de visibilité totale.

En revanche, dans les environnements avec des murs et des vitres, l'avantage de la solution « *Pub/Sub* » devient incontestable. Dans la configuration avec murs, notre solution réduit le nombre de réplications à environ 1 270 pour la plupart des propriétés, soit une diminution de près de 50 % par rapport aux moyennes de la solution standard (environ 2 400). Plus impressionnant encore, dans l'environnement avec vitres, certaines propriétés ne sont plus du tout répliquées, tandis que d'autres voient leur volume de réplication réduit à environ 1 300, contre 2 400 à 2 500 pour la solution standard. Ces réductions substantielles du volume de données transmises se traduiraient directement par une diminution de la charge de travail des serveurs et, par conséquent, de leur consommation énergétique, conformément à notre hypothèse initiale.

En conclusion, bien que notre implémentation actuelle du modèle « *Pub/Sub* » présente des opportunités d'optimisation encore non exploitées, elle démontre déjà un potentiel considérable pour améliorer l'efficience du traitement des données dans les « *MMOGs* », particulièrement dans les environnements de jeu complexes et à grande échelle qui caractérisent les « *MMOGs* » à partir des années 2010.

CHAPITRE V
DISCUSSION

L'étude conduite dans le présent mémoire s'est attachée à explorer l'impact potentiel du modèle de communication « *Pub/Sub* » sur l'efficience énergétique des serveurs de « *MMOGs* ». L'originalité de notre démarche réside dans son ancrage explicite dans le domaine de l'informatique verte, approche encore relativement marginale dans la recherche sur les architectures réseau des jeux vidéo. Cette section vise à prendre du recul sur l'ensemble des travaux effectués, en identifiant leurs forces et leurs limites, ainsi qu'en suggérant des pistes d'investigation complémentaires pour des recherches futures.

5.1 VALIDITÉ DE L'HYPOTHÈSE FONDAMENTALE

Dans le cadre de cette recherche, nous avons posé comme hypothèse fondamentale que la consommation énergétique des équipements réseau (routeurs, commutateurs, serveurs, etc.) est directement influencée par le volume de données qu'ils doivent traiter. Cette hypothèse constitue la pierre angulaire de notre approche visant à améliorer l'efficience des architectures de communication dans les « *MMOGs* ». Elle nous permet d'établir un lien causal entre la réduction du volume de trafic réseau et l'économie d'énergie réalisable, offrant ainsi une perspective concrète pour l'informatique verte appliquée au secteur des jeux vidéo en ligne.

Afin de valider empiriquement cette hypothèse, nous avons mis en place une expérimentation qui regroupe cinq ordinateurs et un commutateur CISCO 10 Gb/s. Nos quatre ordinateurs clients envoient de la bande passante de manière contrôlée grâce au logiciel *iPerf3* vers le cinquième (le serveur), tandis que nous suivons la consommation énergétique du commutateur à l'aide d'un capteur non invasif *SCT013* [47] couplé à un microcontrôleur programmable *Arduino* [48] pour mesurer la consommation électrique en temps réel (voir la figure 5.1 et 5.2). Cependant, les résultats de cette expérimentation se sont révélés non concluants, la consommation énergétique observée demeurant invariablement identique, quelle que soit la charge réseau appliquée.

---

47. https://www.gotronic.fr/art-capteur-de-courant-5-a-sct013-005-22483.htm?srsltid=AfmBOorwm86RLadMa9jsx7-IEja7zGnWL_lzn-zAFwLRZs3C9Z1v05dI, Saisi le 7 juillet 2025

48. https://store.arduino.cc/collections/nano-family/products/nano-esp32, Saisi le 7 juillet 2025

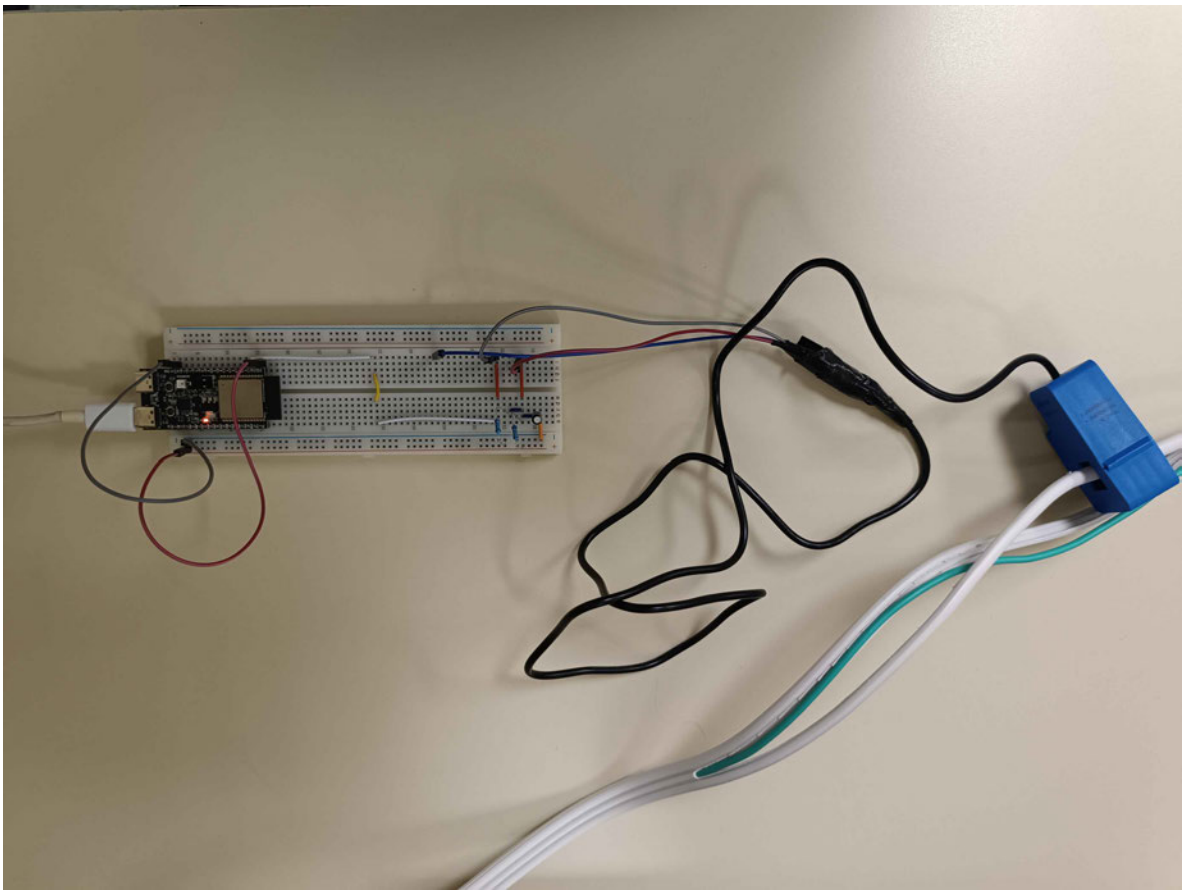

FIGURE 5.1 : Montage *Arduino* et *capteur SCT013*

Après analyse approfondie de notre dispositif expérimental, nous avons identifié que nous ne générions pas suffisamment de données pour solliciter réellement le commutateur au-delà de sa capacité nominale. Ce dernier parvenait donc à supporter la charge sans nécessiter d'augmentation de sa puissance de traitement, et par conséquent, sans variation notable de sa consommation énergétique. Cette observation souligne la difficulté de reproduire à l'échelle d'un laboratoire les conditions réelles d'une architecture réseau industrielle, particulièrement dans le contexte des « *MMOGs* » qui manipulent des volumes de données considérablement plus importants.

Il est donc intéressant de noter que, bien que la théorie et la littérature scientifique s'accordent sur la relation entre volume de données et consommation énergétique, il n'est pas toujours évident de mettre en place des supports expérimentaux suffisamment robustes pour simuler fidèlement les conditions réelles d'exploitation d'infrastructures réseau à l'échelle

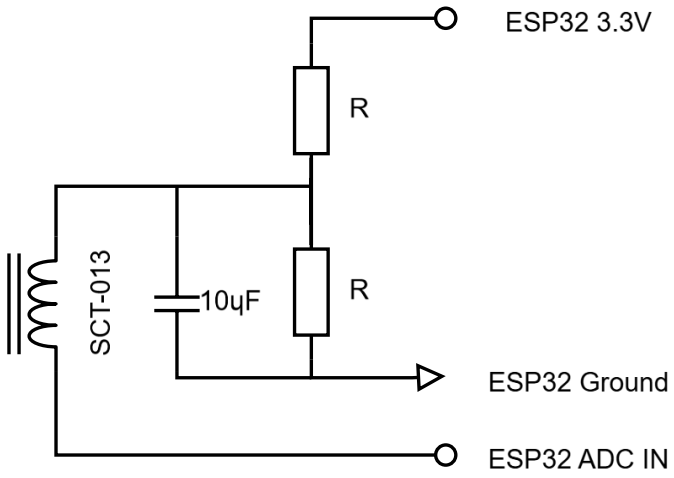

FIGURE 5.2 : Schéma électrique du montage *Arduino* et *capteur SCT013*

industrielle. Cette difficulté méthodologique constitue en soi un enseignement précieux pour les futures recherches dans ce domaine.

5.2 EFFICACITÉ DU MODÈLE PUB/SUB ET GESTION DE L'INTÉRÊT

L'un des résultats les plus significatifs de notre recherche concerne la démonstration de l'efficacité du modèle de communication « *Pub/Sub* » pour réduire la quantité de données échangées dans un contexte de jeu massivement multijoueur en ligne. Nos expérimentations ont révélé que cette réduction est particulièrement notable dans les environnements où les règles du jeu limitent les interactions entre les joueurs (visibilité, système d'équipe, etc.), validant ainsi l'hypothèse selon laquelle une transmission plus ciblée des informations permettrait de diminuer la charge globale du réseau.

Toutefois, il est important de reconnaître que les gains d'efficience observés proviennent en grande partie de l'algorithme de gestion de l'intérêt que nous avons implémenté. Ce mécanisme, qui détermine quelles informations sont pertinentes pour quels clients en fonction de leur contexte spatial, pourrait théoriquement être intégré dans des solutions de réplication standard, sans nécessairement adopter le modèle « *Pub/Sub* » dans son intégralité. Par exemple, le système de réplication natif d'*Unreal Engine* pourrait être étendu pour intégrer une gestion de l'intérêt plus sophistiquée, tout en conservant son architecture de communication actuelle.

Cette observation soulève une question méthodologique importante : dans quelle mesure les améliorations constatées sont-elles attribuables au modèle de communication « *Pub/Sub* » lui-même, plutôt qu'à la simple intégration d'une gestion d'intérêt plus efficace ? Pour répondre à cette question, une expérimentation complémentaire pourrait être envisagée, comparant trois approches :

1. La solution standard actuelle d'*Unreal Engine*, sans gestion d'intérêt avancée ;
2. La solution standard d'*Unreal Engine*, augmentée d'un algorithme de gestion d'intérêt similaire à celui utilisé dans notre solution « *Pub/Sub* »
3. Notre solution complète basée sur le modèle « *Pub/Sub* »

Une telle expérimentation permettrait d'isoler plus précisément l'impact du modèle de communication « *Pub/Sub* » au-delà de la simple intégration d'une gestion d'intérêt efficace. Cette distinction est cruciale pour évaluer la pertinence d'une refonte complète des architectures de communication des jeux massivement multijoueurs vers le modèle « *Pub/Sub* ».

5.3 COÛTS ÉNERGÉTIQUES DISTRIBUÉS ET ARCHITECTURE GLOBALE

Notre recherche, orientée vers l'informatique verte et le développement durable, propose une solution qui introduit un « *broker* » comme intermédiaire de communication entre l'autorité du jeu et les joueurs. Ce choix architectural soulève une question essentielle : les économies d'énergie réalisées au niveau du serveur de jeu ne sont-elles pas simplement déplacées vers le « *broker* », sans réduction nette de la consommation globale ?

En effet, le « *broker* » doit lui-même consommer des ressources pour assurer son rôle de médiation, notamment pour le routage des messages entre les émetteurs et les récepteurs. Dans une architecture traditionnelle « *C/S* », cette charge est directement assumée par le serveur de jeu. Notre solution la déplace vers une entité distincte, ce qui pourrait simplement redistribuer la consommation énergétique plutôt que de la réduire.

Cependant, plusieurs arguments suggèrent que cette redistribution pourrait néanmoins entraîner une réduction nette de la consommation énergétique globale :

1. **Spécialisation fonctionnelle :** le « *broker* », étant spécialisé dans la gestion des messages, peut potentiellement accomplir cette tâche de manière plus efficiente qu'un

serveur de jeu polyvalent qui doit également gérer la logique du jeu, les simulations physiques, l'intelligence artificielle, etc.

2. **Filtrage précoce :** le « *broker* » peut filtrer les messages non pertinents avant qu'ils n'atteignent le serveur, réduisant ainsi la charge de travail globale du système.
3. **Élasticité optimisée :** la séparation des responsabilités entre le serveur de jeu et le « *broker* » permet une allocation plus précise des ressources en fonction des besoins, potentiellement plus efficiente qu'une solution monolithique.

Néanmoins, une évaluation rigoureuse de cette hypothèse nécessiterait une mesure précise de la consommation énergétique de l'ensemble de l'infrastructure, incluant à la fois le serveur de jeu et le « *broker* », comparée à celle d'une architecture traditionnelle. Cette mesure pourrait faire l'objet d'un travail de recherche ultérieur, apportant une perspective plus complète sur l'efficience énergétique globale de l'approche proposée.

Par ailleurs, une évolution architecturale plus radicale pourrait être envisagée conjointement à des travaux en cybersécurité : la suppression complète du serveur dédié, en déplaçant l'autorité du jeu directement vers les joueurs, tout en conservant le « *broker* » comme médiateur des communications. Cette approche offerte par le « *broker* », inspirée des architectures pair-à-pair tout en bénéficiant de la gestion centralisée des communications, pourrait théoriquement éliminer entièrement les coûts énergétiques associés au maintien d'un serveur dédié. Toutefois, cette évolution soulèverait d'importantes questions de sécurité et de fiabilité, nécessitant des mécanismes robustes pour prévenir la triche et assurer la cohérence du monde de jeu en l'absence d'une autorité centrale. Cette solution, bien que de plus en plus étudiée dans la recherche académique, est encore insatisfaisante d'un point de vue pratique pour être appliquée dans l'industrie, comme nous l'avons vu dans l'état de l'art.

5.4 ÉQUILIBRE ENTRE EFFICIENCE ÉNERGÉTIQUE ET PERFORMANCES

Notre recherche s'est principalement concentrée sur l'aspect énergétique des solutions proposées, une perspective relativement novatrice dans le domaine du jeu vidéo où les considérations de performance (latence, fluidité, fidélité graphique) priment généralement sur l'efficience énergétique. Ce choix délibéré d'orienter notre travail vers l'informatique verte ouvre un champ d'investigation important, mais soulève également des questions sur l'équilibre optimal entre efficience énergétique et performances perçues par l'utilisateur.

En effet, certaines optimisations visant à réduire la consommation énergétique pourraient potentiellement affecter négativement l'expérience de jeu des joueurs. Par exemple, une gestion d'intérêt trop restrictive pourrait entraîner des artefacts visuels désagréables, comme l'apparition soudaine d'objets ou de personnages lorsqu'ils entrent dans la zone d'intérêt d'un joueur. De même, l'introduction d'intermédiaires supplémentaires dans la chaîne de communication, comme notre « *broker* », pourrait théoriquement augmenter la latence globale du système.

Nos expérimentations n'ont pas observé l'affect sur les performances des scénarios testés, mais une évaluation plus complète de cet équilibre serait précieuse. Une telle étude pourrait intégrer des métriques de performance objectives (latence, fréquence d'images par seconde, etc.) ainsi que des évaluations subjectives de l'expérience utilisateur, mises en relation avec les économies d'énergie réalisées. L'objectif serait d'identifier les configurations optimales permettant de maximiser l'efficience énergétique tout en maintenant une qualité d'expérience acceptable pour les joueurs.

Cette approche s'inscrirait dans une perspective plus large de « développement durable du jeu vidéo », visant à concilier les exigences environnementales avec les attentes des utilisateurs. Elle pourrait également inciter l'industrie à intégrer plus systématiquement des considérations d'efficience énergétique dans la conception de leurs moteurs de jeu et de leurs infrastructures réseau.

5.5 PERSPECTIVES D'AMÉLIORATION ET TRAVAUX FUTURS

Notre travail, bien qu'apportant des contributions significatives à la compréhension de l'impact du modèle « *Pub/Sub* » sur l'efficience énergétique des serveurs de jeux, laisse entrevoir plusieurs pistes d'amélioration et d'investigation complémentaires.

Premièrement, notre implémentation actuelle du modèle « *Pub/Sub* » pourrait bénéficier de l'intégration des mécanismes d'optimisation déjà présents dans les solutions standards comme *Unreal Engine* (détection des valeurs identiques, compression des données vectorielles, etc.). Cette intégration permettrait d'évaluer plus précisément le potentiel réel d'amélioration de l'efficience apporté par le modèle « *Pub/Sub* » dans un contexte d'optimisation maximale.

Deuxièmement, nos expérimentations se sont limitées à un environnement de jeu simplifié et contrôlé. Une évaluation dans le contexte d'un jeu réel, avec des mécaniques de gameplay

complexes et un nombre plus élevé de joueurs, apporterait des informations précieuses sur l'élasticité et l'applicabilité pratique de notre approche. Cette évaluation pourrait également intégrer une plus grande variété de scénarios de jeu et de configurations spatiales, reflétant la diversité des situations rencontrées dans les « *MMOGs* ».

Troisièmement, une analyse plus fine de la relation entre le volume de données transmises et la consommation énergétique effective des infrastructures de jeu complètes (incluant serveurs de jeu, « *broker* », routeurs, et autres composants réseau) permettrait de valider plus rigoureusement les bénéfices environnementaux de notre approche. Cette analyse pourrait s'accompagner d'une évaluation de l'empreinte carbone globale des différentes architectures, prenant en compte non seulement la consommation énergétique directe, mais aussi l'énergie grise associée au matériel nécessaire.

Enfin, l'exploration de modèles hybrides, combinant les forces des approches « *C/S* », « *P2P* » et « *Pub/Sub* », pourrait ouvrir des voies prometteuses vers des architectures encore plus efficientes. Par exemple, un système dans lequel l'autorité du jeu serait dynamiquement répartie entre les joueurs en fonction de leur contexte géographique et de leurs capacités matérielles, avec un « *broker* » centralisé assurant la coordination des communications, pourrait potentiellement améliorer à la fois l'élasticité et l'efficience énergétique du système.

En conclusion, notre recherche a démontré le potentiel du modèle de communication « *Pub/Sub* » pour améliorer l'efficience énergétique des serveurs de jeux massivement multijoueurs. Bien que nos résultats soient prometteurs, des questionnements subsistent et des améliorations sont envisageables. Nos travaux constituent une base solide pour l'intégration des préoccupations environnementales dans la conception des architectures réseau des « *MMOGs* ». Cette perspective, encore peu explorée dans la littérature scientifique, pourrait contribuer significativement à la réduction de l'empreinte écologique du secteur numérique, tout en maintenant la qualité de l'expérience offerte aux joueurs.

CONCLUSION

Dans le cadre de cette recherche, nous avons exploré l'impact du modèle de communication « *Pub/Sub* » sur l'efficience du traitement des données dans les jeux vidéo massivement multijoueurs en ligne. Cette démarche s'inscrit dans un contexte dans lequel les préoccupations environnementales liées au secteur numérique deviennent de plus en plus prégnantes, faisant de l'informatique verte une nécessité incontournable.

Notre question de recherche principale était la suivante : un modèle de communication de type « *Pub/Sub* » peut-il contribuer à réduire la consommation énergétique des serveurs de jeux vidéo massivement multijoueurs tout en maintenant la qualité de service ? Pour y répondre, nous avons structuré notre démarche en plusieurs étapes complémentaires, allant de l'établissement des fondements théoriques à l'expérimentation pratique.

Dans un premier temps, nous avons établi les prérequis nécessaires à la compréhension de notre recherche. Nous avons ainsi présenté les fondements de l'informatique verte, en détaillant son évolution historique et ses différentes phases d'application à travers le cycle de vie des équipements informatiques. Nous avons ensuite abordé les spécificités des jeux massivement multijoueurs en ligne, en mettant en évidence les défis techniques qu'ils impliquent, notamment en matière d'architecture réseau. L'étude des différents modèles de communication (*DR*, *TR* et « *Pub/Sub* ») nous a permis de souligner les avantages potentiels du modèle « *Pub/Sub* », particulièrement en ce qui concerne la gestion ciblée des échanges d'informations. Enfin, nous avons démontré, tant sur le plan théorique qu'empirique, la relation directe entre le volume de données traitées par les équipements réseau et leur consommation énergétique, établissant ainsi une base solide pour notre hypothèse principale.

Notre revue de l'état de l'art a révélé que plusieurs solutions hybrides combinants différents modèles de communication (« *C/S* », « *P2P* », « *Pub/Sub* ») ont été proposées pour améliorer les performances des « *MMOGs* ». L'analyse comparative des travaux de Cecin (*FreeMMG 2*) [43], Carter (*NAHPALM*) [48], Shepelenko (architecture de super-nœuds) [46], Shongwe (architecture P2P) [49] et Cañas (variantes du modèle « *Pub/Sub* ») [19] a mis en évidence leurs forces et leurs faiblesses respectives. Nous avons constaté que les solutions basées sur le « *P2P* » offraient généralement une bonne latence, mais au prix d'une consommation de bande passante élevée, limitant leur accessibilité aux joueurs les plus limités sur ce point. Les approches intégrant le modèle « *Pub/Sub* » présentaient quant à

elles un potentiel intéressant en matière de gestion dynamique des flux d'informations, bien que leur élasticité reste à améliorer. Cependant, nous avons identifié une lacune générale dans la littérature concernant l'évaluation précise de l'efficience énergétique de ces solutions, soulignant ainsi la pertinence de notre recherche.

Face à ces constats, nous avons proposé une contribution en deux volets. Premièrement, nous avons formulé une nouvelle définition scientifique du terme « *MMOG* », fondée sur des critères objectifs, vérifiables et invariables. Cette définition stipule que les « *MMOGs* » sont des jeux vidéo qui emploient des mécanismes d'élasticité contrôlés en interne, indépendants de la volonté des joueurs, pour s'adapter au nombre de joueurs connectés. Nous avons également introduit une distinction entre les « *MMOGs* » à haute densité et à faible densité, en fonction de leur mode de distribution des joueurs. Cette clarification conceptuelle constitue une avancée significative pour la recherche future dans ce domaine.

Deuxièmement, nous avons développé une expérimentation visant à comparer les performances d'une solution de réplication standard (basée sur le système natif d'*Unreal Engine*) avec une implémentation personnalisée fondée sur le modèle « *Pub/Sub* ». Notre analyse préliminaire a révélé que la réplication des propriétés des objets constituait la majorité du trafic réseau dans un jeu multijoueur typique, représentant plus des deux tiers du volume total des données échangées. Cette observation a orienté notre protocole expérimental, qui a consisté à mesurer la bande passante consommée par les deux solutions dans différentes configurations spatiales simulant diverses situations de jeu.

Les résultats de nos expérimentations ont confirmé notre hypothèse initiale : le modèle de communication « *Pub/Sub* » permet effectivement de réduire significativement la consommation de bande passante des serveurs de jeux par rapport aux solutions traditionnelles, tout en maintenant un niveau de synchronisation adéquat entre les clients. Cette réduction s'explique principalement par la capacité du modèle « *Pub/Sub* » à filtrer et à cibler précisément les données transmises, en envoyant uniquement les informations pertinentes aux entités qui en ont réellement besoin. Compte tenu de la relation établie entre le volume de données traitées et la consommation énergétique des équipements réseau, ces résultats suggèrent que l'adoption du modèle « *Pub/Sub* » pourrait contribuer de manière substantielle à l'amélioration de l'efficience énergétique des serveurs de jeux massivement multijoueurs.

En conclusion, ce mémoire apporte une contribution significative à la recherche sur l'informatique verte appliquée aux jeux vidéo massivement multijoueurs en ligne. Nos travaux

démontrent que des choix judicieux en matière d'architecture de communication peuvent permettre de réduire l'empreinte environnementale de ces applications, sans compromettre leur qualité de service. La définition rigoureuse du terme « *MMOG* » que nous proposons offre également un cadre conceptuel solide pour les recherches futures dans ce domaine.

Plusieurs perspectives s'ouvrent à l'issue de ce travail. Il serait notamment intéressant d'étendre notre expérimentation à des jeux plus complexes et à des situations de charge plus variées, afin de confirmer la robustesse de nos résultats dans des contextes plus proches des conditions réelles d'exploitation. L'exploration de modèles « *Pub/Sub* » optimisés spécifiquement pour les différentes catégories de« *MMOGs* » (haute et faible densité) constitue également une piste prometteuse. Enfin, une quantification plus précise des économies d'énergie réalisables grâce à l'adoption du modèle « *Pub/Sub* » dans différentes configurations d'infrastructure serait précieuse pour orienter les choix technologiques des développeurs et des opérateurs de jeux.

À l'heure où les préoccupations environnementales prennent une importance croissante dans tous les secteurs d'activité, nos travaux soulignent l'importance d'intégrer des considérations d'efficience énergétique dès la conception des architectures de communication des jeux vidéo massivement multijoueurs. Cette approche proactive, visant à optimiser l'utilisation des ressources tout en préservant l'expérience utilisateur, représente une voie prometteuse pour concilier le développement du secteur du jeu vidéo avec les impératifs de durabilité environnementale.

BIBLIOGRAPHIE

[1] H. Ferreboeuf, M. Efoui-Hess, et X. Verne, “Environemental impacts of digital technology : 5-year trends and 5G governance,” The shift project, Rapport Technique.

[2] J. Hilchie, “Amateurs de jeux vidéo au Canada Faits essentiels 2020,” Association canadienne du Logiciel de Divertissement, Rapport Technique. [En ligne]. Repéré à : https://theesa.ca/wp-content/uploads/2022/11/RCGEF_fr.pdf

[3] S. Murugesan, “Harnessing Green IT : Principles and Practices,” vol. 10, nº 1, pp. 24–33. [En ligne]. Repéré à : https://ieeexplore.ieee.org/abstract/document/4446673?casa_token=IbSnvNrWJaMAAAAA:KE7fxm-XiYolutJ3pZgRJgP5cOOnGC_aTEEox6jpKf6A9OkysIDyB0ol8K2k4S_TYiEAmJHKvbpj

[4] R. Kennedy et J. Waltzer, *Monopoly : The Story Behind the World’s Best-Selling Game*. Gibbs Smith, 2004.

[5] H. A. Davidson, *A short history of chess*. Crown, 2012.

[6] J. Gregory, *Game Engine Architecture*, third edition éd. A K Peters/CRC Press.

[7] S. P. Raja, “Green Computing : A Future Perspective and the Operational Analysis of a Data Center,” vol. 9, nº 2, pp. 650–656. [En ligne]. Repéré à : https://ieeexplore.ieee.org/abstract/document/9481183?casa_token=7iebaoZ8QecAAAAA:K3L0yr47GyoyH2q9wJXzUa09jLhRQNJSBVgLFaTkW5S512OIgZp6SIGgyUK-ihsKN8RT1ug7csY

[8] B. Saha, Dr. B.C Roy, et M. Abul kalam Azad, “Green Computing Current Research Trends,” vol. 6, nº 3, pp. 467–469. [En ligne]. Repéré à : http://www.ijcseonline.org/full_paper_view.php?paper_id=1830

[9] G. Boyd, E. Dutrow, et W. Tunnessen, “The evolution of the ENERGY STAR® energy performance indicator for benchmarking industrial plant manufacturing energy use,” vol. 16, nº 6, pp. 709–715. [En ligne]. Repéré à : https://www.sciencedirect.com/science/article/pii/S0959652607000601

[10] The immersion cooling technology : Current and future development in energy saving -

ScienceDirect. [En ligne]. Repéré à : https://www.sciencedirect.com/science/article/pii/S1110016822001557

[11] P. Kurp, "Green computing," vol. 51, nº 10, pp. 11–13. [En ligne]. Repéré à : https://dl.acm.org/doi/10.1145/1400181.1400186

[12] J. Han, H. Ijuin, Y. Kinoshita, T. Yamada, S. Yamada, et M. Inoue, "Sustainability Assessment of Reuse and Recycling Management Options for End-of-Life Computers-Korean and Japanese Case Study Analysis," vol. 6, nº 3, p. 55. [En ligne]. Repéré à : https://www.mdpi.com/2313-4321/6/3/55

[13] D. Bulka et D. Mayhew, *Efficient C++ Performance Programming Techniques by Released November 1999*. Addison Wesley.

[14] J. F. Christofoli, "Authority distribution in a proxy-based massively multiplayer game architecture," Thèse de doctorat, Florida State University, 2006.

[15] D. Barkai, "An introduction to peer-to-peer computing," *Intel Developer update magazine*, pp. 1–7, 2000.

[16] H. S. Oluwatosin, "Client-server model," *IOSR Journal of Computer Engineering*, vol. 16, nº 1, pp. 67–71, 2014.

[17] L. Yang et P. Sutinrerk, "Mirrored arbiter architecture : a network architecture for large scale multiplayer games," dans *Proceedings of the 2007 summer computer simulation conference*, 2007, pp. 709–716.

[18] A. Tveit, Rein, J. Iversen, et M. Matskin, "Scalable Agent-Based Simulation of Players in Massively Multiplayer Online Games," *International conference on telecommunication*, 2014.

[19] C. Cañas, K. Zhang, B. Kemme, J. Kienzle, et H.-A. Jacobsen, "Publish/subscribe network designs for multiplayer games," dans *Proceedings of the 15th International Middleware Conference*, ser. Middleware '14. Association for Computing Machinery, pp. 241–252. [En ligne]. Repéré à : https://dl.acm.org/doi/10.1145/2663165.2663337

[20] A. Sharma. Understanding Request And Response Model : A General Overview. Medium. [En ligne]. Repéré à : https://medium.com/@imakashsharma135/understanding-request-and-response-model-a-general-overview-831c24d2288

[21] "Transmission Control Protocol," RFC 793, septembre 1981. [En ligne]. Repéré à : https://www.rfc-editor.org/info/rfc793

[22] J. Postel, "Ietf.org/rfc/rfc768.txt," https ://www.ietf.org/rfc/rfc768.txt.

[23] What is Pub/Sub ? The Publish/Subscribe model explained. [En ligne]. Repéré à : https://ably.com/topic/pub-sub

[24] M. Perry, *MQSeries Publish/Subscribe Applications*. Redbooks. [En ligne]. Repéré à : https://www.redbooks.ibm.com/abstracts/www.redbooks.ibm.com

[25] N. Deakin, "Java Message Service," 2012.

[26] Apache Kafka. Apache Kafka. [En ligne]. Repéré à : https://kafka.apache.org/documentation/

[27] Pulsar Overview | Apache Pulsar. [En ligne]. Repéré à : https://pulsar.apache.org/docs/3.3.x/concepts-overview/

[28] K. D. Foote. A Brief History of the Internet of Things. DATAVERSITY. [En ligne]. Repéré à : https://www.dataversity.net/brief-history-internet-things/

[29] IoT connections worldwide 2022-2033. Statista. [En ligne]. Repéré à : https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[30] D. Soni et A. Makwana, "A SURVEY ON MQTT : A PROTOCOL OF INTERNET OF THINGS(IOT)," *International conference on telecommunication*, 2017.

[31] W. Gan, J. C.-W. Lin, H.-C. Chao, et J. Zhan, "Data mining in distributed environment : A survey," vol. 7, n° 6, p. e1216. [En ligne]. Repéré à : https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1216

[32] P. T. Eugster, P. A. Felber, R. Guerraoui, et A.-M. Kermarrec, “The many faces of publish/subscribe,” vol. 35, n° 2, pp. 114–131. [En ligne]. Repéré à : https://dl.acm.org/doi/10.1145/857076.857078

[33] V. Setty, R. Vitenberg, G. Kreitz, G. Urdaneta, et p.-u. family=Steen, given=Maarten, “Cost-Effective Resource Allocation for Deploying Pub/Sub on Cloud,” dans *2014 IEEE 34th International Conference on Distributed Computing Systems*, pp. 555–566. [En ligne]. Repéré à : https://ieeexplore.ieee.org/abstract/document/6888931

[34] B. Stroustrup, “What is object-oriented programming ?” vol. 5, n° 3, pp. 10–20. [En ligne]. Repéré à : https://ieeexplore.ieee.org/abstract/document/2020?casa_token=BntJNpUXhRgAAAAA:Mr0OdWwLyTuipWr7N3sc1b_xlFVMe1ILlxFEP9kvVl8cLiy35ruWVZLWGHFIrGOeXLrenS_DYfg

[35] E. Fidler, G. Li, et S. Mankovski, “The PADRES Distributed Publish/Subscribe System,” *8th International Conference on Feature Interactions in Telecommunications and Software Systems*, 2005.

[36] D. Tanvir Ahmed et S. Shirmohammadi, “A Dynamic Area of Interest Management and Collaboration Model for P2P MMOGs,” *2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, p. 8. [En ligne]. Repéré à : https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4700100

[37] IEA – International Energy Agency. IEA. [En ligne]. Repéré à : https://www.iea.org/search

[38] Transfert des paquets par les routeurs - Planification du développement du réseau dans Oracle® Solaris 11.2. [En ligne]. Repéré à : https://docs.oracle.com/cd/E56338_01/html/E53779/ipplan-43.html

[39] V. Eramo, M. Listanti, F. G. Lavacca, P. Iovanna, G. Bottari, et F. Ponzini, “Trade-off between power and bandwidth consumption in a reconfigurable xhaul network architecture,” *IEEE Access*, vol. 4, pp. 9053–9065, 2016.

[40] F. Kaup, “Energy-efficiency and performance in communication networks : Analyzing energy-performance trade-offs in communication networks and their implications on future network structure and management,” Thèse de doctorat, Université technique

de Darmstadt, 2017. [En ligne]. Repéré à : https://api.semanticscholar.org/CorpusID:33552087

[41] M. Zhang, C. Yi, B. Liu, et B. Zhang, “Greente : Power-aware traffic engineering,” *The 18th IEEE International Conference on Network Protocols*, pp. 21–30, 2010. [En ligne]. Repéré à : https://api.semanticscholar.org/CorpusID:524778

[42] J.-H. Huh, “Reliable user datagram protocol as a solution to latencies in network games,” *Electronics*, vol. 7, n° 11, p. 295, 2018.

[43] F. R. Cecin, “Peer-to-peer and cheat-resistant support for massively multiplayer online games.” [En ligne]. Repéré à : https://lume.ufrgs.br/handle/10183/131877

[44] C. J. Carter, A. El Rhalibi, et M. Merabti, “A novel scalable hybrid architecture for mmog,” dans *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. IEEE, 2013, pp. 1–6.

[45] C. Carter, A. El Rhalibi, et M. Merabti, “Analysis of a novel hybrid p2p architecture,” dans *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*. IEEE, 2014, pp. 508–513.

[46] S. Shepelenko, “Applying supernode architecture for scalable multiplayer computer game,” Thèse de doctorat, University of Tartu, 2017.

[47] B. Shongwe, “PEER-TO-PEER NETWORK ARCHITECTURE FOR MASSIVE ONLINE GAMING,” Mémoire de maîtrise, University of the Witwatersrand, 2014.

[48] C. J. Carter, A. El Rhalibi, et M. Merabti, “A novel scalable hybrid architecture for MMOG,” dans *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pp. 1–6. [En ligne]. Repéré à : https://ieeexplore.ieee.org/abstract/document/6618435?casa_token=FCQyT36Nw3sAAAAA:qlD6gOTkoC-ib5Nc1KQnPYNZD5tazRE7rPriLHBep9b3NZBkWXYhitbxc2YWx-9VTZYn8Cb6Jgg

[49] B. Shongwe, “Peer-to-peer network architecture for massive online gaming,” Thèse de doctorat, University of the Witwatersrand, Johannesburg, 2014.