Approche multi-agent pour la collaboration multi-plateforme dans un contexte de défense navale

P. Quentel[1,2], Y. Kermarrec[1], L. Grivault[2], P. Le Berre[2], L. Savy[2]

[1] IMT Atlantique, Lab-STICC, F-29238 Brest, France

[2] Thales Defence Mission Systems, Brest et Elancourt, France

Mai 2023

Résumé

Dans la littérature, l'allocation des tâches à diverses entités a été largement étudiée. Cette problématique apparait également dans le cas des études collaboratives des systèmes de senseurs navals où il est question de pouvoir allouer des tâches à des senseurs en se basant sur des services. Les caractéristiques d'autonomie, de communication, d'intelligence et de distribution des agents font du système multi-agent (SMA) un choix judicieux dans le cadre de l'affectation de tâches senseurs dans un réseau multi-plateforme. Cet article présente en premier lieu le cadre complexe du combat collaboratif naval. Ensuite, nous proposons des modèles d'agents : l'agent plateforme, l'agent service et l'agent tactique. Enfin, nous proposons de faire un point sur l'avancée pratique de nos concepts et sur les méthodes d'évaluation que nous souhaitons appliquer pour valider ces concepts au regard de nos besoins.

Mots-clés

Planification multi-agent, communication, systèmes multi-senseurs, négociations multi-agents

Abstract

In the literature, the allocation of tasks to various entities has been widely studied. This problem also appears in the case of collaborative studies of naval sensor systems, where the question is to be able to allocate tasks to sensors based on services. The characteristics of autonomy, communication, intelligence and distribution of the agents make the Multi-Agent System (MAS) a wise choice in the context of sensors task allocation in a multi-platform network. Firstly, this paper presents the complex framework of naval collaborative combat. Then, we propose agent models : the platform agent, the service agent and the tactical agent. Finally, we propose to show the practical progress of our concepts and the evaluation methods we wish to apply to validate these concepts with respect to our needs.

Keywords

Multi-agent planning, communication, multi-sensor systems, multi-agent negotiations

1 Introduction

Cette introduction permet de comprendre les enjeux liés au contexte d'application des travaux présentés et de saisir la problématique qui en résulte.

1.1 Contexte

L'évolution du contexte de défense aéronaval nécessite une modification majeure des architectures des systèmes de senseurs. Depuis dix ans, la direction générale de l'armement (DGA) a amorcé des recherches et financé des programmes dans le but d'améliorer les capacités opérationnelles des systèmes de combat. De nos jours, les systèmes de senseurs coopèrent, les données des senseurs sont partagées entre les plateformes par le système de gestion de combat (CMS : *Command Mission System*). Le CMS de chaque plateforme permet la gestion des senseurs locaux ainsi que le suivi des pistes et des objets de la zone à surveiller. Chaque plateforme maintient une situation tactique locale à l'aide de ses senseurs, ainsi qu'une situation tactique globale grâce à l'échange d'information avec les autres plateformes au moyen de liaisons de données tactiques (LDT).

Depuis quelques années, les senseurs tendent à devenir des systèmes complexes [20], capables de partager de la donnée, de communiquer et de collaborer. Dans le cadre de notre étude, nous nous intéressons aux senseurs de guerre électronique (GE), mais aussi aux radars et aux capteurs optroniques. Nous détaillerons la notion de *senseur* en section 3. Les besoins technico-opérationnels de notre architecture concernent la détection, la localisation et le pistage de menaces.

De multiples schémas de partage de l'information ont été implémentés dans des systèmes de senseurs aéroportés. Cependant, d'après Beal et al. [1], l'affectation de tâches à des senseurs dans le but de collecter ces informations est difficile et doit, en général, être réalisée manuellement.

1.2 Problématique

L'intelligence artificielle distribuée (IAD) est une branche de l'IA utilisée pour la résolution de problèmes complexes qui, souvent, ne peuvent pas être gérés par un système centralisé [11]. Les algorithmes d'IAD sont classés en trois catégories : l'IA parallèle, la résolution de problèmes distri-

bués (DPS) et les systèmes multi-agents (SMA) [5]. Wooldridge, dans son ouvrage [26] qui introduit les SMA, caractérise l'agent BDI (*Belief, Desire, Intention*), purement communicant, comme un système informatique qui est situé dans un environnement, et qui peut exécuter des actions sur l'environnement avec autonomie, ceci afin d'accomplir un objectif défini. Le système multi-agent comprend des agents intelligents qui peuvent communiquer entre eux à l'aide de protocoles de communication.

Dans le cadre du combat collaboratif naval, nous souhaitons concevoir une architecture qui associe des senseurs hétérogènes de différentes plateformes pour effectuer des actions collaboratives, et ce de manière distribuée. La recherche sur les SMA met en avant les agents intelligents comme métaphore naturelle pour traiter de la résolution distribuée de problèmes complexes [13]. Ainsi, l'approche multi-agent rendrait possible l'allocation dynamique de tâches à des senseurs en réseau dans un contexte de collaboration multi-plateforme et multi-senseur. Cet article présente des concepts d'agents permettant de remplir cet objectif.

Les articles [16, 22, 23, 24, 25] ont étudié la problématique de l'allocation de tâches à des agents, cependant dans notre cas ce sont les agents qui proposent des tâches à nos senseurs qui sont alors considérés comme des ressources. L'allocation est un procédé qui fait correspondre des tâches à un ensemble de senseurs et qui tente de maximiser une utilité globale. L'objectif de nos travaux est d'optimiser l'utilisation des senseurs dans un contexte dense en nombre de menaces, il est donc nécessaire de sélectionner les tâches qu'ils effectueront en considérant des critères et des priorités.

Dans cet article, nous définissons le contexte dans lequel s'applique le besoin d'allocation de tâches à des senseurs, puis nous présentons des concepts d'agents permettant de répondre à ce besoin.

La section 2 de l'article présente des travaux relatifs à l'architecture des systèmes de senseurs et à l'allocation de tâches dans les SMA. Ensuite, nous formalisons le problème de l'allocation de tâches dans le contexte de défense naval en section 3. La section 4 présente notre conception des agents. Le développement de ces concepts et les méthodes choisies pour les évaluer sont présentés en section 5. Enfin, nous concluons en section 6 sur les travaux effectués et détaillons quelques pespectives à nos travaux.

2 Travaux connexes

Des travaux sur l'architecture des systèmes de senseurs et sur l'allocation de tâches dans les SMA sont présentés dans cette section.

2.1 Architecture

Actuellement, dans le domaine du combat aéronaval, chaque senseur est associé à une ou plusieurs fonctions qu'il peut accomplir. Dans l'architecture existante, les fonctions sont utilisées indépendamment les unes des autres et leur coordination est laissée à une initiative humaine.

Un travail de thèse sur l'architecture pour l'ordonnancement de systèmes multi-senseurs [7] utilise les concepts des SMA. Un agent est créé lors de la détection d'un objet sur le théâtre des opérations. Celui-ci assure le suivi et la collecte des données relatives à l'objet détecté (sa position, sa vitesse, etc.). Ces agents représentent la situation tactique globale et proposent des planifications de tâches senseurs dans le temps. Cette architecture est fonctionnelle dans un cadre mono-plateforme, notre approche étend cette solution à un ensemble de plateformes multi-milieux (aérien, terrestre et maritime).

2.2 L'utilisation des SMA pour l'allocation de tâches

Beal et al. [1] décrivent un système à base d'agents pour allouer des tâches aux senseurs dans le but de réduire le nombre de plateformes (ici des drones) pour accomplir un objectif. Des concepts d'agents sont présentés ainsi qu'un algorithme d'allocation de tâches.

Les auteurs proposent deux types d'agents : les agents *plateformes* et les agents *tâches*. Un agent *plateforme* est créé pour chaque ressource d'une plateforme (hypothèse : une seule ressource par plateforme). La particularité de cet agent est qu'il se décline en plusieurs agents *projetés*, en prenant en compte la position future anticipée de la plateforme. Un agent est créé pour chaque tâche et il communique avec les agents *plateformes* à portée de communication pour déterminer quelle plateforme accomplira la tâche. L'algorithme d'allocation de tâches prend en compte deux métriques afin de classer les tâches proposées : le nombre de plateformes qui couvrent déjà la tâche et la priorité de la tâche. Un budget correspondant au temps de disponibilité du senseur est fixé, les tâches assignées sont prises dans l'ordre d'importance et dans la limite de ce budget. Les travaux restent tout de même limités au partage de la ressource de senseurs hautement directifs (caméras embarquées par exemple). Les auteurs démontrent qu'il est possible d'utiliser moins de plateformes pour effectuer le même nombre de tâches en se servant uniquement du partage de la ressource disponible. Les senseurs ne collaborent donc pas dans le but de proposer de nouvelles fonctionnalités.

Ponda et al. utilise l'allocation de tâches afin que des agents hétérogènes dans un environnement dynamique puissent effectuer des missions [16]. Ces missions impliquent l'exécution de différentes tâches comme la reconnaissance, la surveillance, la classification de cibles, et les opérations de secours. Certains agents, des UAV (*Unmanned Aerial Vehicules*), peuvent effectuer des tâches différentes. Pour des cas avec un nombre important d'agents, les approches centralisées deviennent rapidement impossibles à mettre en œuvre en raisons des conflits d'allocations, de la complexité et du nombre d'interactions, il est donc nécessaire d'adopter des architectures décentralisées.

Dans l'article [25], le problème d'allocation de tâches s'intéresse plus particulièrement à une tâche unique, avec un seul robot, et des affectations à durée prolongée. Un agent effectue une tâche à la fois, et chaque agent peut se voir affecter plusieurs tâches dans un planificateur. Trouver la solution optimale à cette allocation de tâches dans un environnement temps-réel devient infaisable au niveau des calculs

à mesure que le nombre de tâches et d'agents augmente.
Des approches pour l'allocation de ressources sont étudiées dans [4], le modèle d'agent considéré suit trois sous-comportements (*acting*, *communicating* et *planning*). L'article soulève le problème du goulot d'étranglement dû aux communications, point névralgique de notre approche. De plus dans un système dynamique le problème d'allocation de ressources varie au cours du temps rendant impossible la résolution du problème de manière optimale. Dans l'approche proposée par les auteurs, la communication des agents n'est possible que si les agents font partie du même ensemble connecté, par message direct ou par diffusion « broadcast ».
Dans le cadre de la chasse aux mines sous-marines, Milot et al. [15] proposent une approche décentralisée d'allocation de tâches multirobots par enchère. La mission est composée de trois objectifs : détection, identification et neutralisation. Les robots se retrouvent en concurrence pour « acheter » des tâches lors d'enchères.

2.3 Méthodes d'allocation de tâches dans les SMA

Les articles [22, 23, 27] présentent un état de l'art sur les méthodes d'allocation de tâches dans les SMA. La performance optimale d'un système global est un concept qui dépend de la perception de chaque agent et des contraintes du système. De ce fait le terme « utilité », souvent utilisé dans d'autres travaux, représente une valeur ou un coût affilié à cette allocation de tâches.
Il existe de nombreuses méthodes d'allocation [23] : les enchères ou marchés ; la théorie des jeux ; les techniques basées sur l'optimisation qui englobent les heuristiques ou DCOP (*Distributed Constraint Optimization Problems* [4]), les optimisations déterministes (algorithme hongrois) ou les métaheuristiques (*PSO, Bees algorithm, ant colony*) ; les approches orientées apprentissages (*MARL : Multi-Agent Reinforcement Learning*) ; et enfin les approches hybrides qui combinent plusieurs stratégies.
Nous nous intéressons plus particulièrement aux méthodes qui semblent le mieux répondre à nos objectifs, à savoir les algorithmes d'enchères (*auction-based*) qui permettent l'allocation de tâches de manière décentralisée et flexible. Les méthodes d'enchères sont efficaces bien que non optimales, et la mise à l'échelle est possible, car les coûts en calcul et en communication sont modérés [23]. Ces caractéristiques sont importantes pour nos besoins et contraintes détaillés dans la section 3.2. Dans les méthodes par enchère, les agents misent sur des tâches, la mise la plus élevée remporte l'allocation de la tâche. La méthode habituelle pour déterminer le vainqueur est d'avoir un « commissaire-priseur » qui reçoit et évalue les mises centralement. Deux algorithmes par enchère sont proposés par Choi et al. [3], le CBAA (*Consensus-Based Auction Algorithm*) et le CBBA (*CB Bundle Algorithm*). Le premier répond au problème d'allocation unique, tandis que le second correspond à l'allocation multiple. Le CBBA est un algorithme décentralisé qui utilise d'abord une heuristique gloutonne pour sélectionner les tâches, puis il applique un consensus pour supprimer le problème du chevauchement de tâches. Il existe d'autres algorithmes, notamment des variantes du CBBA, mais aussi le CNP (*Contract Net Protocol*) qui est un protocole standardisé permettant d'allouer ainsi que de réassigner des tâches à des agents.
L'approche du CBBA permet une décision décentralisée, nécessaire lorsqu'il y a de nombreux échanges dans de grandes équipes d'agents. De plus, cet algorithme est polynomial en temps de calcul ce qui lui permet une mise à l'échelle facilitée avec le nombre de tâches ou la taille du réseau qui augmentent. Enfin, des objectifs de conception différents, des modèles d'agents et des contraintes peuvent être incorporés en définissant des fonctions de scores appropriées. Le besoin en synchronisation du CBBA rend son utilisation dans des applications temps réel moins efficace, les mêmes auteurs tentent de compenser cet inconvénient avec le ACBBA (Asynchronous CBBA), une extension de l'algorithme existant [12]. Le comportement des agents que nous proposons se rapproche des algorithmes par enchères, particulièrement le CBBA, qui devra être adapté à nos besoins et contraintes.

3 Formulation du problème

Cette section a pour but de présenter le cadre des études afin de comprendre l'approche menée et les choix effectués.

3.1 Définitions

Définition 1. *Une plateforme correspond à différentes entités militaires comme des bâtiments de surface ou des aéronefs. Elle est caractérisée par sa position géographique (longitude, latitude et altitude), sa vitesse, ses communications et les senseurs qu'elle possède définissant ainsi ses capacités.*
Définition 2. *Un senseur (communément appelé capteur) est un instrument qui détecte et répond à certaines entrées provenant d'un environnement [24]. Il est caractérisé par sa portée de détection, son type de senseur et les services qu'il propose.*
Dans nos études, les senseurs récupèrent des données électromagnétiques dans un théâtre d'opération. Nous considérons trois types de senseurs différents, proposant des capacités propres. Les senseurs de GE comme le RESM (*Radar Electronic Support Measure*) ont la capacité de détecter des signaux radar en restant discrets, ils peuvent aussi localiser et identifier des cibles. Les radars comme les FCR (*Fire Control Radar*) sont des senseurs actifs capables de détecter, de localiser et de classifier des cibles aériennes ou de surfaces en fournissant des données de distance, de direction et de vélocité [14]. Enfin, les senseurs optroniques sont utilisés pour faire de la veille infrarouge (ou IRST : *Infra-Red Search and Track*) qui consiste à repérer et localiser des menaces par leur chaleur. Chaque senseur ne rempli qu'une partie des informations concernant une cible. Par exemple, le radar fournit la position et la distance de la cible, la GE permet d'obtenir des droites de visée et ne collecte des informations de distance que par défilement (déplacement de la plateforme réceptrice). L'optronique ne mesure pas la

distance mais permet de recueillir les informations de position et d'identification.

Définition 3. *Une tâche représente une réservation de ressource pendant un intervalle de temps. Une tâche peut avoir des contraintes de précédence si une autre tâche doit être effectuée avant celle-ci (voir définition 5 plan senseur).*

Il existe des tâches mono-senseurs et multi-senseurs tout comme il y a des allocations instantanées ou prévues dans le temps. De plus, un senseur peut lui aussi effectuer une seule ou plusieurs tâches simultanément. Nous nous limiterons aux tâches instantanées dans un premier temps.

Définition 4. *Une ressource est nécessaire à l'exécution d'une tâche. Dans un cadre général, les ressources sont variées : des données, des capacités de calculs, de la mémoire, de l'énergie, ou d'autres entités [2].*

Les senseurs représentent les ressources principales de nos travaux, cependant, pour qu'ils puissent fonctionner, ils ont aussi besoin de ressources de calcul, de fréquences de fonctionnement, de mémoire, etc. Dans un premier temps, nous nous limiterons aux senseurs sans nous préoccuper des autres attributs des ressources.

Définition 5. *Un plan senseur P_k [7] est composé d'un ensemble de tâches T_i, avec des contraintes les liants entre elles, affectées à une ou plusieurs ressources. (voir Figure 1)*

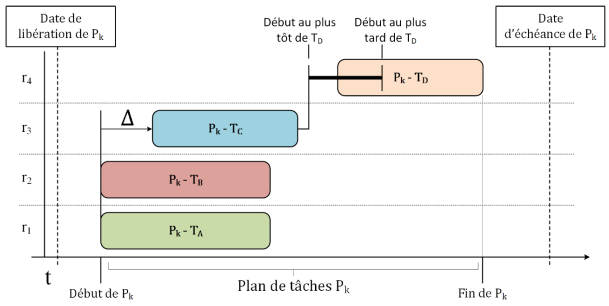

FIGURE 1 – *Schéma d'un plan senseur*, de [7] p.110

La plupart des plans senseurs ne comprennent qu'une seule tâche, mais d'autres peuvent en nécessiter plusieurs, par exemple : une tâche d'acquisition, puis une tâche de traitement, les deux ne pouvant être effectuées simultanément.

Définition 6. *Un service senseur est une capacité (ou une fonction) que proposent un ou plusieurs senseurs [18]. Les senseurs déclarent les services qu'ils peuvent accomplir dans un registre qui sera ensuite partagé à des consommateurs de services. Le service est caractérisé par une qualité de service (QoS), des règles, et des ressources qui permettent son utilisation. Le service répond à un besoin de l'utilisateur sans que celui-ci n'ai besoin de connaître son fonctionnement.*

Le modèle d'architecture à base de services permet l'intégration de nouveaux services avec des mécanismes de découverte de services. Les services peuvent évoluer ou être modifiés. Dans notre architecture, un service est décrit par un identifiant, une performance, un temps d'utilisation et des ressources dont il a besoin pour fonctionner. Ainsi, différentes ressources peuvent compléter un même service. Le choix du service s'effectuera donc en considérant sa faisabilité en fonction de critères tels que la position ou les performances des ressources, mais aussi la précision des mesures.

Définition 7. *Une requête opérationnelle est issue d'un ordre de mission interne ou externe à la plateforme. Cette requête décrit génériquement les besoins durant une opération, elle comprend notamment le service opérationnel, la zone géographique d'application, la durée effective de la requête, les contraintes, les priorités, la qualité de service désirée, etc.*

La localisation ou l'identification, deux fonctionnalités des senseurs vus dans la *Définition 2*, sont désignées comme des services opérationnels (ou hauts niveaux). Une requête de service opérationnelle de localisation discrète pourra par exemple être accomplie par différents services senseurs comme la TDOA (*Time Difference of Arrival*) ou la FDOA (*Frequency Difference of Arrival*) tout deux présentés dans [21]. Ces services permettent la géolocalisation passive, le radar ne sera alors pas utilisé, car contraint par un choix opérationnel de discrétion.

3.2 Besoins du système et métriques d'évaluation

Notre système multi-agent, en plus de répondre à la problématique d'allocation de tâches, devra montrer qu'il propose une solution sous certaines contraintes et avec des besoins définis :

- **Mise à l'échelle :** Le système doit pouvoir supporter une certaine charge, l'augmentation du nombre de plateformes et donc de senseurs impactera le nombre d'agents créés, il faut donc déterminer les limites de l'approche proposée ;
- **Contraintes temporelles :** En raison du contexte d'utilisation (le combat collaboratif), l'allocation de tâches devra être rapide, le temps mis pour obtenir une allocation de tâches effective devra être mesuré ;
- **Robustesse et résilience :** Le système doit s'adapter si des senseurs ou des plateformes sont ajoutés ou retirés, ce qui implique des changements sur l'allocation de tâches. Cela peut se mesurer en testant d'ajouter ou de supprimer des plateformes, ou en simulant une défaillance d'un senseur ;
- **Communication :** La communication entre les agents implique une augmentation des échanges inter-plateformes, il est donc nécessaire de quantifier le besoin en bande passante avec l'approche proposée ;
- **Modularité et adaptabilité :** Le contexte multi-milieu et multi-plateforme nécessite un travail sur le modèle des agents, nous devons pouvoir proposer de nouveaux services qui s'intégreront dans la liste des services, et ce avec des plateformes variées ;
- **Efficacité :** Le système doit maximiser le nombre de tâches accomplies tout en s'assurant de la prise en compte des tâches les plus importantes.

4 Conception des agents

Dans le cadre de nos travaux, nous étendons les concepts d'agents expliqués par Grivault et al. [9]. L'agent a pour objectif principal de collecter le plus de données possible sur un objet unique du théâtre des opérations grâce à des senseurs. Dans cette optique, l'agent en question va proposer des plans qui seront envoyés à un ordonnanceur. L'ordonnanceur sera chargé de positionner temporellement les tâches (durée, début et fin de tâche, etc.). L'architecture proposée est centralisée, les agents sont situés sur la même plateforme et il n'y a pas de collaboration avec d'autres plateformes. La vision de l'architecture interne d'un agent reste similaire dans notre approche multi-agent. La distribution des agents sur les différentes plateformes constitue une proposition d'architecture permettant l'usage de services multi-plateformes, ce qui étend donc les travaux des auteurs précédemment cités.
La conception architecturale de l'agent est présentée plus en détail dans l'article [8]. Sa structure interne comporte une mémoire, une messagerie et un ensemble de fonctions. L'agent récupère des informations de son environnement. Il alimente ses fonctionnalités grâce à la base externe de connaissances qui contient les capacités et informations collectés par les senseurs, ainsi que les données opérationnelles (par exemple des renseignements connus sur une cible potentielle). La mémoire, initialement vide pour chaque agent générique, se complète par les informations qui proviennent de la plateforme, des requêtes opérationnelles issues de l'exécutif et enfin des senseurs actifs qui engrangent des données sur les cibles. Un mécanisme de communication est présent dans le modèle agent mais n'est pas explicité, nous proposons d'utiliser le protocole AMQP (*Advanced Message Queuing Protocol*).
Dans cet article, nous proposons trois types d'agents qui dérivent d'un agent générique. Ils comprennent une mémoire interne, des fonctions (initialisation de l'agent, démarrage de l'agent, mise à jour de la mémoire, etc.) et un système de communication utilisant RabbitMQ. Dans cette section, nous présentons trois concepts d'agents :

- L'agent *plateforme* (voir Définition 1) a pour rôle de maintenir la base de connaissance de la plateforme qu'il symbolise et des plateformes alliées. Il crée un agent *service* et il actualise sa situation tactique locale par la création et la mise à jour des agents *tactiques* ;
- L'agent *service* représente la liste des services que fournit le système (voir Définition 6), chaque plateforme en possède un. Il a pour objectifs de proposer des services disponibles et de planifier les services en échangeant des messages avec les agents *tactiques* ;
- L'agent *tactique* [9] exprime un objet du théâtre des opérations. Il est donc créé en réponse à la détection d'une piste « senseur ». Dans nos travaux, cet agent propose des plans senseurs à l'agent *service*.

Ensemble, ils permettent d'allouer des tâches à des senseurs dans un contexte multi-plateforme.

4.1 Agent *plateforme*

L'agent *plateforme* représente logiciellement une plateforme. Sa mémoire interne comprend des informations sur sa position, sur ses senseurs et leur disponibilité, sur les connexions avec les autres plateformes, etc. Nous supposons que la faculté de communication entre deux plateformes est vérifiée en amont. Nous supposons que les liaisons sont directes dans un réseau maillé (en réalité, la communication peut s'effectuer par saut, ce qui engendre une surconsommation de la bande passante et une augmentation de la latence).
Les interactions avec les autres agents s'effectuent avec le modèle *publish/subscribe* que fournit le protocole AMQP. Les mécanismes d'échanges que nous proposons se rapprochent de ceux utilisés pour la plateforme Triskell3S [19]. Triskell3S propose un protocole de communication commun entre tous les agents. Les agents d'une plateforme A peuvent alors communiquer avec des agents d'une plateforme B par le mécanisme de *Publish/Subscribe* de MQTT (Message Queuing Telemetry Transport). Ainsi, dans nos travaux, pour chaque échange nous créons un producteur sur l'agent émetteur et un consommateur sur l'agent récepteur. Par exemple dans le cas de partage des ressources disponibles, l'agent *plateforme* publie les ressources sur une clé de routage (*routing key*) « platform.resource » et il s'abonne à une clé de liaison (*binding key*) « *.resource ». Les échanges entre les agents sont présentés sur la figure 2, les messages d'une plateforme vers une autre transitent par le réseau tandis que les échanges internes ($Plateforme \leftrightarrow Service$ ou $Service \leftrightarrow Tactique$) se basent sur des données partagées.
L'agent *plateforme*, par des échanges avec ses pairs, met à jour la base de connaissance des ressources alliées. Nous nous intéressons principalement aux échanges entre des agents appartenant à des plateformes différentes, ainsi l'agent *service* pourra récupérer la liste des ressources directement depuis la plateforme à laquelle il appartient. De plus, les agents *plateformes* décident ensemble du choix des agents *tactiques* qu'ils prennent en charge. En effet, les plateformes possèdent une copie des données de tous les agents *tactiques*, mais un seul agent tactique concernant une même piste propose des plans. Cela permet d'avoir de la redondance et de ne pas perdre de données tactiques si une plateforme quitte le réseau.

4.2 Agent *service*

Pour chaque agent *plateforme*, un agent *service* est associé. Celui-ci possède la connaissance sur les services senseurs et leur disponibilité. Un service senseur a besoin de ressources pour fonctionner. L'agent *service* va donc récupérer la liste des ressources de l'ensemble des plateformes et créer une table interne de services uniques comme nous pouvons le voir dans la table 4.2. Cette table peut être élaborée en préparation de mission en référençant les services possibles, ils sont alors inutilisables par défaut et activés lorsque les plateformes et leurs ressources requises (RP_A dans la table pour « Ressource Plateforme A ») existent bien dans le réseau. Elle peut aussi être complétée en cours de

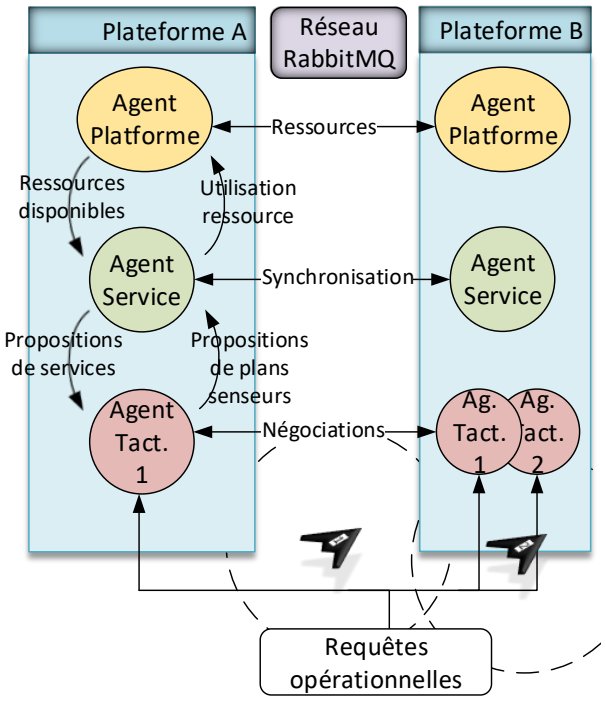

FIGURE 2 – Echanges entre agents

Service	Performance	Ressources	Durée
S1	580	RP_A, RP_B	10s
S2	450	RP_A, RP_C	10s

TABLE 1 – Connaissance de l'agent sur les services disponibles

mission, dans ce cas les plateformes doivent partager, en plus des ressources, les fonctionnalités de leurs senseurs qui permettront alors de les faire correspondre à des services senseurs. Par exemple, il faut au minimum deux senseurs de GE pour effectuer un service de TDOA, les plateformes capables d'effectuer ce service partagent sur le réseau leur capacité. L'agent *service* déclinera alors les choix possibles de ce service avec la durée nécessaire d'utilisation des ressources pour le mener à terme. Ainsi dans le cas de trois plateformes, il sera possible d'exploiter un service de TDOA utilisant les senseurs des plateformes A et B, ou bien les senseurs des plateformes B et C. Une performance est calculée selon plusieurs métriques comme la distance entre les plateformes ou les performances des senseurs.

L'agent *service* actualise à intervalle régulier sa table de services et partage ses mises à jour de services avec une performance acceptable à l'agent *tactique*. Celui-ci a également besoin de la durée d'utilisation de la ressource nécessaire à l'accomplissement du service.

Ensuite, il reçoit les propositions de plans senseurs des agents *tactiques* avec des scores associés et il planifie l'utilisation des services en réservant les meilleurs plans en fonction des ressources disponibles. L'agent *service* peut être vu comme un « commissaire-priseur » d'un algorithme de coordination par enchère, en effet il obtient des demandes d'allocation de manière centralisée tout en étant distribué sur différentes plateformes. Le score qu'il reçoit correspond donc à la mise que l'agent *tactique* lui propose. La phase de l'allocation locale (voir figure 4) se termine lorsque chaque agent tactique possède un plan d'accepté, ou lorsque la ressource est épuisée. La phase suivante, celle de l'allocation globale, consiste à demander l'activation de la ressource à l'agent plateforme. Afin de synchroniser les demandes de planification des agents services, une base de données distribuée existe au niveau de chaque plateforme. Chaque planification validée par l'agent service est envoyée sur un topic et mise en file d'attente FIFO pour validation auprès de la plateforme. Si le plan est finalement refusé, l'agent service partage le statut du plan à l'agent tactique qui retourne dans un état de planification. Un agent satisfait attend que son plan soit terminé pour en proposer de nouveaux. Un plan accepté peut aussi être refusé, si un meilleur plan qui utilise les mêmes ressources est proposé. Au cours du temps, l'agent service vérifie que les plans acceptés ne sont pas annulés et il incrémente le score des plans en cours pour éviter qu'ils soient annulés en fin d'utilisation du service.

4.3 Agent *tactique*

L'agent *tactique* est la représentation logicielle d'une piste (un objet ou une cible sur le théâtre d'opération). Sa mémoire interne comprend des informations sur la position, l'attitude et la vitesse de la cible, ses intentions pour compléter les connaissances de la piste et des informations d'identification (si disponibles). Les agents *tactiques* fonctionnent par requête opérationnelle, les besoins pour la mission vont influencer les choix de leurs plans. Par exemple, un opérateur a défini une zone de veille, si la menace détectée se situe hors zone alors l'agent *tactique* sera créé, mais inactif, et ce jusqu'à ce qu'une nouvelle requête lui ordonne le contraire.

L'agent *tactique* propose d'accomplir des objectifs opérationnels selon une suite logique nommée DRIL (Détection, reconnaissance, idenfication et localisation), pouvant être pris dans un ordre différent selon la situation. Ainsi, le choix des plans senseurs différera en fonction de l'objectif opérationnel en cours.

Il est possible que deux plateformes aient reçu la piste d'une même cible, donc deux agents *tactiques* sont créés pour un même objet (voir figure 2, où un objet situé entre les plateformes A et B amène à la création d'un agent *tactique* sur chaque plateforme). Dans la réalité, une corrélation entre les pistes doit être réalisée afin de déterminer qu'il s'agit bien de la même menace. Dans notre cas, nous supposons la corrélation comme parfaite en considérant que les plateformes connaissent a priori cette information.

L'agent *tactique* va étudier la faisabilité de plans senseurs en fonction des services proposés par l'agent *service*. Pour cela, il va calculer un score pour chaque plan qui sera valable pendant un court intervalle de temps. Ce score dépendra de la performance du service, de la position des plateformes possédant la ressource par rapport à la cible, de la priorité de la requête opérationnelle et des contraintes im-

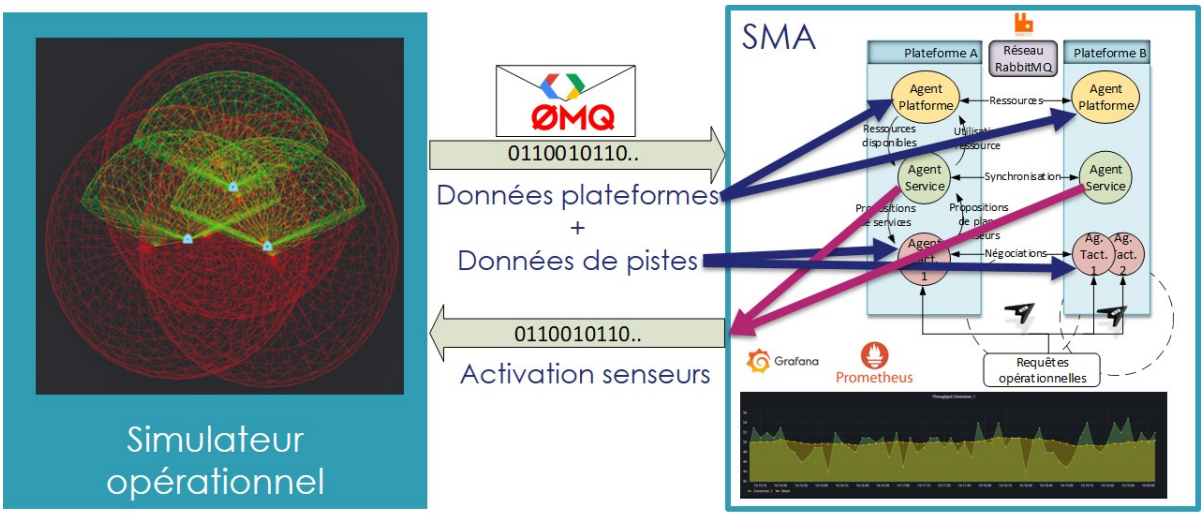

FIGURE 3 – Schéma de l'architecture globale

posées par cette requête (une requête avec contrainte de discrétion aura un score faible si les ressources proposées par le service ne sont pas discrètes également). Les plans senseurs qu'il soumettra comprendront : le service en question, le score calculé, le moment où le plan a été calculé, et les ressources nécessaires à sa réalisation. L'objectif de l'agent est de compléter ses données sur la cible afin d'améliorer sa connaissance tactique sur celui-ci. Dans ce but, il va fournir égoïstement (algorithme *glouton*) son ou ses meilleurs plans senseurs à l'agent *service*. Il ne se coordonne pas avec les autres agents avant d'envoyer ses plans et il ne sait pas si ses plans seront acceptés mais il maximise son propre score. Nous supposons que l'agent ne planifie qu'une utilisation de service à la fois, il ne pourra pas présenter un nouveau plan senseur tant que son plan n'est pas terminé ou annulé.

La figure 4 présente un exemple simplifié des échanges entre un agent *tactique* et un agent *service*.

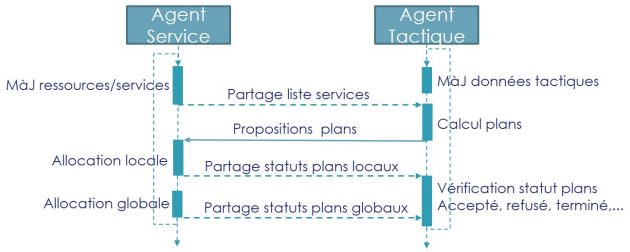

FIGURE 4 – Exemple simplifié d'un échange entre un agent tactique et un agent service

5 Expérimentations et méthodes d'évaluation

Dans la section précédente, nous avons présenté des concepts d'agents afin de répondre à la problématique de l'allocation de tâches à des senseurs. Dans cette partie, nous présentons les expérimenations et les méthodes d'évaluation dans le but de valider l'architecture multi-agent proposée.

Pour introduire cette partie, nous allons présenter un simulateur que nous utilisons. Ce simulateur, développé en C++, exploite le moteur de jeux vidéos *Unreal Engine 5*. Il permet la création de scénarios mettant en jeu plusieurs plateformes, il est possible de les ajouter et de les faire se déplacer, tout en activant/désactivant divers senseurs. L'exécution du scénario nous permet de récupérer des données des plateformes et des pistes qu'ils détectent. Avec un interfaçage combinant ZeroMQ [10] et le langage de description d'interface *Protocol Buffers*, nous pouvons récupérer ces précieuses données simulées dans le but de faire fonctionner notre logiciel multi-agent, qui est un logiciel externe au simulateur. Cependant, nous nous limiterons dans un premier temps à des données factices en envoyant des données chiffrées avec ZMQ au cours de la simulation, créant ainsi des agents. Le schéma de l'architecture globale est présenté en figure 3.

5.1 Développement de l'outil de validation

Les agents sont en cours de développement en langage Java. L'agent est représenté par un *thread* et ses actions suivent une machine à états finis. Par exemple, l'agent *plateforme* suit plusieurs états comme la mise à jour des données de la plateforme ou la mise à jour des agents tactiques.

Le récepteur ZeroMQ est exécuté sur un *thread* dédié, il réceptionne en permanence les données du simulateur. Chaque plateforme présente dans le scénario apportera la création d'un agent *plateforme*, tandis que la réception des données de pistes créera ou mettra à jour un agent *tactique*. Les agents *services* seront créés à l'initialisation des agents *plateformes*.

Le *Message-Oriented Middleware* (MOM) RabbitMQ implémente le protocole AMQP, nous l'utilisons pour modéliser les échanges entre les agents avec les concepts de *producteurs* et de *consommateurs* présentés dans l'article [17],

où les choix technologiques sont détaillés.
Dans le but de vérifier le bon fonctionnement des échanges entre les agents, nous affichons la mémoire de l'agent sous forme de tableau, comme nous pouvons le voir dans un exemple en figure 5. De la même façon, nous pouvons observer l'évolution de la liste des services et de l'utilisation des ressources en cours de simulation.

AGENT_NAME	AGENT_TYPE	AGENT_ID	POSITION	PLATFORM_RESOURCES
AgentPlatform_0	Platform Agent	3	[0.884; 0.313]	[Resource [resouceType=R1,...
AgentPlatform_1	Platform Agent	9	[0.131; 0.245]	[Resource [resouceType=R1,...
AgentPlatform_2	Platform Agent	15	[0.006; 0.412]	[Resource [resouceType=R1,...
AgentPlatform_3	Platform Agent	21	[0.257; 0.775]	[Resource [resouceType=R1,...

FIGURE 5 – Mémoire de l'agent plateforme

5.2 Méthodes d'évaluation et métriques considérées

Pour évaluer notre approche, nous devrons justifier d'une réponse à nos besoins mentionnées dans la sous-section 3.2. D'une part, nous voulons évaluer la faisabilité de nos concepts au regard des contraintes en bande passante entre les plateformes. L'utilisation de RabbitMQ [6] pour les échanges entre nos agents nous permet d'utiliser les outils Grafana et Prometheus pour la gestion du réseau, l'article [17] présente l'utilisation de ces outils dans le cadre de nos travaux. Les métriques de débits et de latences seront observées au cours de la simulation sur des graphes temporels. Le but étant de constater le besoin en bande passante de notre système en fonction de plusieurs paramètres : le nombre de plateformes, le nombre de senseurs actifs et le nombre de menaces dans l'environnement. De plus, l'évolution des paramètres devra être faite en dynamique, afin de constater la résilience de notre système. C'est une étape importante pour la validation de l'architecture multi-agent.
D'autre part, nous voulons améliorer la qualité de l'allocation de tâches en optimisant nos algorithmes. Les performances de l'allocation seront impactées par l'augmentation du nombre de services disponibles dans la base de données. Un délai trop important dans la proposition des plans senseurs pourrait impacter significativement les plans à caractères prioritaires et donc urgents. Il faut donc observer l'impact du nombre de services sur le système.
En dernier lieu, nous pourrons également observer la charge de calcul du système multi-agent sur chaque plateforme, en fonction du nombre de services et du nombres d'agents *tactiques* présents.

6 Conclusion et perspectives

Dans cette article, nous avons présenté des concepts d'agents pour la collaboration multi-plateforme dans un contexte de défense navale. Dans un premier temps, nous avons expliqué le cadre complexe du combat collaboratif naval. Puis nous avons détaillé le fonctionnement de trois agents : l'agent *plateforme*, l'agent *service* et l'agent *tactique*. Ensemble, ils forment un système multi-agent qui sera intégré dans les architectures des systèmes de senseurs futurs. Enfin, nous avons proposé des méthodes pour évaluer nos concepts vis-à-vis de nos besoins.
Pour la suite des travaux, nous évaluerons les performances de l'approche multi-agent proposée avec l'aide de l'outil Grafana et des métriques évoquées dans cet article. Ensuite, nous devrons proposer plusieurs scénarios opérationnels simulés afin de mettre en pratique des tests de performances du système pour s'assurer qu'il réponde bien aux besoins opérationnels. Enfin, une étude architecturale globale intégrant le système multi-agent sera menée avec notamment des réflexions sur le choix des interfaces, sur les échanges entre les composants, etc.
Remerciements. Ces travaux ont été effectués dans le cadre d'une thèse CIFRE sur l'architecture des systèmes navals chez Thales DMS en collaboration avec IMT Atlantique et le GIS Cormorant.

Références

[1] Jacob Beal, Kyle Usbeck, Joseph Loyall, Mason Rowe, and James Metzler. Adaptive opportunistic airborne sensor sharing. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 13(1) :1–29, 2018.

[2] Ellie Beauprez, Luc Bigand, Anne-Cécile Caron, Maxime Morge, and Jean-Christophe Routier. Réaffectation de tâches de la théorie à la pratique : état de l'art et retour d'expérience. In Jean-Paul Jamont, editor, *29ème francophones sur les systèmes multi-agents (JFSMA)*, JFSMA 2021. Collectifs cyber-physiques, pages 51–60, Bordeaux, France, June 2021. Cépaduès.

[3] Han-Lim Choi, Luc Brunet, and Jonathan P How. Consensus-based decentralized auctions for robust task allocation. *IEEE transactions on robotics*, 25(4) :912–926, 2009.

[4] Alaa Daoud, Flavien Balbo, Paolo Gianessi, and Gauthier Picard. Un modèle agent générique pour la comparaison d'approches d'allocation de ressources dans le domaine du transport à la demande. In *JFSMA 2021 : 29ème Journées Francophones sur les Systèmes Multi-Agents*, pages 127–136, Bordeaux, France, June 2021. Cépaduès.

[5] Ali Dorri, Salil S Kanhere, and Raja Jurdak. Multi-agent systems : A survey. *IEEE Access*, 6 :28573–28593, 2018.

[6] David Dossot. *RabbitMQ essentials*. Packt Publishing Ltd, 2014.

[7] Ludovic Grivault. *Architecture multi-agent pour la conception et l'ordonnancement de systèmes multi-senseur embarqués sur plateformes aéroportées*. PhD thesis, Sorbonne université, December 2018.

[8] Ludovic Grivault, Amal El Fallah-Seghrouchni, and Raphaël Girard-Claudon. Agent-based architecture for multi-sensors system deployed on airborne platform. In *2016 IEEE International Conference on Agents (ICA)*, pages 86–89. IEEE, 2016.

[9] Ludovic Grivault, Amal El Fallah-Seghrouchni, and Raphaël Girard-Claudon. Next generation of airborne

platforms from architecture design to sensors scheduling. In *2017 IEEE International Conference on Agents (ICA)*, pages 60–65. IEEE, 2017.

[10] Pieter Hintjens. *ZeroMQ : messaging for many applications*. " O'Reilly Media, Inc.", 2013.

[11] Michael N Huhns. *Distributed Artificial Intelligence : Volume I*, volume 1. Elsevier, 2012.

[12] Luke Johnson, Sameera Ponda, Han-Lim Choi, and Jonathan How. Asynchronous decentralized task allocation for dynamic environments. In *Infotech@ Aerospace 2011*, page 1441. 2011.

[13] Vicente Julian and Vicente Botti. Multi-agent systems. volume 9, page 1402. MDPI, 2019.

[14] Stéphane Kemkemian and Myriam Nouvel-Fiani. Toward common radar & ew multifunction active arrays. In *2010 IEEE International Symposium on Phased Array Systems and Technology*, pages 777–784. IEEE, 2010.

[15] Antoine Milot, Estelle Chauveau, Simon Lacroix, and Charles Lesire Cabaniols. Allocation par enchères et planification hiérarchique pour un système multi-robot, application au cas de la chasse aux mines. In *JFSMA 2022 : 30èmes Journées Francophones sur les Systèmes Multi-Agents*, 2022.

[16] Sameera Ponda, Josh Redding, Han-Lim Choi, Jonathan P How, Matt Vavrina, and John Vian. Decentralized planning for complex missions with dynamic communication constraints. In *Proceedings of the 2010 American Control Conference*, pages 3998–4003. IEEE, 2010.

[17] Paul Quentel, Yvon Kermarrec, Ludovic Grivault, Pierre Le Berre, and Laurent Savy. A rabbitmq-based framework to deal with naval sensor systems design complexity. Publication acceptée à *Computing Conference 2023*, Juin, Londres, Royaume-Uni, Inpress.

[18] Duncan Russell and Jie Xu. Service oriented architectures in the provision of military capability. In *UK e-Science All Hands Meeting*. Citeseer, 2007.

[19] Alexandre Schmitt, Valérie Renault, Florent Carlier, and Pascal Leroux. De l'iot à l'iot-a : une approche pour des communications dynamiques. In *27emes Journées Francophones sur les Systèmes Multi-Agents (JFSMA)*, 2019.

[20] A El Fallah Seghrouchni and L Grivault. Multi-agent paradigm to design the next generation of airborne platforms. *Aerospace Lab*, pages 1–8, 2020.

[21] Hugo Seuté, Laurent Ratton, and Antoine Fagette. Passive sensor planning for tdoa/fdoa geolocation under communication constraints. In *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2020.

[22] Vaishnavi Singhal and Deepak Dahiya. Distributed task allocation in dynamic multi-agent system. In *International Conference on Computing, Communication & Automation*, pages 643–648, 2015.

[23] George Marios Skaltsis, Hyo-Sang Shin, and Antonios Tsourdos. A survey of task allocation techniques in mas. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 488–497, 2021.

[24] Itshak Tkach and Yael Edan. *Distributed heterogeneous multi sensor task allocation systems*. Springer, 2020.

[25] Joanna Turner, Qinggang Meng, Gerald Schaefer, and Andrea Soltoggio. Distributed strategy adaptation with a prediction function in multi-agent task allocation. 2018.

[26] Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & sons, 2009.

[27] Bing Xie, Jing Chen, and Lincheng Shen. Cooperation algorithms in multi-agent systems for dynamic task allocation : A brief overview. In *2018 37th Chinese Control Conference (CCC)*, pages 6776–6781, 2018.