Comparison of Time Synchronization Techniques in a Distributed Collaborative Swarm System

Vincent Autefage, Serge Chaumette, Damien Magoni

Comparison of Time Synchronization Techniques in a Distributed Collaborative Swarm System

Vincent Autefage
Univ. Bordeaux, LaBRI
Talence, France
autefage@labri.fr

Serge Chaumette
Univ. Bordeaux, LaBRI
Talence, France
serge.chaumette@labri.fr

Damien Magoni
Univ. Bordeaux, LaBRI
Talence, France
magoni@labri.fr

***Abstract*—Time drift in distributed systems is an important issue especially for mobile ad hoc networks. Indeed, it can break the synchronization between the internal clocks of the nodes. Consequently, protocols over such networks should consider this problem in the case where they need to use time information. In this paper, we define a new collaborative system for autonomous mobile robots called AMiRALE. We present three versions of this system each with different time constraints. We provide its theoretical description as well as simulation results according to a park cleaning scenario. Results show that time synchronization has a limited impact on the efficiency of our collaborative system.**

Keywords—Time synchronization, autonomous network, mobile network, distributed system, collaboration.

I. Introduction

According to several studies [1] [2] [3], there were more than 10 billions connected and communicating objects across the world in 2013. This number is expected to grow up to reach 30 to 80 billions in 2020. Among those connected objects, there were more than 4 billions civilian robots in 2012 which represented a market of more than 10 billion euros [4]. This market should exceed 100 billion euros before 2020 [1]. One of the emerging robotic approaches is *collaboration* [5] [6] [7] which is the way to use several robots in order to achieve a global goal. Over the last few years, collaboration between robots has been widely studied through several topics as swarm architecture [8], distributed intelligence [9] [10], task-allocation [11] and fault tolerance [12]. In a realistic approach, there is no central control agent which manages each individual robot. Indeed, a distributed system is preferred, which means that each robot in the swarm is independent and communicates with the others in order to adapt its own behavior. This decentralized approach, compared to a centralized scheme, brings several advantages such as natural exploitation of parallelism, reliability, and scalability [13]. Despite these benefits, distributed systems face an important challenge which is time synchronization [14]. Indeed, even though each entity of the swarm has its own clock, the distributed aspect implies that those clocks can drift and thus not be synchronized [15].

In this paper we propose a new distributed system which enables collaboration between several mobile robots. We have developed several versions of our proposal which take care of the time drift problem between the clocks of the robots. We aim to investigate the performance differences of those versions in order to highlight the pros and cons of choosing one of them. The remainder of this paper is organized as follows. We first provide a short explanation about the time drift problem and its common existing solutions in distributed systems. In Section III, we describe our new collaboration model and its several time synchronization adaptations. In Section IV, we present and discuss several simulation results. We finally conclude and present future work in Section V.

II. Time Synchronization

In distributed systems, time is a key problem. Indeed, each node of such a network should have its own internal clock and the time can drift differently from one node to another for several reasons like temperature variations or quartz properties [15]. This problem has been well known for many years and has led to the creation of the NTP protocol [16]. In MANETs [17] (*Mobile Ad-Hoc Networks*), nodes are moving, involving various disconnections in the network over time. In other words, in a such network, it is not possible to ensure that a communication is always possible between two nodes. Therefore, making one (or several) of these nodes a central time synchronization agent (as for NTP) is not suitable.

Most of the previous studies on collaborative swarms of mobile nodes do not address the problem of time synchronization between nodes. Indeed, even if their communication protocols involve time (through timeouts and periods for instance), they usually consider that time is synchronized by a global entity [18] [19] [20] (e.g., GPS or global timing agent). Even in this case, it is possible for a node to loose for an arbitrary time the connection with the global timing agent. For instance, it has been proven that vegetation or landform could lead to communication failures with a GPS system and thus it could introduce temporal aberrations between nodes [21]. Moreover, a large part of mobile robots and connected objects are not equipped with such synchronization mechanisms because of energy, bandwidth, hardware and unstable connections constraints [22].

Three main approaches exist in order to overcome the time drift problem in distributed systems [23]. The *relative ordering* orders events without time reference. In this scheme, communications between robots don't contain any direct time reference (i.e., date or clock value) [24]. In the *relative timing*, nodes take into account the disparity of drift time from the others. In this scheme, communications between robots usually contain time deviation values (i.e., delta) instead of absolute dates [25]. Finally the *global timing* ensures that all clocks are synchronized by using a global timing agent (e.g., NTP) [16].

In this paper, we present three versions (i.e., one per synchronization mechanism) of a distributed collaborative system for a swarm of heterogeneous robots and we study their efficiency differences through a park cleaning scenario.

III. AMiRALE: a Collaborative System

In this section we present AMiRALE (*Asynchronous Missions Relay for Autonomous and Lively Entities*), a collaborative and distributed system for heterogeneous swarms of mobile nodes (e.g., robots, UAVs). AMiRALE enables a swarm of specialized mobile nodes to perform common tasks collaboratively.

A certain set of events (e.g., temperature threshold exceeded, suspicious movement) requires a reaction of the swarm (e.g., sending a signal to a control center, triggering an alarm). A *mission* is the set of information that describes such an action to be performed by the swarm. A *sensor* is a node that is able to detect an event (through its sensors) and create the corresponding mission. A *solver* is a node that is able to apply the specific action required by the mission. Finally, a *forwarder* is just a node which forwards a mission through the swarm. The swarm is able to manage several types of missions, one node can thus have several roles. Moreover, a sensor node for a specific mission type is not always a solver for this same type (because of energy consumption, space and weight limitations, etc.). Therefore, the system requires a dissemination mechanism and a task-allocation strategy. Each node has its own clock, its own memory (missions' database) and an unique identifier. At a regular period, nodes share some parts of their database to their neighbors and update the status of the received missions. There is no synchronization between the nodes, all operations are made locally and communications are performed through broadcasts. One of the fields of the mission description is its internal *state* which evolves while the mission is propagated through the network. This state can take five different values which are strictly ordered. This order is used later in this paper to describe the evolution of the system:

1) *start*: the mission has been created but no node is currently trying to solve it;
2) *will*: the mission has been caught by a solver, and it is preparing to solve it (e.g., moving to the location where the mission takes place);
3) *do*: the mission is currently being solved;
4) *abort*: the mission has been dropped because it has apparently already been solved (e.g., a target object that the solver is supposed to remove has disappeared);
5) *end*: the mission is solved.

The *start* state is only set at mission creation time by the node that generates it. The other states can only be set by solvers. A forwarder is a *read-only node*; i.e., it cannot modify the state field. As explained before, states are strictly ordered: $start < will < do < abort < end$. Each mission description contains a state, a creator identifier and creation date, the date of last modification and the identifier of the last node that updated it.

When a node receives a newer version of one of the missions it is aware of, it updates this mission in its local memory and broadcasts the new version. When a solver gets a mission in *start* state that it is able to solve, it turns the state field to *will* and prepares itself to solve it (e.g., by moving to a specific location if required). When it begins solving the mission, it turns its state field to *do*. If the solver detects that the mission has apparently already been solved (e.g., a target object that the solver is supposed to remove has disappeared), it turns the state field to *abort*. Finally, when it has succeeded in solving the mission, it turns the state field to *end*. A solver can be in *will* or *do* mode for only one mission at a time. Furthermore, a solver can stay in the *will* (resp. *do*) state only for a limited time. If a time threshold is exceeded, the solver leaves the mission if and only if it is informed by another solver that this last one is now taking care of the same mission. Additionally, if a solver is informed that the mission it is dealing with (*will* or *do* state) has evolved to a greater state, it leaves the mission and updates the corresponding description. For a specific mission type, if a node is neither a sensor, nor a solver, it is considered as a forwarder.

Since several decisions and improvements are application-dependent (e.g., creating a new mission for an event which is already considered, not sharing a mission because it has been already shared enough for a certain amount of time), AMiRALE includes several user defined functions called *filters* in order to improve the behavior of the distributed system.

As explained before, each mission contains two time information: the date of creation and the date of last update. In the case where all nodes of the swarm have their clocks synchronized (i.e., *global timing*), missions can be shared without modifications. The first version of AMiRALE is based on this assertion. The two other versions of AMiRALE are working on desynchronized swarms. In other words, each node of the swarm has its own clock and no time synchronization is operated on these clocks. The *relative timing* version uses delta times rather than raw dates in messages. This is the only difference, the internal behavior is the same as the global timing version. The last version operates in *relative ordering*. In this version, the shared missions do not contain any time reference. Thresholds overruns are considered from the moment when the node has received the mission. Here, the global behavior is quite different, since time constraints are less accurate compared to the two other versions. A node can switch from one version of AMiRALE to another if it considers that the requirements are no longer met (e.g., loss of the GPS signals). This capability enables to avoid the issue of communication failures between nodes and timing agents, introduced in Section II.

These three versions of AMiRALE have been formalized through an extended dynamic graph relabeling formalism called ADAGRS [20] [26] (*Asynchronous Dynamicity Aware Graph Relabeling System*) which is only based on one way broadcasts and local computations. The limitations imposed by this model are the key to ensure a totally decentralized and distributed approach. The complete model of the three versions is available in a technical report [27]. Due to space limitations, we cannot write the detailed model in this paper.

IV. Experimentation

A. Simulation Scenario

In order to study the differences between the time synchronization versions of AMiRALE, we evaluate each one through a park cleaning scenario called ParCS [28]. This scenario offers a solution to achieve a selective collection of waste in a park by using an autonomous swarm composed of UGVs (*Unmanned Ground Vehicles*). These vehicles are self-organized so as to manage the collection of waste and to clean the park. Each UGV is specialized, which means that it can only clean one kind of garbage (e.g., paper, glass, compost, plastic). This specialization makes the swarm heterogeneous. However, each UGV is able to sense any kind of garbage. In this scenario, each mobile robot is moving by following a *random way-point mobility model* [29]. All waste pieces are thrown uniformly in the park.

B. Simulation Setup

AMiRALE is used here to enable the collaboration between the mobile robots. Several optimizations in the model are performed (through user filters introduced in Section III) in order to improve the global efficiency of the swarm. When a sensor detects a garbage, a mission is generated only if no other mission concerning the same garbage already exists in the missions' database of this sensor (filter f^e_{ignore}). A solver takes the mission relative to the closest garbage that it can clean (filter f^e_{pass}). A node does not broadcast a mission if this last one is is marked as solved and no other contrary information has been received for more than 1000 seconds. Also, a mission is not shared if a similar version has been received for less than 100 seconds without contradiction (filter f^e_{blind}). If two solvers are targeting the same mission and they are close enough to exchange their respective versions, the farthest node from the garbage leaves the mission for the benefit of the other (filter f^e_{select}). A solver is allowed to keep a mission in *will* or *do* state for 1000 seconds. Each robot broadcasts its known missions every 5 seconds.

We have run our simulations in a dynamic graph simulator called JBotSim [30] [31]. This software is actually a Java library which enables the design of low and high level scenarios and behaviors of communicating heterogeneous mobile nodes. Our simulation parameters are described in Table I.

Table I. Experimentation parameters

Parameter	**Value**
Park size	1000 x 1000 m
Robots size	1 x 1 m
Robots speed	5 m / second
Robots sensing range	30 m
Robots communication range	30 m
Robots communication delay	1 second
Number of runs per simulation	500

For the *relative timing* and the *relative ordering* AMiRALE versions, the internal clock of each node is desynchronized by more than 1 second which means that no node has the same clock value at the same moment. We evaluate here the complete cleaning time and the average number of shared missions in function of several parameters:

- number of types: number of existing garbage types in the park (Figure 3);
- number of garbage: number of garbage per type (Figure 1);
- number of robots: number of robots per type (Figure 2);
- frequency of failures: frequency at which a random robot is deleted from the map and replaced by a new similar and memory-empty one (Figure 4).

C. Simulations Results

Figure 1 exhibits the total cleaning time in seconds and the average number of shared missions for the different AMiRALE versions (*global timing*, *relative timing* and *relative ordering*) as a function of the number of garbage per type. The number of types is set to 6 as the number of robots per type (i.e., 36 robots in total).

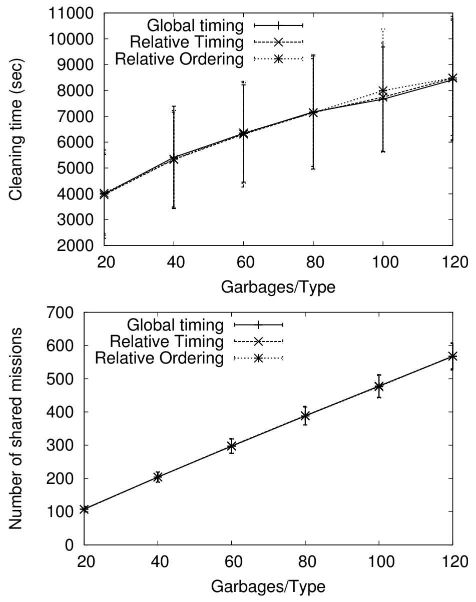

Figure 1. Cleaning time and number of shared missions for 6 types and 6 robots per type as a function of the number of garbage.

As expected, the cleaning time and the number of shared missions increase as the number of garbage is more and more important. We can notice that the three versions provide similar performance results. Also we notice that the number of shared missions increases to 600 missions for 120 wastes per type (i.e., 720 wastes in total). As explained before, a shared mission contains a type, 2 node identifiers (creation and last update), 2 time information (for the *global timing* and the *relative timing* versions), a state and a data. Here the data is the position of the target garbage (e.g., GPS coordinates). Since GPS coordinates for a point can be stored in 80 bits [32] [33] (with a precision of about 1 meter), we can simply assume that the store size for a mission will not require more than 171 bits (32 bits per time information, 8 bits per node identifier, 3 bits for the state, 8 bits for the mission type, and 80 bits for the garbage coordinates). We have also to add the sender position and the sender identifier which brings the store size to not exceed 260 bits. If we consider that a node has to send 600 missions every 5 seconds, we can assume that the requiring bandwidth is at least a bit more than 30 kbps which is slow enough to use standard protocols usually used for communications between robots like Wi-Fi [34], Bluetooth [35] and ZigBee [36].

Figure 2 and Figure 3 exhibit the total cleaning time in seconds and the average number of shared missions respectively as a function of the number of robots per type and the number of types. The number of types is set to 6 and the number of garbage per type is set to 60 (i.e., 360 garbage in total) for the Figure 2. The number of robots per type is set to 6 and the number of garbage per type is set to 60 for the Figure 3. In this last case, the number of robots increases too, since the number of robots is given per type. We can notice that the total cleaning time decreases while the number of robots increases. Also, the average number of shared missions increases in both cases. In Figure 2, when the number of robots per type increases, the number of sensors increases too. Therefore the number of created missions is more important. In Figure 3, the increase of the number of types implies an expansion of the number of waste pieces too. This is the reason why the number of shared missions is more important.

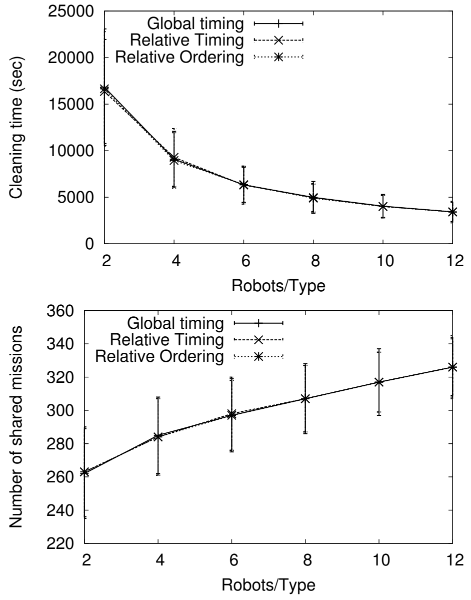

Figure 2. Cleaning time and number of shared missions for 6 types and 60 garbage per type as a function of the number of robots.

Figure 4 exhibits the total cleaning time in seconds and the average number of shared missions for the different solutions as a function of the failures frequency. The number of types is set to 6 as the number of robots per type and the number of garbage per type is set to 60 (i.e., 36 robots and 360 garbage in total). At each reset, a random robot is removed from the map and replaced by a fresh new similar and empty-memory one.

We can notice that the three versions are not affected by the reset in terms of completion time. Indeed, even if the memory of the removed robot is lost, the replication mechanism inside the AMiRALE system allows the fresh new replacing robot to recover a coherent global mission state of the scenario by communicating with the other nodes. We can thus claim here that the three versions of AMiRALE are resilient to node failures even at a high frequency (e.g., every 100 seconds). Also, we notice that the number of shared missions is more important for a high reset frequency (i.e., small value between two resets). The reason is that the replication mechanism requires to share an important part of the database in order to enable the new empty-memory node to recover the status of each active mission.

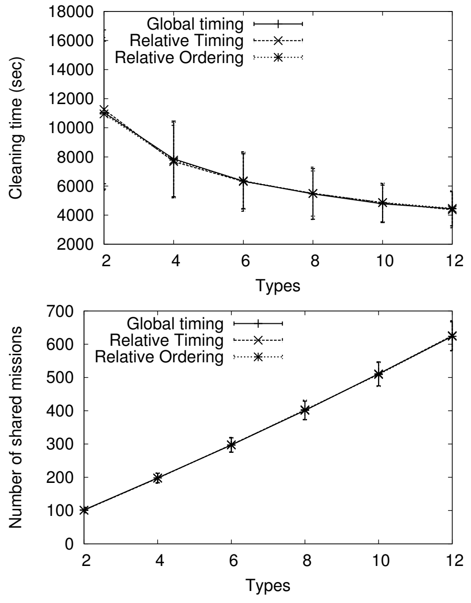

Figure 3. Cleaning time and number of shared missions for 60 garbage per type and 6 robots per type as a function of the number of types.

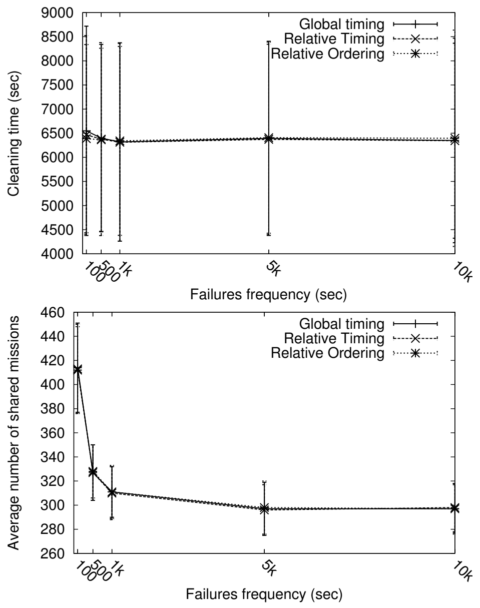

Figure 4. Cleaning time and number of shared missions for 6 types, 6 robots per type and 60 garbage per type as a function of the failure frequency.

V. Conclusion

In this paper, we have introduced the three versions of AMiRALE, a collaborative and distributed system for heterogeneous swarms of mobile nodes. Each version is dedicated to a particular solution of the time drift problem in distributed systems. We have detailed the theoretical concepts and have performed several simulations on a park cleaning scenario in order to compare the three versions of AMiRALE. We have shown that the three versions provide similar performance results in terms of total cleaning time, average number of shared missions and resilience to node failures. We have also exhibited the fact that the three versions of AMiRALE are compliant with common communication protocols for robots in terms of bandwidth requirements (i.e., Wi-Fi, Bluetooth and ZigBee). For future work, we plan to add other vehicle types such as *Unmanned Aerial Vehicles* to our park cleaning scenario in order to improve the sensing process. We will run a prototype of AMiRALE on virtual machines interconnected by a network emulator such as *NEmu* [37] [38] in order to carry out performance measurements. We also plan to experiment our scenario in a real park with real vehicles [28]. We have already started a collaboration with the Mugen Company[1] for this purpose.

Acknowledgment

This work is partly funded by the *Direction Générale de l'Armement*[2] and the *Région Aquitaine*[3].

References

[1] Cap-Digital, "Cahier de tendances - marchés et leviers," Tech. Rep., 2014.

[2] J. Bradley, J. Barbier, and D. Handler, "L'internet of everything, livre blanc," Tech. Rep., 2013, (CISCO Systems).

[3] A. Regalado, "Business adapts to a new style of computer," *MIT Technlology Review - The Internet of Things*, pp. 1–2, July/August 2014.

[4] CERNA, "Ethique de la recherche en robotique," Tech. Rep., November 2014.

[5] N. El Zoghby, V. Loscri, E. Natalizio, V. Cherfaoui *et al.*, "Robot cooperation and swarm intelligence," *Wireless Sensor and Robot Networks: From Topology Control to Communication Aspects*, 2014.

[6] Y. Mohan and S. Ponnambalam, "An extensive review of research in swarm robotics," in *Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, December 2009, pp. 140–145.

[7] Z. Shi, J. Tu, Q. Zhang, L. Liu, and J. Wei, "A survey of swarm robotics system," in *Advances in Swarm Intelligence*, 2012, pp. 564–572.

[8] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for swarm robots," in *IEEE/RSJ Intelligent Robots and Systems*, vol. 1, Jul 1993, pp. 441–447.

[9] L. E. Parker, "Distributed intelligence: Overview of the field and its application in multi-robot systems," in *AAAI Fall Symposium on Regarding the Intelligence in Distributed Intelligent Systems*, 2007.

[10] ——, "Distributed intelligence: Overview of the field and its application in multi-robot systems," *Journal of Physical Agents*, vol. 2, no. 1, pp. 5–14, 2008.

[11] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[12] L. E. Parker, "Reliability and fault tolerance in collective robot systems," *Handbook on Collective Robotics: Fundamentals and Challenges*, pp. 167–204, 2013.

[13] Y. Cao, A. Fukunaga, A. Kahng, and F. Meng, "Cooperative mobile robotics: antecedents and directions," in *IEEE/RSJ Intelligent Robots and Systems*, vol. 1, Aug 1995, pp. 226–234.

[14] A. S. Tanenbaum and M. van Steen, *Distributed Systems: Principles and Paradigms (2nd Edition)*, P. Hall, Ed., Otcober 2006.

[15] D. Mills, "Modelling and analysis of computer network clocks," *Electrical Engineering Department Report*, vol. 9252, 1992.

[16] D. Mills, U. Delaware, J. Martin, J. Burbank, and W. Kasch, *Network Time Protocol Version 4: Protocol and Algorithms Specification*, June 2010, rfc 5905.

[17] J. Mackar and S. Corson, *Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*, IETF, January 1999, rfc 2501.

[18] L. Parker, "Alliance: an architecture for fault tolerant multirobot cooperation," *IEEE Robotics and Automation*, vol. 14, no. 2, pp. 220–240, April 1998.

[19] ——, "Task-oriented multi-robot learning in behavior-based systems," in *IEEE/RSJ Intelligent Robots and Systems*, vol. 3, November 1996, pp. 1478–1487.

[20] 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: First return on experience," in *MILCOM*, November 2011, pp. 2003–2010.

[21] B. B. Irène Girard, Christopher Adrados and G. Janeau, "Application de la technologie gps au suivi du déplacement de bouquetins des alpes," no. XXIV, pp. 105–126, 2009.

[22] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey," *Ad Hoc Networks*, vol. 3, no. 3, pp. 281–323, 2005.

[23] B. Kaur and A. Kaur, "A survey of time synchronization protocols for wireless sensor networks," in *International Journal of Computer Science and Mobile Computing*, 2013, vol. 2, no. 9, pp. 100–106.

[24] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978.

[25] S. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *IEEE INFOCOM*, vol. 1, March 1999, pp. 227–234.

[26] R. Laplace, "Applications et services DTN pour flotte collaborative de drones," Ph.D. dissertation, Université Sciences et Technologies - Bordeaux I, december 2012.

[27] V. Autefage, "Amirale formal model - a service discovery and collaboration system formalism based on dynamic graph relabeling," LaBRI - University of Bordeaux, Tech. Rep., February 2015, https://hal.archives-ouvertes.fr/hal-01114961/.

[28] V. Autefage, A. Casteler, S. Chaumette, N. Daguisé, A. Dutartre, and T. Mehamli, "Parcs-s2 : Park cleaning swarm supervision system – a position paper," in *9th AIRTEC International Congress*, October 2014.

[29] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa, "Stochastic properties of the random waypoint mobility model," *Wireless Networks*, vol. 10, no. 5, pp. 555–567, September 2004.

[30] A. Casteigts, "The jbotsim library," *CoRR*, vol. abs/1001.1435, 2010.

[31] ——, *JBotSim*, http://jbotsim.sourceforge.net.

[32] T. Imielinski and J. Navas, *GPS-Based Addressing and Routing*, IETF, November 1996, rfc 2009.

[33] J. Polk, J. Schnizlein, and M. Linsner, *Dynamic Host Configuration Protocol Option for Coordinate-based Location Configuration Information*, IETF, July 2008, rfc 3825.

[34] IEEE, *802.11*, 2012.

[35] ——, *802.15.1*, 2005.

[36] ——, *802.15.4*, 2011.

[37] V. Autefage and D. Magoni, "Network emulator: A network virtualization testbed for overlay experimentations," in *IEEE CAMAD 17th*, september 2012, pp. 266–270.

[38] V. Autefage, *Network Emulator for mobile universes (NEmu)*, http://nemu.valab.net.

[1] http://mugen-sas.com

[2] http://www.defense.gouv.fr/dga

[3] http://aquitaine.fr