# Influence des modèles de mobilité sur un système collaboratif pour flottes autonomes hétérogènes

Vincent Autefage, Serge Chaumette, Damien Magoni

# *Influence des modèles de mobilité sur un système collaboratif pour flottes autonomes hétérogènes*[†]

Vincent Autefage[1], Serge Chaumette[1] et Damien Magoni[1]

[1] *Univ. Bordeaux, LaBRI, UMR 5800, F-33400, Talence, France*

Cet article s'intéresse à l'impact de trois modèles de mobilité sur un nouveau système collaboratif pour les flottes hétérogènes d'objets mobiles autonomes communicants. Notre système est entièrement distribué et repose uniquement sur des messages asynchrones. Nous donnons une description de notre modèle, basé sur du ré-étiquetage de graphes dynamiques, ainsi que des résultats de simulations reposant sur un scénario de nettoyage de parc à l'aide de drones aériens et de robots terrestres spécialisés. Nous observons que les différents modèles de mobilité influent peu sur les performances de notre système.

**Keywords:** flotte autonome, système collaboratif, modèle de mobilité, ré-étiquetage de graphes dynamiques.

## 1 Introduction

Un système autonome [1] (*Umanned System*, UMS) est une entité mobile possédant un degré variable d'autonomie dans ses décisions et actions possibles. Depuis quelques années, un certain nombre d'applications impliquant notamment des drones (*Unmanned Aerial Vehicle*, UAV) ont été développées aussi bien dans le domaine civil que militaire. Certaines de ces applications reposent sur l'utilisation de plusieurs engins dans le but d'améliorer l'efficacité de la mission en termes de temps d'exécution ou de consommation énergétique [2]. En effet, les engins utilisés étant souvent de taille réduite, l'emport d'une charge utile importante (i.e. capteurs et actionneurs) est souvent impossible. Nous nous intéressons ici à des flottes hétérogènes dynamiques, i.e. des flottes utilisant des engins mobiles de types différents (terrien et aérien) et dont la topologie du réseau sous-jacent change au cours du temps. De plus, chacun des appareils possède des capacités distinctes. Il est donc nécessaire d'implémenter un mécanisme de collaboration permettant aux engins de la flotte de faire appel aux capacités des autres appareils. Dans cet article, nous proposons un nouveau mécanisme de collaboration appelé AMiRALE (*Asynchronous Missions Relay for Autonomous and Lively Entities*). Notre système est entièrement distribué et repose uniquement sur des messages asynchrones. Cette configuration, comparée à une approche centralisée, fournit un certain nombre d'avantages tels que l'exploitation naturelle du parallélisme, la résilience et le passage à l'échelle [3]. La mobilité ayant un impact important sur la connectivité du réseau [4], l'efficacité d'un systèmes distribué est généralement dépendante de celle-ci [5]. Nous nous intéressons donc également à l'impact de cette mobilité sur notre système collaboratif en termes de vitesse d'exécution et de consommation énergétique. Nous proposons une description du modèle théorique de notre système collaboratif, basé sur un mécanisme de ré-étiquetage de graphes dynamiques, ainsi que des résultats de simulations sur un scénario de nettoyage de parc à l'aide de drones aériens et de robots terrestres spécialisés.

## 2 Présentation du système collaboratif

AMiRALE est un mécanisme basé sur la notion de mission qui permet à une flotte d'engins (i.e. drones, robots terrestres, etc.) extrêmement dynamiques de réaliser un ensemble de tâches de manière collaborative. Un certain nombre d'évènements pré-enregistrés (e.g. seuil de température dépassé, son anormal, mouvement suspect) peuvent entraîner une action à effectuer par la flotte (e.g. envoi

[†]Ce travail est co-financé par la *Direction Générale de l'Armement* et la *Région Aquitaine*.

d'informations vers l'extérieur, activation d'une routine). Nous appelons ici *mission* l'ensemble des informations relatives à un (ou des) événement(s) qui vont devoir être échangées afin d'appliquer l'action à effectuer.

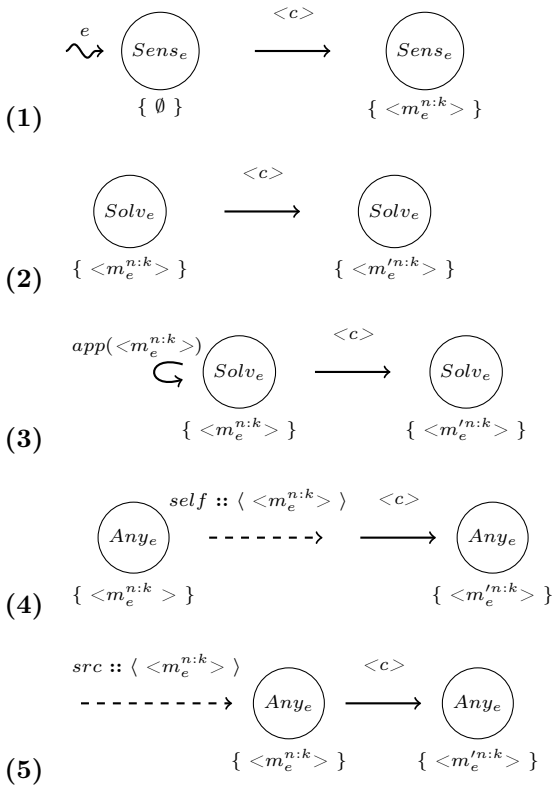
Soit un événement de type $e$, $Sens_e$ (*capteur*) est une entité capable de capturer l'évènement et de générer la mission $m_e^{n:k}$ où $n$ est l'identifiant du créateur de la mission, $k$ est un numéro de séquence local au nœud $n$, incrémenté à chaque nouvelle mission créée. $Solv_e$ (*solveur*) est une entité capable de résoudre une mission de type $e$. Un troisième type de nœuds appelé $Forw_e$ (*relais*) ne fait que transmettre les informations relatives à une mission aux autres entités de la flotte. Pour un type $e$, si un nœud n'est ni $Sens_e$, ni $Solv_e$, il est par défaut $Forw_e$. La flotte pouvant gérer plusieurs types de missions, un même nœud peut avoir des rôles distincts pour des types différents de missions. En effet, pour deux types d'événements différents $e$ et $f$, un nœud peut être $Sens_e$ et $Solv_f$. En outre, un nœud $Sens_e$ n'est pas nécessairement $Solv_e$ et réciproquement. Cette particularité est due aux limitations physiques des appareils utilisés (e.g. poids, taille, capacité énergétique).

Chaque nœud possède un identifiant unique, une mémoire et sa propre horloge (nous considérons ici que les horloges sont synchronisées). Une mission est représentée par un 7-uplet $\{e, k, n, t, s, n', t'\}$ où $e$ est le type de la mission, $k$ le numéro de séquence de la mission et $n$ l'identifiant du nœud créateur. $\{e, k, n\}$ représente donc l'identifiant unique de la mission. $t$ est la date de création de la mission, $n'$ est l'identifiant du nœud qui a mis à jour la mission pour la dernière fois et $t'$ est la date de dernière modification. $s$ représente l'état courant de la mission et peut prendre 5 valeurs distinctes : *start* (mission en attente de traitement), *will* (mission récupérée par un solveur mais pas encore en traitement), *do* (mission en traitement), *abort* (mission abandonnée), *end* (mission terminée). Ces états sont strictement ordonnés, i.e. $start < will < do < abort < end$. Une mission $m_e^{n:k}$ ne peut être créée (état *start*) que par un $Sens_e$. L'état de cette mission ne peut être modifié que par un $Solv_e$. Les $Forw_e$ sont des nœuds en lecture seule. Les données contenues dans les missions ne sont pas représentées dans le modèle.

À intervalles de temps réguliers, les nœuds diffusent à leurs voisins des parties de leur mémoire. Cela permet aux nœuds de mettre à jour et d'unifier les versions des différentes missions que la flotte doit effectuer. Cette diffusion est faite au travers de *broadcasts*. Lorsqu'un nœud reçoit un nouvelle version d'une mission dont il est connaît l'existence, il met à jour sa version locale et diffuse la nouvelle version. Lorsqu'un $Solv_e$ a connaissance d'une mission $m_e^{n:k}$ dont l'état est *start*, il modifie cet état en *will* et se prépare à résoudre la mission (e.g. en se déplaçant vers une position précise si nécessaire). Quand il commence à résoudre la mission, il fait évoluer son état en *do*. Si le solveur détecte que la mission a déjà été résolue (e.g. un objet devant être ramassé à disparu), il fixe l'état de la mission à *abort*. Enfin, quand le solveur estime que la mission est terminée, il fixe l'état relatif à *end*. Un solveur ne peut être engagé en *will* ou en *do* que dans une seule mission au même moment. De plus, un solveur ne peut conserver l'état *will* (resp. *do*) que pour un certain laps de temps $\Psi_{will}^e$ (resp. $\Psi_{do}^e$). Après ce seuil, le solveur abandonne la mission au profit d'un autre solveur s'il est informé que ce dernier a pris la main sur la mission concernée. Enfin, si un solveur est informé qu'une mission dans laquelle il est engagé, a évolué vers un état supérieur, il abandonne la mission et met à jour sa mémoire locale.

Certaines décisions internes du mécanisme collaboratif sont dépendantes de l'application. Par exemple, créer ou non une nouvelle mission suite à un événement déjà considéré ou encore ne pas diffuser une mission car celle-ci a déjà été diffusée suffisamment, sont des décisions qui dépendent entièrement de l'application. Par conséquent, AMiRALE inclut des fonctions utilisateurs appelées *filtres* permettant de personnaliser et de configurer finement les mécanismes de décisions internes du modèle en tenant compte des spécificités de l'application.

AMiRALE a été formalisé en utilisant une version étendue d'ADAGRS [2] [6] qui est un formalisme de ré-étiquetage de graphes dynamiques. Nous présentons en figure 1 les différents types de règles du modèle. Ces règles décrivent les interactions entre un nœud et une mission de type $e$. Pour chacune des règles, les 2 cercles représentent le rôle du nœud (i.e. capteur, solveur et relais) avant et après avoir appliqué la règle. L'ensemble en dessous de chaque cercle représente la version courante de la mission $m_e^{n:k}$. La règle est appliquée uniquement si la condition $c$ est satisfaite. Par simplification, nous avons ajouté un rôle $Any_e$ qui peut s'appliquer à n'importe lequel des rôles précédemment décrits. La description complète du modèle est disponible dans un rapport technique [7].




1. Cette règle est la seule permettant la création d'une mission. Un capteur de type $e$ nommé $n$ créé la mission $m_e^{n:k}$ si la condition $c$ est vérifiée. Cette condition peut, par exemple, être utilisée pour vérifier si une mission similaire est déjà en cours de résolution.
2. Cette règle permet à un solveur de modifier la version courante d'une mission. Après l'application de cette règle, la mission $m_e^{n:k}$ est mise à jour à la nouvelle version $m_e'^{n:k}$.
3. Cette règle permet à un solveur de réagir à un événement applicatif relatif à la mission $m_e^{n:k}$ (e.g. mission résolue, mission abandonnée).
4. Cette règle permet à un nœud de diffuser sa version locale d'une mission. Cette diffusion est décrite par $self :: m_e^{n:k}$ où $self$ représente l'identifiant du nœud émetteur.
5. Cette règle permet à un nœud de mettre à jour une mission suite à la réception d'une version plus avancée. Le message reçu est décrit par $src :: m_e^{n:k}$ où $src$ est l'identifiant de l'émetteur du message.

**Figure 1:** Types de règles du modèle AMiRALE.

## 3 Expérimentation

Nous présentons dans cette section des résultats de simulations au travers d'un scénario de nettoyage de parc [8]. Ce scénario permet de collecter un ensemble de détritus dans un parc à l'aide de drones aériens et de robots terrestres spécialisés, i.e. chaque robot est capable de collecter un seul type de détritus (e.g. papier, verre, plastique, compost). Les drones aériens sont utilisés ici pour repérer les déchets au sol et transmettre leur position aux robots. Les drones sont donc des $Sens_i$ pour tout type $i$ de détritus. Chaque robot est $Solv_i$ pour $i$ un type particulier de déchets. La donnée de la mission est ici la position du déchet relatif.

Notre objectif est ici de comparer les performances d'AMiRALE en termes de vitesse d'exécution du scénario (i.e. parc entièrement nettoyé) et de dépense énergétique (i.e. distance moyenne parcourue par les robots, l'utilisation des moteurs étant la principale source de consommation [9]). Ces résultats sont obtenus en fonction du nombre de déchets dans le parc, ceux-ci étant répartis de manière uniforme. Un robot ne se déplace que s'il est actif pour une mission donnée (i.e. *will* ou *do*). Les drones se déplacent en permanence selon trois modèles de mobilité différents : *Random Waypoint* [10] où les drones se déplacent de manière aléatoire en sélectionnant un point de destination dans le plan ; *Smooth Turn* [11] où les nœuds se déplacent de manière circulaire autour d'un centre virtuel pris pseudo-aléatoirement pendant une durée aléatoire ; *Semi-Random Circular Movement* [12] où les drones se déplacent de manière circulaire autour du centre du plan en changeant aléatoirement la taille du rayon à chaque tour. Le *Smooth Turn* et le *Semi-Random Circular Movement* sont deux modèles de mobilité qui prennent en compte les particularités aérodynamiques des engins volants. Nous mesurons enfin le nombre moyen de voisins des nœuds sur l'ensemble de la simulation ainsi que la proportion moyenne de temps pour laquelle les nœuds sont complètement isolés. Ces résultats sont fournis pour chacun des modèles de mobilité présentés ci-dessus. Nous avons effectué nos expérimentations dans le simulateur de graphes dynamiques JBotSim [13] qui permet de concevoir finement des scénarios utilisant des objets mobiles hétérogènes et communicants. Nos résultats sont décrits dans la figure 2.

| Paramètre | Valeur |
|---|---|
| Taille du parc | 1 x 1 km |
| Taille des robots / drones | 1 x 1 m |
| Vitesse des robots / drones | 5 m / sec |
| Portée de communication | 30 m |
| Portée de détection des robots | 30 m |
| Portée de détection des drones | 90 m |
| Nombre de types de déchets | 6 |
| Nombre de robots par type | 6 |
| Nombre total de robots | 36 |
| Nombre de drones | 6 |
| Nombre de déchets par type | [20; 120] |
| Nombre total de déchets | [120; 720] |
| Nombre d'exécutions par point sur chaque courbe | 500 |

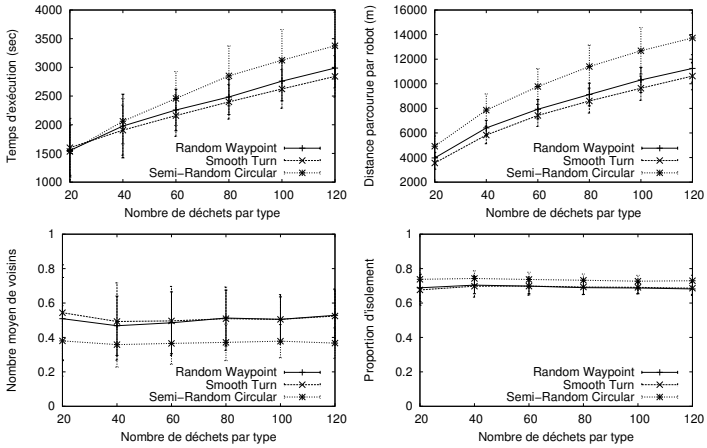



**Figure 2:** Paramètres et résultats de simulation.

## 4 Discussion et conclusion

Nos résultats de simulations montrent une faible différence de performances (tant sur le temps d'exécution que sur la dépense énergétique) entre les modèles de mobilité *Smooth Turn* et *Randow Waypoint*. Le *Smooth Turn* étant une version dérivée du *Random Direction*, ces résultats sont cohérents avec les précédentes études qui portent sur les différences de performances réseaux (i.e. débit, délai, etc.) entre ces deux modèles [5]. La très légère avance du *Smooth Turn* est due au fait que ce modèle préserve une distribution uniforme des drones tout au long du scénario, ce qui implique une meilleure connectivité entre les nœuds et donc une meilleure collaboration grâce à des échanges de messages plus réguliers [11]. Les performances du modèle *Semi-Random Circular* sont en léger retrait. La principale raison est que ce modèle impose aux drones d'effectuer de grandes distances (i.e. cercle complet) avant de revenir au point d'origine. Les robots terrestres étant statiques tant qu'ils n'ont pas de mission à effectuer, les communications entre les drones et les robots sont donc moins fréquentes. Nos courbes présentant le nombre moyen de voisins par nœud ainsi que la proportion de temps pour laquelle les nœuds sont isolés corroborent ces résultats. Nous pouvons enfin remarquer qu'AMiRALE supporte les flottes fortement déconnectées, les nœuds étant isolés entre 60 et 80 % du temps total du scénario.

## Références

[1] NIST. Autonomy levels for unmanned systems. Technical report, Otcober 2008.

[2] S. Chaumette, R. Laplace, C. Mazel, R. Mirault, A. Dunand, Y. Lecoutre, and J.-N. Perbet. Carus, an operational retasking application for a swarm of autonomous uavs. In *MILCOM*, pages 2003–2010, November 2011.

[3] Y.U. Cao, A.S. Fukunaga, A.B. Kahng, and F. Meng. Cooperative mobile robotics : antecedents and directions. In *IEEE/RSJ Intelligent Robots and Systems*, volume 1, pages 226–234, August 1995.

[4] G. Ravikiran and S. Singh. Influence of mobility models on the performance of routing protocols in ad-hoc wireless networks. In *IEEE 59th VTC 2004*, volume 4, pages 2185–2189, May 2004.

[5] Tracy Camp, Jeff Boleng, and Vanessa Davies. A survey of mobility models for ad hoc network research. *Wireless communications and mobile computing*, 2(5) :483–502, 2002.

[6] Rémi Laplace. *Applications et services DTN pour flotte collaborative de drones*. PhD thesis, Université Sciences et Technologies - Bordeaux I, december 2012.

[7] Vincent Autefage. Amirale formal model. Technical report, LaBRI - University of Bordeaux, February 2015. https://hal.archives-ouvertes.fr/hal-01114961/.

[8] Vincent Autefage, Arnaud Casteler, Serge Chaumette, Nicolas Daguisé, Arnaud Dutartre, and Tristan Mehamli. Parcs-s2 : Park cleaning swarm supervision system - position paper. In *9th AIRTEC International Congress*, October 2014.

[9] Yongguo Mei, Yung-Hsiang Lu, Y.C. Hu, and C.S.G. Lee. A case study of mobile robot's energy consumption and conservation techniques. In *ICAR 12th*, pages 492–497, July 2005.

[10] Christian Bettstetter, Hannes Hartenstein, and Xavier Pérez-Costa. Stochastic properties of the random waypoint mobility model. *Wireless Networks*, 10(5) :555–567, September 2004.

[11] Yan Wan, K. Namuduri, Yi Zhou, and Shengli Fu. A smooth-turn mobility model for airborne networks. *IEEE Vehicular Technology*, 62(7) :3359–3370, September 2013.

[12] Wei Wang, Xiaohong Guan, Beizhan Wang, and Yaping Wang. A novel mobility model based on semi-random circular movement in mobile ad hoc networks. *Inf. Sci.*, 180(3) :399–413, February 2010.

[13] Arnaud Casteigts. The jbotsim library. *CoRR*, abs/1001.1435, 2010.