PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Bruno José Olivieri de Souza

An Approach for Movement Coordination of Swarms of Unmanned Aerial Vehicles Using Mobile Networks

Dissertação de Mestrado

Dissertation presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática, PUC-Rio as partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Markus Endler

Rio de Janeiro, March 18th, 2015

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Bruno José Olivieri de Souza

An approach for Movement Coordination of Swarms of Unmanned Aerial Vehicles Using Mobile networks

Dissertation presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática, PUC-Rio as partial fulfillment of the re-quirements for the degree of Mestre em Informática. Approved by the following commission:

Prof. Markus Endler
Advisor
Departamento de Informática – PUC-Rio

Prof. Noemi de La Roque Rodriguez
Departamento de Informática – PUC-Rio

Prof. Francisco José da Silva e Silva
UFMA

Prof. José Eugenio Leal
Coordinator of the Centro Técnico Científico da PUC-Rio

Rio de Janeiro, March 18th, 2015

Bruno José Olivieri de Souza

The author graduated as BSc. in Computer Science at UFF (Universidade Federal Fluminense) – 2003.

Bibliografic data

Souza, Bruno José Olivieri

An Approach for Movement Coordination of Swarms of Unmanned Aerial Vehicles Using Mobile Networks / Bruno José Olivieri de Souza; advisor: Markus Endler. – 2015.

67f ; 30 cm

Dissertação (Mestrado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2015.

Inclui bibliografia

Informática; Unmanned Aerial Vehicles; Swarm Coordination; Flying Robots; Mobile Networks; Publish/Subscribe; Drones; I. Endler, Markus. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

Primarily, I have to thank my family for all support on previous challengers that brought me here, to reach this one itself and furthers. My Family is the main pillar of all my past and further achievements, my ground. My wife, my parents, my two sisters, are the center of my success.

I would like to thank my advisor Markus Endler that was a perfect - and patient - helm and a tutor of my work. Thanks to all my LAC friends, your support as research mates was very important to me. To all Professors of the Department I had the opportunity to get in touch with during this phase I would like to thank you for their time and lessons. I would like to thank my classmate Rafael Pereira, present in all holidays and weekends of hard studies and mutual support.

Finally, I also would like to thank the Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ) and the Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior (CAPES), that partially sponsored this research.

I dedicate this work to my wife Carol, my life.

Abstract

Souza, Bruno José Olivieri; Endler, Markus. **An Approach for Movement Coordination of Swarms of Unmanned Aerial Vehicles Using Mobile Networks**. Rio de Janeiro, 2015. 67p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

This work presents an approach to coordinate swarms of Unmanned Aerial Vehicles (UAV) based on Internet communication provided by mobile phone networks. Several activities can be done by several UAVs flying in formation, such as surveillance and monitoring of mass events, search and rescue tasks, control of agricultural pests, monitoring and forest conservation, inspection of pipelines and electricity distribution networks or even military attack and recognition missions. Coordination of UAVs swarm can be branch in two sub-problems: communication between members of the swarm and the algorithm that controls members' behaviors regarding their movements. The proposed solution assumes the use of a smartphone coupled with each UAV of the swarm, in order to provide the required level of reliable communication on the mobile Internet and run the proposed algorithm for the coordination of swarms of UAVs. Experiments were performed with emulated UAVs and WAN mobile networks. The results have demonstrated the effectiveness of the proposed algorithm, and have shown the influence of the network latency and the UAV speeds on the accuracy of the movement coordination in the swarms.

Keywords

Informática; Unmanned Aerial Vehicles; Swarm Coordination; Flying Robots; Mobile Networks; Publish/Subscribe; Drones..

Resumo

Souza, Bruno José Olivieri; Endler, Markus. **Uma Abordagem para a Coordenação Movimento de Enxames de Veículos Aéreos Não Tripulados usando Redes Móveis.** Rio de Janeiro, 2015. 67p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Esta dissertação apresenta uma abordagem para a coordenação de enxames de Veículos Aéreos Não Tripulados (VANT), baseada na comunicação via Internet provida pelas redes de telefonia móvel. Um grande número de atividades pode ser coberto com uma missão conjunta de VANTs voando em formação, como a vigilância e monitoramento de grandes eventos, tarefas de busca e salvamento, controle de pestes agrárias, fiscalização e conservação florestal, inspeção de oleodutos e redes de distribuição elétrica ou mesmo em missões militares de ataque e reconhecimento. A coordenação de enxame de VANTs pode ser fatorada na comunicação entre os membros do enxame e o algoritmo de controle e inter-relacionamento entre os membros. A solução proposta consiste no uso de smartphones acoplados a VANTs capazes de prover o nível necessário de comunicação confiável sobre a Internet móvel,, e processar o algoritmo proposto para a coordenação dos enxames de VANTs. Experimentos foram feitos através da emulação de VANTs e redes de dados de telefonia que demostraram a eficácia do algoritmo proposto e analisam o impacto da latência de rede e da velocidade dos VANTs sobre a precisão da coordenação de movimento de enxames.

Palavras-chave

Informática; Veículos Aéreos Não Tripulados; Coordenação de Enxames; Robôs Voadores; Redes Móveis; Publicador/Assinante; Drones

Contents

List of Figures

1. Introduction

According to Austin [1], an Unmanned Aerial Vehicle (UAV) can be simply described as any aircraft operated by a computer system and a radio-link, rather than a human aircrew/pilot. In according to the UAV definition adopted in the present work, these machines can be referred to as flying robots, due the fact that they are capable of a certain level of self-control. This work focuses on a specific class of UAVs, or rather a set of multiple UAVs. These can be viewed as a swarm of UAVs, working together to execute a single and common task.

1.1 Motivation

A coordinated swarm of robots has numerous highly important applications, such as search and rescue missions, surveillance and monitoring of mass events, etc. In particular, swarms of UAVs equipped with cameras can become a strategic tool for military and civilian applications. Nevertheless, in order to ensure a good coverage of a site of interest, or record all data pertinent to a specific event/occurrence, it is necessary to coordinate the movements of all robots comprising the swarm. Another important motivation for achieving real-time coordination is the need to avoid collisions between the robots, which may be operating at the same altitude level. Thus far, several studies have been conducted on this topic, and the applications of these robot formations can be found in different disciplines, from engineering and artificial intelligence. While specifics vary, all extant works in this filed share a common feature—the coordination algorithm is based on heuristics, without due analysis on their correctness and, usability [2]. Thus, in order to address the deficiencies in the empirical approaches adopted in these areas, a new line of investigations has emerged, whereby the aforementioned problems are addressed from a distributed perspective [3].

According to Çelikkanat and Şahin [4], a robot swarm consists of a large number of homogeneous, autonomous and relatively simple robots with local sensing and communication capabilities. In practice, non-communicating, communicating or

networking robot swarms can be found. In the first case, the robots navigate completely autonomously and do not communicate their states/positions with each other, whereas communicating and networking swarm members exchange their position information in predefined intervals. This coordination facilitates a higher level of cooperation among them, both with respect to their movements and other actions. The main distinction between communicating and networking swarms is that, while the former category assumes a ubiquitous and uniform wireless communication medium, in the latter category the swarm itself establishes and maintains an ad hoc communication network by searching for, and adjusting to, the best radio signals. Although networking robot swarms may be mandatory for certain applications in remote locations, communicating swarms are a natural fit for urban environments, where one can take advantage of the wide-range coverage and ubiquity of mobile cellular networks.

The last decade has witnessed rapid progress in micro aerial robots—smaller autonomous vehicles with dimensions not exceeding 1 meter width/length and weighing under 1 kg, allowing their flight in indoor spaces [5]. Quadcopters are in this group, and their technology has significantly matured in recent years[6]. In this context, of particular interest for this investigation is providing support for UAV swarms equipped with cameras. More specifically, the focus is on those used for surveillance and public security during mass events, such as concerts, festivals, political demonstrations (e.g., the recent Brazilian "Vai-pra-Rua" movement), New Year's Eve, etc. When deployed in such events, swarms of UAVs would not only enable a wide-angle view of the environment, but also provide the operator with the ability to quickly spot points of turmoil and help identify the people involved.

1.2 Problem Statement

Site coverage and exploration through a swarm of mobile robots is important for several applications such as surveillance, intrusion detection, and distributed sensing in general [7], where it is impossible to pre-instrument the area of interest with stationary sensors, and/or where Points of Interest (POIs) within this area change from time to time. In this case, a swarm of mobile robots could move towards POIs steered by an (human) operator, and if their sensing capabilities around a POI need to be improved or enlarged, then the swarm of robots should move in a rigid formation.

However, movement coordination in swarms of robots is a challenge, since it requires reliable communication and timely mutual sharing of the robot's states. And if a

specific swarm formation is to be maintained, then it must be ensured that their current positions and other movement parameters are timely communicated, the robots are always kept within safe distance from each other so that potential collisions are avoided. In this work, we will focus on UAVs as the mobile robots and on cameras as the sensors carried by them.

For example, consider the following scenario: at New Year's Eve, around one million people gather at the Copacabana beach in a peaceful celebration. Nevertheless, local foci of turmoil, robbery and rampage are likely and can potentially occur at this event. In order to prevent these minor incidents from escalating further, the metropolitan police of Rio has set up a fleet of twenty autonomously flying UAVs, equipped with cameras and in charge of monitoring the event. In addition to the fly-by-wire patrol flight mode (along pre-determined trajectories), each UAV is also capable of switching from patrol flight mode to swarm flight mode for turmoil recording. In swarm flight mode, several aircraft enter a circular flight formation around the UAV that first spotted the POI, e.g., place of rampage. As a result of this capability, the set of UAVs not only increments the number of cameras able to record the scene, thus widening the perspective, but also becomes less susceptible to the risks of vehicle knock-downs, which is always a risk in such cases.

A swarm of UAVs can be a very useful surveillance tool, as it provides ground security personnel with pertinent and timely images and information that can be used to get situational awareness and better plan action. When UAV swarms are deployed, they allow better coverage of vast or very dense areas from the sky. This advantage is yielded by the ability of multiple UAVs to function in a formation, coordinating their image capture to cover a larger area on the ground.

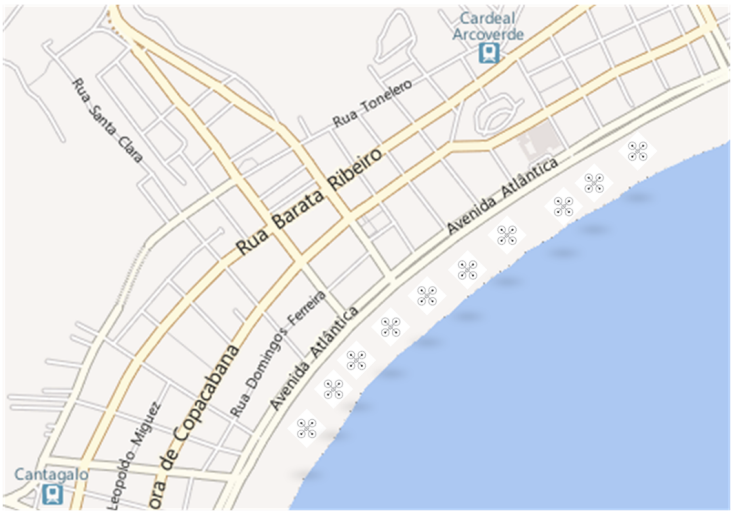

Figure 1 - Several UAV distributed along Copacabana beach.

1.3
Objective

The goal of this work is to devise an approach to coordinate a group of UAVs which should work in tandem to cover dynamically allocated POIs in previously uncharted and instrumented urban areas.

Our approach aims to make possible for a human operator choose an UAV to act as leader and a number of slaves to flight with that leader as a swarm. In sequence, the leader recruits the requested number of slaves and these slaves will then properly position themselves in a circle around the leader. The human operator can then send steering commands to the leader to move towards a POI. The selected slaves should then continuously maintain their expected position in the swarm, around the leader.

1.4
Summary of our Approach

We refer UAVs flying autonomously, out of a swarm. as in Patrol mode. Similarly, UAV acting together are in Swarm mode, being leader or slaves. In order to coordinate movements of several UAVs in swarm mode in almost real-time, a suitable strategy to coordinate them must be devised and executed. Such strategy relies on a strategy to enable intra-UAV communication and a distributed algorithm run by all UAVs.

This work bases on UAV swarms achieving such patrol and swarm flying modes by relying on Smartphone-based UAV control and conventional mobile networks. This approach yields two main advantages: (1) it can be readily used for a wide array of commercial UAV airships that are capable/ powerful enough to carry an off-the-shelf smartphone; and (2) it extends the UAV remote control capabilities via the Smartphones which acts as a processing unit and a communication hub connected to the Internet. To obtain the necessary wide-area communication support required for the envisaged UAV swarm applications, the approach presented here relies on conventional 3G/4G mobile networks,since they provide data-links with good quality almost anywhere, especially in urban areas.

This work contributes to the extant knowledge and practical work in the field by proposing a novel coordination approach for the formation and continuous flight control of UAV swarms. A generic hardware architecture for UAVs forms the basis of this approach, whereby off-the-shelf Smartphones mounted on each of the UAVs provide the

means for swarm-internal communication, assisted by a coordination algorithm that utilizes group communication features of a mobile communication middleware.

Regarding the inter-swarm coordination control, we propose a distributed algorithm executed on each UAV to control their distinct behaviors and tasks when acting in swarm mode. In order to coordinate a swarm of UAVs, a set of related issues must be analyzed and treated in a suitable manner, such as: recruitment and selection of slave UAVs; position designation; location update control and others. All these elements define the UAV swarm functionality. More specifically, information on how a group of UAVs will cooperate within a swarm, which will be the swarm formation, how it will be maintained by the UAVs, and how they will communicate with each other to achieve this flight coordination. These and other computational problems will be addressed in this work, mainly in section 3.5. There, we will present and analyse our distributed algorithm for coordination of swarms. It should be noted this work does not propose an novelalgorithm that addresses completely the two main challenges studied in robotics and wireless communication—exploration area decomposition, and trajectory planning [8].

1.5 Organization

The remainder of this dissertation is structured as follows: in the next section, technological background of this work is briefly addressed; in Section 3, our approach for coordinating a UAV swarm is presented; Section 4 is dedicated to the hardware and software architectures and the software implementation; in Section 5, the results of an emulation-based evaluation of the swarm coordination and other considerations are discussed; Section 6 focuses on other research work related to Movement Coordination of Swarms of Unmanned Aerial Vehicles by Using Mobile Networks; and, finally, Section 7 provides the main conclusions yielded by this analysis and offers some suggestions for further work in this field.

2.
Background

This chapter presents the main concepts underlying this work, with brief explanations about UAV, UAS, Mobile Network characteristics, the SDDL Middleware, SDDL's GroupDefiner component, and the concrete UAV chosen for this work.

2.1
Unmanned Aerial Vehicles

As noted in the introduction, a UAV can be simply described as an aircraft with its aircrew removed and replaced by a computer system and a radio-communication link. The UAV itself is merely a part, albeit an important one of entire system, contributing to the intended functionality. For example, zeppelins are used for long and slow movement aerial surveillance; airplanes for long distances and quadcopters for agile movement hovering over dense urban areas. Even though technological advances have made it possible to design and implement increasingly sophisticated UAVs, UAV usage is not a new concept, as many projects aiming to produce aircrafts without crews have been proposed since beginning of the 20st century [9].

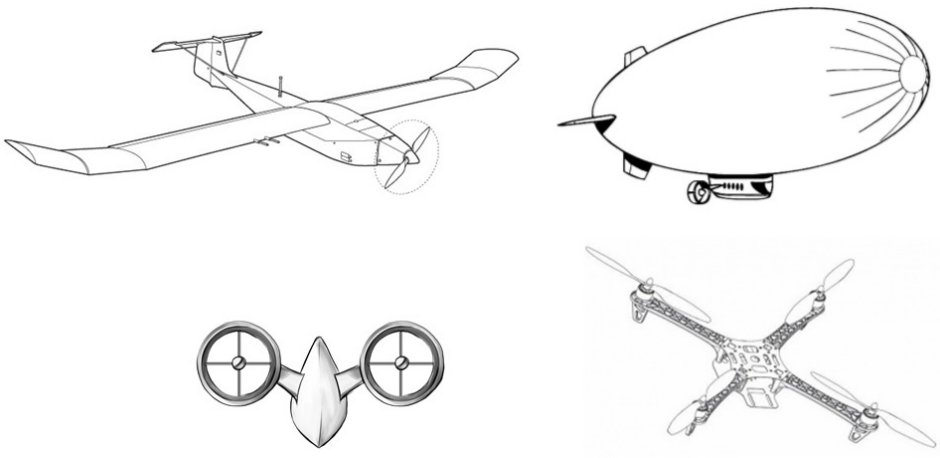

Figure 2 - Diverse sorts of UAVs: starting with a fixed wing above and left; a Zeppelin on its right side; a bicopter down and left and a quadcopter down right.

In the extant literature on the subject, UAVs are referred to by many different names, such as Drones, Flying Robots or RPAS (remote piloted aircraft systems). In all cases, the airship has just an electronic intelligence and control subsystem. Thus, flight is controlled either autonomously by onboard processors or through remote control by a pilot located on the ground. In this work, the focus is on UAV systems capable of performing certain autonomous tasks, such as takeoff, landing and traveling to a Point of Interest (POI) based on simple operator commands, as quadcopters are. These devices can be used in several applications, such as surveillance and monitoring on mass events, search and rescue tasks, high-precision agriculture, monitoring and forest conservation, inspection of pipelines and energy transmission towers or even military missions of hunting and attack.

Despite this variety of UAV models and a wide range of possible applications, this work approaches UAVs mainly from a computing perspective, instead of its flying characteristic, proposing a way of managing and coordinating several UAVs as an Unmanned Aerial System (UAS) with a single common purpose.

2.2
Unmanned Aerial Systems

Unmanned Aerial System (UAS) comprises several UAVs and a number of sub-systems, which include the UAV , their payload (i.e. sensors), the control station (aircraft launch and recovery sub-systems, when applicable, including recharge bays or transport reels. In those systems, the input required from the operators is usually limited to instructing the UAV to climb or descend, turn to the left or to the right, or even to perform tasks, such as patrol a certain area or perimeter. As noted above, UAS comprises a set of UAVs performing individual tasks in parallel, or acting together to reach a single common goal. Figure 3 presents an example of general arrangement with a swarm on right and an operator connecting theirselfes to the Internet to act together.

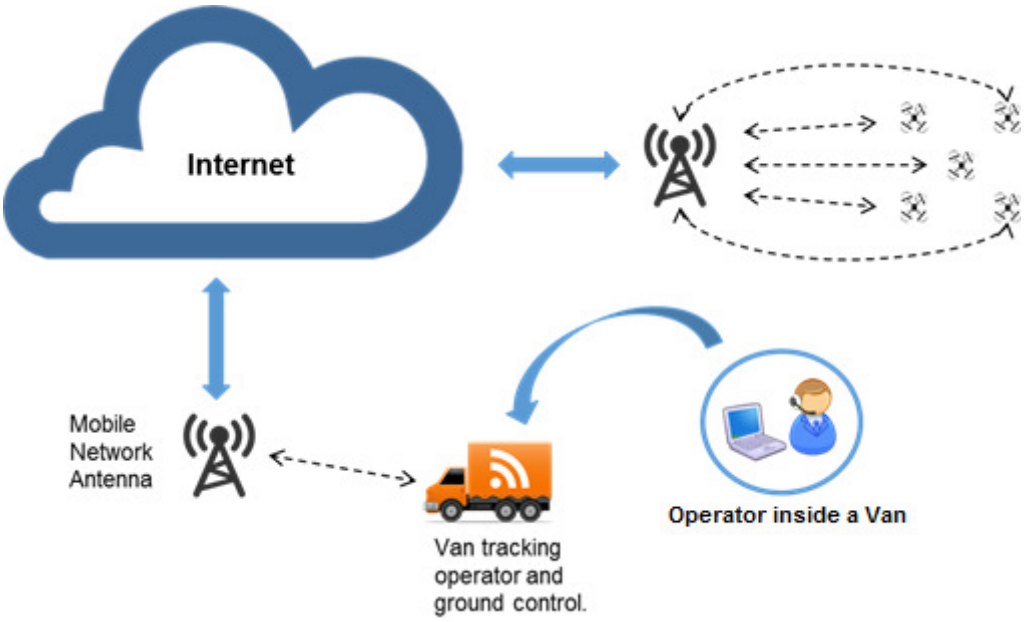

Figure 3 – Example of an UAS. Four slaves UAV surrounding a leader UAV and all of them in communication with a mobile ground station through Internet provided by a mobile network infrastructure.

2.3
Mobile Internet Networks

Regarding our algorithm communication needs of this work, we propose to use Internet access provided by mobile networks to establish communication between several UAVs, in order to enable and control a UAS.

The current mobile network technologies (2G/3G/4G) provide wide-range Internet connection through IP networks for mobile phones. Hence these phones may play the role of universal communication (and processing) hubs for the UAVs. This decision to use the mobile Internet brings some advantages and limitations, as noted below:

Pros

- Reliance on a well-established set of wireless communication standards and protocols;
- Ability to benefit from already installed and tested networks;
- Wide (and growing) wireless coverage in metropolitan areas;
- Continuous improvements and optimizations of the used protocols and radio systems;
- Potential to utilize a large set of smartphones off-the-shelf radios to reach communication.

Cons

- Higher and less predictable network latency compared with the latencies in MANs or WLANs;
- Higher package loss rates due the common handover process between base-stations providing telecom data services;
- Due to more frequent handovers caused by the high mobility of UAVs, it is harder to maintain continuous data streams in these networks;

The SDDL middleware used in this work and presented in section 2.5 has a mobile protocol that that help to avoid or mitigate the problems mentioned as items 2 and 3 in the above list.

2.4 Communication Paradigm

For communication, we chose the event-based paradigm Publish-Subscribe (Pub/Sub),. It brings advantages do to asynchrony and referential decoupling among publishers and subscribers, as well as its intrinsic support for optimized 1-to-N message delivery. eliminating the need for replicated messages with the leader's position, and enabling a single message dispatch and multiple reception by the slave UAVs.

By choosing this model, the goal is to reduce the number of messages as much as possible—a requirement arising from a Publish-Subscribe paradigm—while ensuring QoS guarantees, such as guaranteed delivery. We explore these two advantages by controlling the position of all swarm members by communicating only once per leader location update. More specifically, the analysis on section 5, goal is to achieve timely and efficient mobile Internet access by using the SDDL middleware, as will be described in more detail in subsequent sections.

2.5 SDDL

Figure **4** illustrate the Scalable Data Distribution Layer (SDDL) middleware extends OMG's DDS Data-Centric Publish-Subscribe protocol to mobile nodes through a scalable gateway approach. In this context, Gateways refer to the points of contact between WLAN and mobile Internet nodes and thus enable them to work together. Each Gateway is a DDS node that is also responsible for managing the wireless connections with a large number of mobile nodes and handling inbound and outbound messages from the mobile nodes (MNs). Thus, the middleware employs two communication protocols—

DDSs and MRUDP [6]. These govern the communication within the SDDL core, as well as the Mobile Reliable UDP (MR-UDP)[5][7] for the inbound and outbound communication between the SDDL core and the mobile nodes. The SDDL core can accommodate several types of nodes, including Gateways, Processing Nodes (i.e., nodes that process the mobile-sensor data generated by the mobile nodes, such as their geographic positions), GroupDefiners, or monitoring nodes operated by humans (i.e., Controllers), used for displaying the mobile node's current position (or UAV sensor data), managing groups, and sending messages to the MNs. SDDL has components responsible for catching undelivered messages. Those components tries to deliver them again or allow the user treat them as desirable and this feature is used by our work.

For enabling communication and coordination among the UAVs in a swarm, in this work, SDDL middleware is used as shown in Figure 4. It shows a complete setup of UAVs and a ground station trucks communicating through Internet by using SDDL. All UAVs are piggyback a smartphone that enables communication among them and with the ground stations, through the SDDL core.

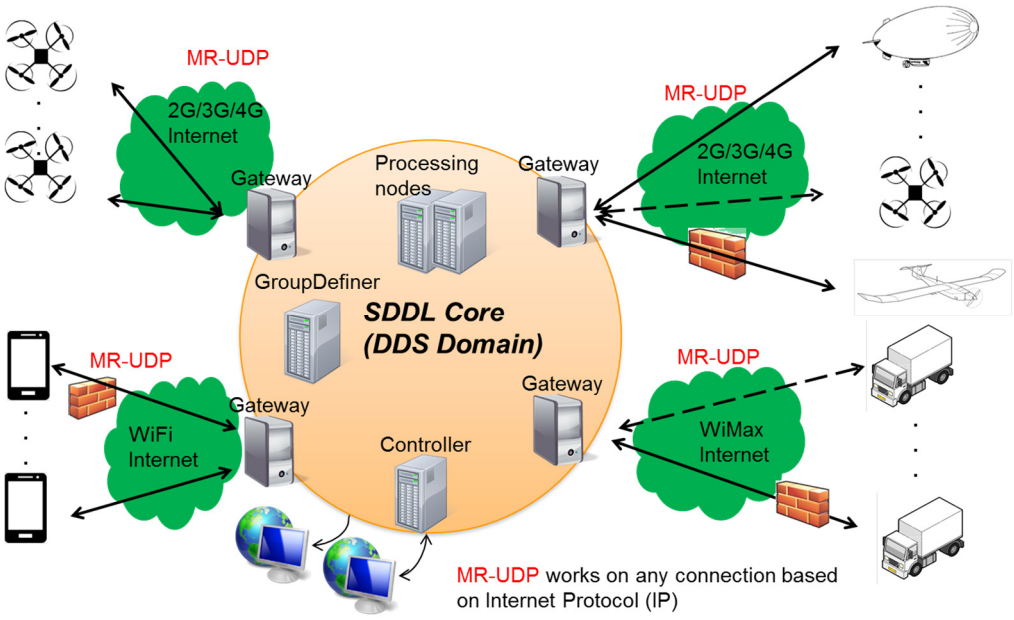

Figure 4 - SDDL Core running in a private cloud accessible through Internet. Each UAV member of a swarm uses its local smartphone to connect to the Internet reaching SDDL and ground station also reach SDDL through Internet.

2.6
Group Management in SDDL

The GroupDefiner, yet another SDDL Core node, is responsible for managing the group membership information of all mobile nodes, in our case, the UAVs. Its main task is to check all messages exchanged among UAVs and update the UAV's group membership accordingly. More specifically, the GroupDefiner dynamically groups UAVs by their current status, namely Free mode, Leader or Slave role. It is capable of managing simultaneously multiple groups in swarm mode, by using the ID of the Leader UAV to separate them. Whenever a GroupDefiner detects a group membership change for any of the UAVs, it announces this to all Gateways, allowing the messages to be groupcast accordingly. As a result, each position update message produced by the UAV designated as the swarm leader is automatically groupcast to all the UAVs that are in the same communication group (the slaves), without the need for each UAV to explicitly request to join the group. More information about SDDL's dynamic group management support can be found in the work of David et al. [10]. Figure 4 shows the main components of the SDDL middleware involved in the UAV communication and group management, with groups presented in the red ovals. Figure 5 shows a groupcast message sent by the Controller and being being delivered just for a group of quadcopters inn connected to different gateways to the SDDL core.

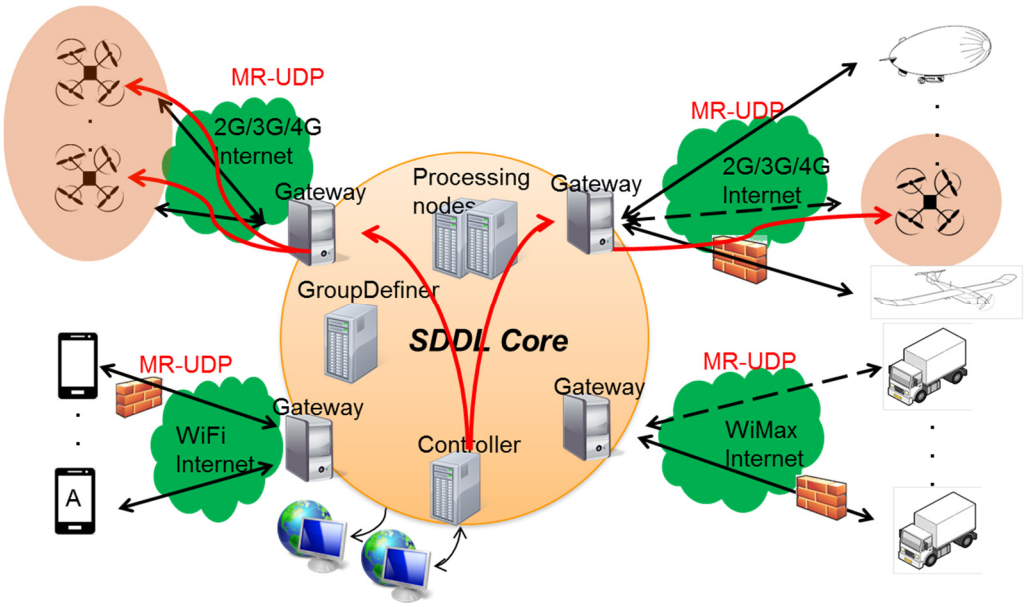

Figure 5 - Show a group message being delivered just for a group of quadcopters.

2.7
Choice of Unmanned Aerial Vehicle

Although several UAV types are in use nowadays, this work is based on quadcopters—a specific kind of multirotors—as these present some advantages for meeting the work's goals. First, because of to not being fixed-wing aircraft, quadcopters are capable of hovering over a stationary POI. Thus, the device can hold its position replicating helicopter capabilities. Yet, due to their mechanical simplicity, quadcopters have some notable advantages over helicopters. For example, instead of requiring complex and technically advanced mechanisms that enable the use of a simple rotor, as helicopters do, a quadcopter has four propellers with simple motor connection without helicopters rotors.

As a result, the aircraft is capable of moving around with four degrees of freedom, flying in three dimensions from a point A (latitude A, longitude A, height A) to a point B (latitude B, longitude B, height B). This feature by itself is very useful to implement coordination algorithms without concern of the aerodynamics, since the position and directional vector describing UAV motion are independent of aerodynamic lift pressure (which is an essential factor for airplanes), or even temperature or winds (as is the case for Zepelins).

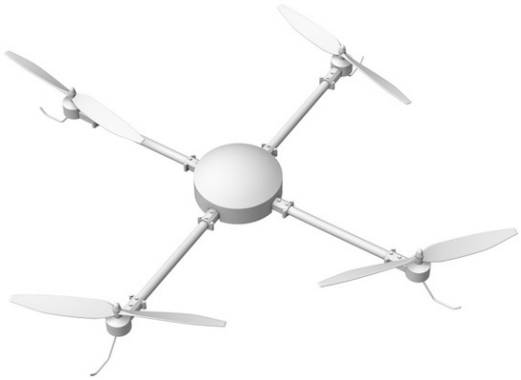

Figure 6 - A Quadcopter sample, with its four engines and propellers and a core with its flight controller.

2.8
UAV Swarm Coordination

During the last three decades, multiple autonomous robot systems have been studied extensively [11] [12][13][14][15][16]. This growing interest in these devices and

their potential applications was motivated by the premise that certain tasks could be better performed by employing multiple simple robots rather a single (or a few) powerful robot [17].

In their work, Pal *et al.* [17] noted that mobile robot systems have been studied in different disciplines within engineering and artificial intelligence, as a result of different physical and logical areas of interest. In the specific case addressed in this analysis, where the focus is on managing movements of flying robots in a swarm, some algorithm needs and problems that do not exist in usual swarm pattern formations arise and must be appropriately addressed. In fact, the main goal of their work is to enable UAVs to flock around their leader, which controls the entire formation, instead of each device trying to arrange itself independently. Nonetheless, it must be noted that each UAV is responsible for obtaining the correct position in the formation, and is guided by the information provided by the leader and a shared compass context. This context takes in consideration that, while each UAV has its own compass, all devices in the swarm share the same North Magnetic Pole calibration.

In the extant works in this field [2][3][16][18][19][20], some common models for distributed coordination algorithms have been defined, mostly based on Synchronous, Semi-synchronous and Asynchronous communication paradigms. The work presented here belongs in the Semi-synchronous category, and its robot-operation cycle can be summarized as follows:

Wait: A not synchronized phase, when a slave UAV waits for information of the leader's new location.

Look: A phase commencing at the point in which a UAV identifies its own position and contextual situation (such as battery level or actual direction or status) and is able to use this data to take decisions within the scope of the algorithm that governs its behavior.

Compute: Execution of the given algorithm, obtaining a target "point to go" (PG).

Move: Moving towards the PG.

As a general rule, a swarm of mobile robots typically comprises devices characterized by few computing resources [17]. However, this work uses a slightly different approach, based on Smartphones, which possess much more extensive resources than simple mobile robots.

3.
Coordination of Swarms of Unmanned Aerial Vehicles

This section describes the distributed coordination protocol running in each UAV, enabling them to form a swarm. It is responsible for coordinating the flight actions of multiple UAVs towards the common goal of flying in a specific swarm formation.

This protocol relies on the SDDL communication middleware and its group communication and management functions, which were described in previous chapter. The aim of communication management is optimizing the number of messages exchanged between the UAV swarm members for sharing their current positions and status.

The coordination protocol is leader-based in a sense that one of the UAVs steers the motion of the entire swarm around itself in an orderly manner. Once a leader is appointed by the operator, the remaining UAVs are queried and some are designated as slaves of this leader UAV and thus gather around it. In other words, it is the operator's responsibility to determine the number of slaves and the leader UAV, which will then be responsible for "recruiting" the designated number of slaves. The protocol is also capable of managing several swarms at the same time, using the SDDL Core Group Manager resources concurrently and proposed algorithm controls. Our work also addresses possible deadlocks or starvation issues by handling distinct swarm recruitment and release controls.

3.1
System Model

In order to fully understand the coordination protocol proposed in this work, it is necessary to outline the assumptions made, i.e. the system's model, as well as the main hypotheses related to the characteristics of the UAVs, the communication links and the positioning technology. The main assumptions are:

- Each UAV is equipped with the two sensors—GPS and a digital compass;
- Each UAV is able to carry a Smartphone with the required sensors;
- Each Smartphone has continuous mobile Internet access;
- UAVs do not fail or deplete their batteries during operation. If any of these events occur, the affected UAV disengages from its mission and returns to the launch area to land;

- The (wired) interfaces between the Smartphone, the integration module and the UAV's flight control board are reliable and the latency of the control commands is negligible;
- The flight control board of the UAV reacts reliably to steering controls, such as MoveToPosition(), IncreaseSpeed(), DecreaseSpeed(), GoToAltitude(), etc. This functionality is controlled logically by the Smartphone and executed analogously by the Flight Control Board, acting as a remote control;
- The wireless signal of the mobile network covers the entire geographic area of interest;
- The maximum transmission delay over the wireless link of the mobile network is δ ms. This assumption is described and analyzed in detail in section 3.6 below;
- The system is capable of avoiding collisions, as explained in section 3.6;

3.2 Hypothetic Scenario

In order to illustrate better the functionality of the proposed system, the previously described hypothetical scenario of Rio de Janeiro downtown in May 2013 is used, when several hundred thousand people gathered in an overall peaceful demonstration. Despite the protest's natural entropy, there were constant reports of local foci of turmoil, robbery and rampage. In order to prevent these small incidents from escalating further, Rio's metropolitan police could set up a fleet of twenty autonomous UAVs equipped with cameras and in charge of monitoring the event. Initially operating in fly-by-wire patrol flight mode (along pre-determined trajectories), each UAV would be capable of switching to a swarm flight mode, for turmoil recording or live surveillance of the situation. In this swarm flight mode, several UAVs would enter a flight formation around the UAV that first spotted a point of interest (a.k.a. the leader UAV). In adopting this strategy, the UAVs increment the number of cameras directed at the scene, thus widening the police's perspective on it. Moreover, the entire formation also becomes less susceptible to the risks of vehicle knock-down by individuals willing to hinder the surveillance.

The present work allow performing the illustrated surveillance with UAV by using mobile networks as an Internet provider channel and smartphones as radios and aboard processors nodes in each UAV.

3.3 Approach for UAV Cooperation

In this work, quadcopters were chosen as the preferred UAV type, for the reasons outlined in section 2.1. The swarm is said to be properly formed when the desired number of slaves surround the leader and assume their expected relative positions. As noted earlier, using airborne cameras could be a major advantage in such situations.

In order to increase the coverage of a given area, in the proposed model, multiple UAVs are assisting a single UAV, remotely controlled by an operator, as shown in Figure 3. This arrangement focuses on positioning the leader in the middle of a circle before arranging the slaves around it, thus increasing the area coverage. The operator, acting as a overseer, can, just by controlling the leader, direct several UAVs over a POI, thus increasing the visibility, as required.

3.4 Proposed Hardware

The approach adopted in this work, with the aim of building a controllable UAV, features a Smartphone-centered architecture design. In this architecture, one Smartphone is mounted on and dedicated to one UAV in the swarm and is responsible for local processing, wireless communication, sensing the airship's position and converting remote flight-control commands to the control board of the airship. Virtually any UAV can be used for this purpose, as the proposed architecture has a modular integration component that acts as a bridge between the Smartphone and the airship's control board. The only requirements are that the Smartphone has to run Java, and should have a GPS sensor, a digital compass and a 2G/3G/4G radio with Internet access capability. With respect to the airship functionality, the implicit assumption is that it has sufficient energy to perform the tasks, i.e., perform the takeoff and landing, reach the swarm, and do the swarm flight task area autonomously. The key idea is propose a functional decoupling to take individual advantages of Smartphones and a well tested UAV flight control boards.

The aforementioned requirements can be met by several Hobby Remote Controlled Models (R/C), such as aircraft, helicopters, multi-rotors and Zeppelins. For practical purposes, a specific type of multi-rotor controller device, the MultiWii quadcopter, was selected for this work, since it is based on open hardware and open source. However, the proposed approach could be easily adapted to other quadcopter flight controllers.

The choice of a quadcopter as the airship is due to its simple mechanical and control characteristics. As these devices have flight control boards, they can hover autonomously over a given spot and be controlled via simple commands, such as UP, DOWN, BACKWARDS, FORWARDS, LEFT, RIGHT and YAW (meaning turn left or right without moves, just turning the frontal angle). The flight control boards with the expected requirements are off-the-shelf components that can be purchased in Hobby-Model stores.

An Integration Hardware outlined in Figure 7 was projected to integrate the Smartphone, which runs the presented algorithm, with the UAV, enabling sending commands for movements. It is capable of receiving directions from the coordination protocol executing in the Smartphone, described in section 4 such as go forward or turn right. Moreover, it can translate them into ordinary remote control commands in Pulse With Modulation (PWM) encoding, which is widely used in Remote Control equipment's. The integration component has a physical interface linking it with the Smartphone via an USB cable and with the R/C with four Jump Cables, one for each channel expected by the quadcopter.

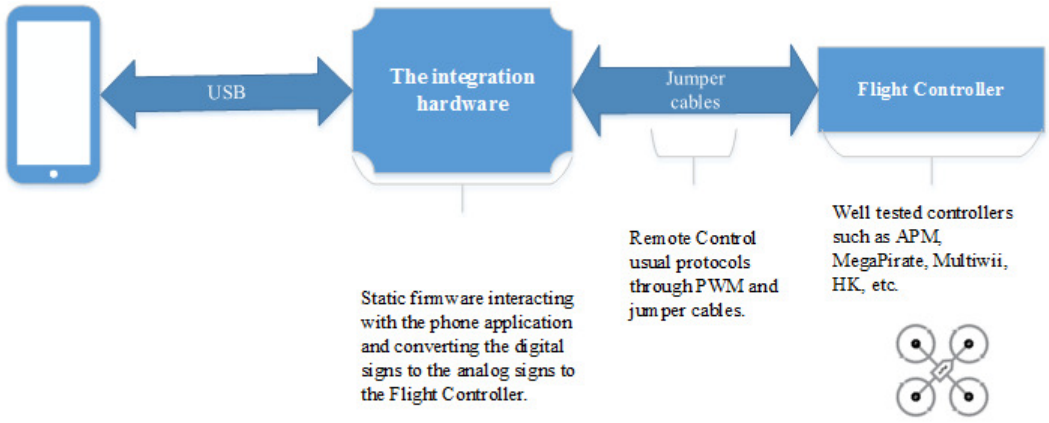

Figure 7 - The hardware architecture of the UAV.

3.5 An Algorithm for Swarm Coordination

In order to achieve coordination of the swarm, the following distributed algorithm is proposed, allowing each UAV to process the incoming commands itself, without requiring a central processing node. Instead, we simply propose an effective coordination algorithm for establishing and maintaining a particular swarm flight formation (i.e., where a leader UAV is uniformly surrounded by *n* slave UAV).

As indicated above, the algorithm relies on a mobile communication middleware, and all UAVs act independently.

3.5.1 System Description

In this work, it is assumed that the geographic region of interest is monitored by a set of m UAVs. Moreover, all UAVs will fly either in Patrol or Swarm mode. In Patrol

mode, all UAVs are autonomously patrolling the region within the virtual borders of a perimeter of interest, previously determined by the operators, and are transmitting camera images to the operators' console. In this mode, the UAVs are members of the group named FREEGRP, managed by the GroupDefiner component of SDDL. In both Patrol and Swarm modes, all UAVs send Location Updates (LU) with their current geographic position to the SDDL core, making it possible to maintain organized swarm and ensuring availability of leader control based on its position.

In Patrol mode, each UAV flies autonomously and can receive operator commands. These commands could be: a request to change the patrolling over an area, a request to "return to base" or the assignment of a particular UAV as a leader of a swarm (with a number of required slave UAVs). The possible UAV states are depicted in Figure 8.

In Swarm mode, a UAV may play the Leader or Slave role. In the latter case, each device starts to interpret only Leader commands, in order to calculate and reach its expected position. If, on the other hand, a particular UAV is assigned the role of a Leader, it receives coordinates for a desirable position from the operator and proceeds to that position. Once there, it stays there, hovering, and propagates its own coordinates to its slaves via a SDDL's groupcast. Figure 8 shows states exchanging between Patrol/Free, Slave and Leader modes.

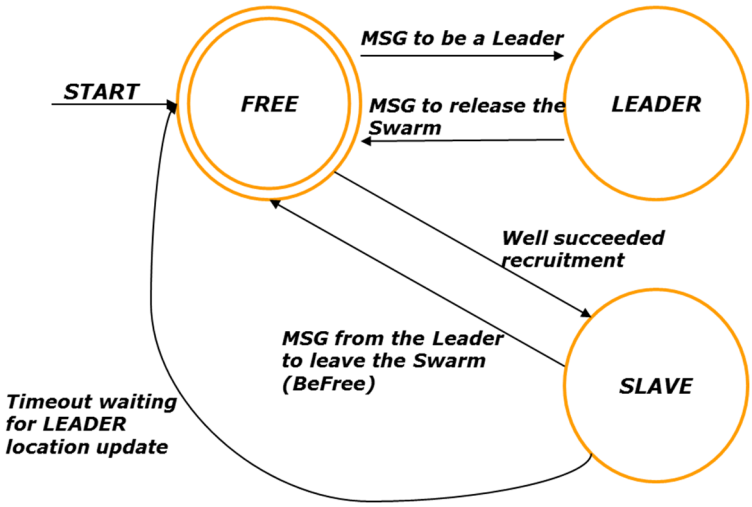

Figure 8 - UAV Swarm States: Free, Leader or Slave

Whenever Ground Control (GC) needs that a particular UAV should be escorted by a swarm (e.g. because some relevant event was spotted from some UAV camera image, this and other UAVs will enter the Swarm mode. For this, the GC operators define the UAV that will assume the Leader role and a number (say, *n*) of additional UAVs (from

the total of *m* devices) that will be requested to establish the swarm (where $n < m$). At this moment, all the UAVs in Patrol mode receive a group message (group FREEGRP) with a request to send their current GPS coordinates to the UAV designated as the Leader. Figure 9 show the swarm geometrical characteristics.

When the Leader receives Slave candidates' messages, it waits for a time interval equal to $2*\delta$, during which the remaining UAVs should transmit their position information. Once in possession of this data, the Leader selects the *n* "most appropriate" UAVs that are to become part of its swarm. The criteria used for selecting the slave UAVs vary, depending on the task: it might be the distance to the Leader, the airship's residual energy, or any combination of these or other airship data. The Leader UAV then sends its command informing all UAVs chosen as his slaves about their relative position to the remaining swarm members, which immediately switch to the Slave role and proceed accordingly. By intercepting these unicast messages, the GroupDefiner obtains the necessary data on the set of UAVs that will become slaves of the Leader and assigns them into the corresponding LEADERIDGROUP. Thus, from this point on, the Leader ID characterizes this entire group. Consequently, any message sent by the Leader is automatically groupcast by SDDL to its slave UAVs.

The swarm formation will assume a circular form around the Leader, with radius *r*, in order to widen visual coverage of the detected occurrence at the ground. All *n* slave UAVs will be positioned on this circle, $\theta = 360/n$ degrees apart from each other. For this formation to be executed, all slave UAVs will receive a unicast message from the Leader, providing θ and their respective relative positions on this circle (clockwise) around the Leader's position (xL,yL). More specifically, the first slave will take position (r*cos(θ), r*sin(θ)), the 2nd slave will be placed at (r*cos(2θ), r*sin(2θ)), and the *i*-th slave will position itself at (r*cos ((i+1)θ),r*sin ((i+1)θ)). All Slave UAVs acknowledge the receipt of this command and start moving to their *relative positions* in the circular formation around the Leader, which will remain unchanged during the entire swarm formation. The angle θ is measured from the direction designating geographical north, which is known by each UAV through their compass and is referred to as mutual North.

From this moment on, the Leader will continuously transmit its absolute position (*xL,yL*) to all its Slaves through groupcast to LEADERIDGROUP. This position-update message will be received by all slave UAVs within the $2*\delta$ ms time limit, and will allow them to constantly update their absolute positions and keep the swarm formation.

This mode of operation will remain until the GC operator sends a "swarm dissolve" message to group LEADERIDGROUP, which will instruct all UAVs in the swarm to return to Patrol mode. Once their mode of operation has been changed, all UAVs will resume their task of ordinary patrolling, which they perform autonomously.

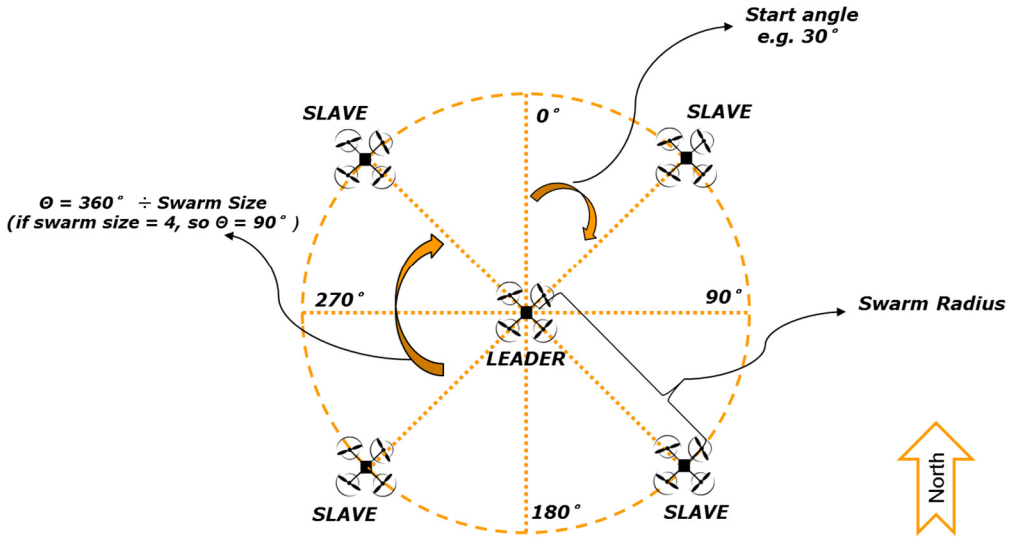

Figure 9 - Swarm sample describing its relative positions and main formation parameters.

3.5.2 Pseudo-code

In this section, we show the pseudo-code of the proposed coordination algorithm. All UAVs initialize with their variables myLeader as null and myStatus as FREE. Two message commands from SDDL are used — GRPCAST(group Id, message) and UNICAST(uav Id, message) — which send a newLeader message to the group FREEGRP.

```
1: public enum UAVMessageTypes {
2:     newLeader,                ▷ A new leader has been chosen
3:     slaveCandidate,           ▷ Candidacy message from a slave candidate
4:     beSlave,                  ▷ Leader confirmation to a candidate
5:     leaderUpdate,             ▷ Leader location update message
6:     beFree;                   ▷ Leader releasing a slave
7: }
```

```
1: public enum UAVSwarmStatus {
2:   FREE,                                                          ▷ Out a swarm, patroling
3:   LEADER,
4:   SLAVE;
5: }
6: public class SwarmCommand {
7:   public      UAVMessageTypes   type;
8:   public      int               idSender;
9:   public      int               newLeader;
10:  public      int               nSlaves;
11:  public      int               posSlave;
12:  public      double            slaveAngle;
13:  public      Double            leaderLatitude;
14:  public      Double            leaderLongitude;
15: }
16: procedure STARTUP()
17:     myStatus ← FREE                                             ▷ Starts Patroling
18:     myLeader ← null
19:     while true do                  ▷ A parallel thread receives msgs in OnNewSwarmMessage()
20:         if myStatus == FREE then
21:             hoverAround()                                       ▷ Stays Patroling
22:         if myStatus == LEADER then
23:             wait 2σ                ▷ When being a leader, stays updating its position to its swarm
24:             newMsg ← new SwarmCommand(myID)
25:             newMsg.myPosition ← readMyGPS()
26:             GROUPCAST(LEADERIDGROUP, newMsg)
27: procedure ONNEWSWARMMESSAGE(SWARMCOMMAND MSG)                   ▷ Runs in a thread to treat msgs
28:     switch msg.type do
29:         case newLeader
30:             newLeaderMsg_v1(msg)                                ▷ A command to became a leader
31:         case slaveCandidate
32:             addSlave_v1(msg)                                    ▷ A leader receiving a slave candidate msg
33:         case beSlave
34:             beSlave_v1(msg)            ▷ A slave candidate receiving confirmation msg from its leader
35:         case leaderUpdate
36:             leaderUpdate_v1(msg)                                ▷ A leader command/location for a slave
37:         case beFree                                             ▷ A release command to a leader or a slave
38:             myStatus ← FREE
39:             myLeader ← null
40:             mySpeed ← mySpeed − 60%
41: procedure NEWLEADERMS_V1(SWARMCOMMAND MSG)
42:     if msg.id ≠ myId then                                       ▷ It may become a Slave
43:         newMsg ← newSlaveCandidate(myId, myPosition)
44:         UNICAST(msg.id, newMsg)                                 ▷ Candidating for Slave
45:     else                                                        ▷ It should became a Leader
46:         myStatus ← LEADER
47:         swsz ← msg.SwarmSize
48:         SlaveCandidates ← ∅
49:         GROUPCAST(FREEGROUP, msg)                               ▷ Invitation for Slave Candidates
50:         wait 2σ for all slave candidate's positions             ▷ In a message receiver Thread
51:         mySlaves[] ← Choose(swsz, SlaveCandidates)              ▷ Starts with a callback from previous line
52:         i = 0
53:         for each mySlaves[] do                                  ▷ Confirmation msg to each choosen slave
54:             newMsg ← new beSlave(myId, slaveAngle)
55:             newMsg.slaveAngle ← (360/swsz × ++i) + 30
56:             UNICAST(mySlaves[i].id, newMsg)
57: // swsz stands for the number of slaves / swarm size
58: procedure ADDSLAVE_V1(SWARMCOMMAND MSG)
59:     synchronized (SlaveCandidates){
60:         SlaveCandidates.add(msg.sender, msg.pos)}               ▷ Slave candidates to sort after a recruiting timeout
```

Figure 10 - Swarm control algorithm.

```
1: procedure BESLAVE_V1(SWARMCOMMAND MSG)
2:     myLeader ← msg.sender
3:     myStatus ← SLAVE
4:     myAngle ← msg.SlaveAngle
5:     mySpeed ← mySpeed + 20%
6: procedure LEADERUPDATE_V1(SWARMCOMMAND MSG)        ▷ Salves setting its own position on a swarm
7:     tmp.Pos.X ← r.cos(myAngle)
8:     tmp.Pos.Y ← r.sin(myAngle)                     ▷ msg.position has Leader's position (xL, yL)
9:     destination ← tmpPos ⊕ msg.position            ▷ Slave corrects its position to destination
10:    moveToPosition(destination)
11:    // ⊕ stands for the addition of coordinates
```

3.6 Discussions

The control algorithm developed in this work is based upon three main pillars—the Smartphone-centric hardware architecture, the groupcast communication and the SDDL middleware and its dynamic group management capability, and the distributed coordination algorithm—each entailing both

```
1: String SwarmPair = "INSERT INTO SwarmPairs "                    ▷ CEP rule to select pairs of UAV
2:    + "SELECT luSlave AS slave, luLeader AS leader FROM"
3:    + "SlaveLocationUpdate.window AS luSlave, "
4:    + "LeaderLocationUpdate.window AS luLeader "
5: cepAdmin.createEPL(SwarmPair).setSubscriber(new PairSubscriber());   ▷ CEP thread to run above rule
6: function PAIRSUBSCRIBER(UAVLEADERSLAVEPAIR P)
7:     if (r − tolerance < distance(p.Leader, p.Slave) < r + tolerance) then
8:         return true
9:     else
10:        return false
```

benefits and limitations. A central requirement of the proposed architecture was to support UAV coordination in a wide-area setting, using a 2G/3G/4G mobile networks. The use of a Smartphone as the communication hub enables both a straightforward implementation of the coordination logic (realized by the proposed coordination algorithm) on a relatively powerful but yet light-weight device and a modular design. However, it also entails some constraints discussed bellow.

3.6.1 The coordination Algorithm

The coordination algorithm proposed here is a distributed algorithm with low message complexity, both in the phase of swarm formation and during the swarm mode flight. More specifically, as soon as a new Leader is determined with *n* slaves, it will broadcast a recruitment message to all other UAVs (*m*), some of which will reply, yielding a maximum of 2*(*m*-1) messages. Next, the Leader sends out unicast messages for the *n* selected slaves (resulting in *n* messages), which also determine the members of the LEADERIDGROUP. A swarm formation is considered acting correctly when its

slaves were recruited and are following its leaders as well. To ensure that the chosen UAVs are flying in swarm formation, the Leader only needs to send one groupcast message (to LEADERIDGROUP) each time it changes its position, as each of its slave UAVs will be able to recalculate its own absolute position. Thus, each slave recruitment process requires only a small and predictable number of messages. In case of an unsuccessful recruitment, another one will start.

3.6.2 Errors and Incorrect Message Faults

As the swarm formation runs from a distributed algorithm, potential scenarios involving deadlocks and starvation are possible to occur. Hence, during the recruitment phase, additional controls must be put in place in order to prevent deadlocks and starvations.

These issues will be explained taking a hypothetical scenario with ***n*** UAVs. Consider that two leaders (say, LA and LB) start their slave recruitment concurrently - approximately at the same time - whereby each is instructed to recruit ***m*** slaves, where ***m* > *n*/2**. As both Leaders are sending messages intended to recruit their respective slaves, it would be impossible for both to complete this phase successfully. In fact, if the entire system is capable of exchanging messages simultaneously and randomly, no leader would reach LEADER status and recruit the required number of slaves. In addition, if LA and LB make reattempts of recruitment after exactly same time intervals, the entire system could reach a livelock status or compromises the liveness property of the entire system. Further potential problems might arise due to recruitment messages being lost, causing the recruitment phases to fail. However, each of these problems have been considered in our algorithm, as will be explained in the subsequent paragraphs.

Despite the message delivery guarantees provided by the communication middleware (SDDL), our approach also includes a UAV status control flow which ensures secure swam formation and maintenance processes as shown in Figure 10. The UAV in the swarm could reach five different status, of which three are of longer duration, (FREE, SLAVE and LEADER) and two are just intermediary states (CANDIDATING and RECRUITING). To change its own swarm status, a UAV processes one and only one swarm message at a time respecting the arrival order. This process is depicted in Figure 11: when a UAV has myStatus = FREE, it processes only the swarm messages pertaining to the formation process. In addition, as the designated leader sends his recruitment messages only for and from FREE members, this step has the quantity of messages reduced by the fact that a specific group, FREEGROUP, is held by GroupDefiner. During

the recruiting period, the designated leader waits for the candidates' messages in RECRUITING status. When a UAV in FREE status receives the recruiting message, it may or may not respond, depending on its own priorities. When it does reply, it assumes the CANDIDATING status and waits for the confirmation from the leader.

In both cases, message losses, unexpected delays or even concurrent processes could lead to a deadlock or a leader starvation situation. In such a case, timeout and retry (reattempt) procedures are initiated. A UAV in CANDIDATING status remains in this status until it receives the recruitment confirmation or until a candidacy timeout of β seconds is reached, after which it returns to FREE status. A UAV in RECRUITING status keeps receiving candidacy messages (msgType = slaveCandidate) until a timeout of random(α) + β seconds (with $\alpha > \beta$), after which it counts the candidates to verify whether the swarm formation is complete. For leaders in recruitment state we use a random α time (in seconds) as timeout to make sure that – eventually – concurrent leaders in recruiting procedures will initiate their retries in different moments. Leader in recruitment state have a larger timeout period than slaves candidates ($\alpha > \beta$) in order to guarantee the availability of slaves after failed recruitments. If the leader has received replies (msgType = slaveCandidate) from the required number of candidates, it confirms the recruitment (msgType = beSlave); otherwise, it immediately starts a new recruitment process from the beginning. without further delays. Figure 11 and Figure 12 show the recruitment flow and timeouts behavior.

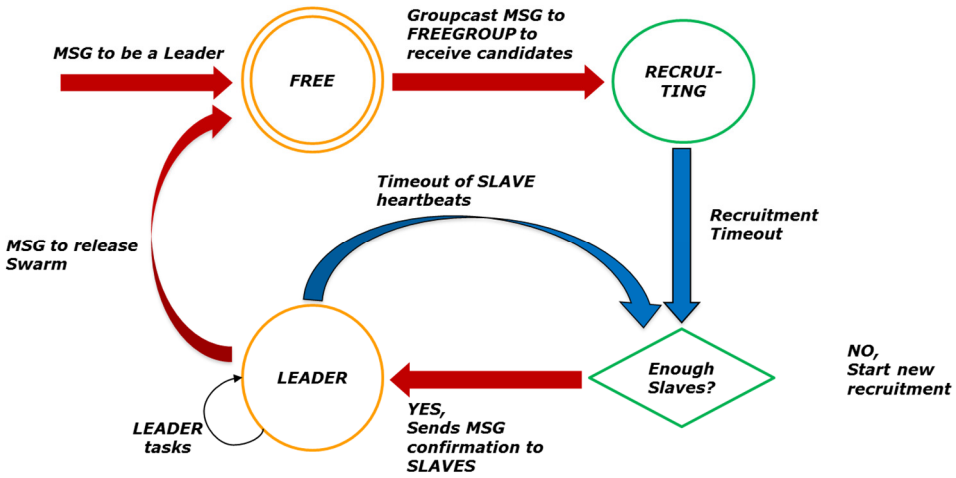

Figure 11 – Leader recruitment control flow.

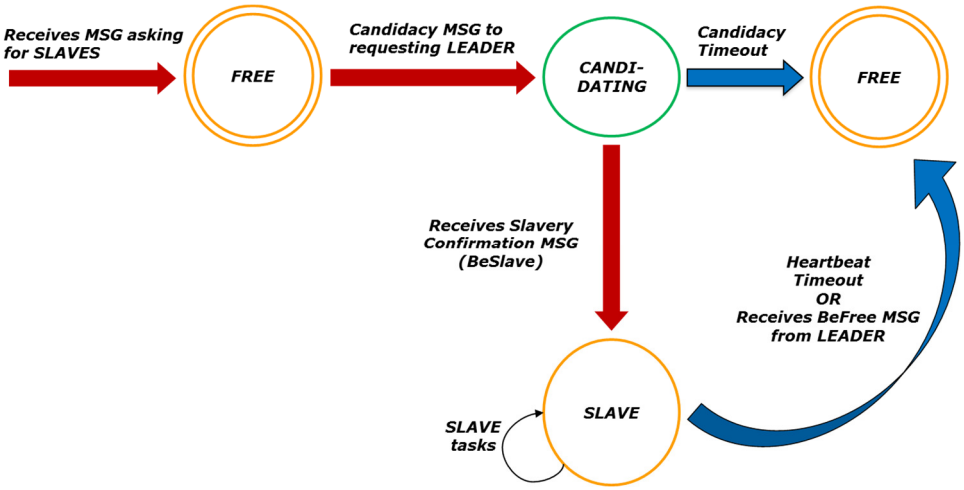

Figure 12 - Slave recruitment control flow

The protocol described above prevents deadlocks and starvation of multiple recruiting leaders in the entire system, by using timeouts for state decision tasks and random retries. In order to address incorrectness due the processing of wrong or fault messages, the proposed algorithm prevents wrong messages (such as location updates from unknown leaders and candidacy messages from a slave already in a swarm) from being processed. All processing status discards non-expected messages. However, in FREE and RECRUITING status, it would be possible for a maleficent UAV to cause a temporary unavailability of a recruitment phase by sending fake messages or never accepting their recruitment confirmation.

3.6.3 Collision Avoidance

To prevent collisions between UAVs flying in swarm mode, the proposed approach could assume that each UAV is equipped with sensors, such as a sonar or laser range, which could be used to continuously check each UAV's path relative to the other UAVs' positions. Alternatively, collision avoidance in the swarm could also be handled by using SDDL's groupcast, albeit employing a slightly different inter-UAV protocol. More specifically, each UAV would groupcast its position and direction vector to a processing node within the SDDL core, at high frequency. Next, this processing node would check for possible collisions among the members of a LEADERIDGROUP, and promptly alert the involved UAVs whenever necessary. However, this approach is difficult to determine

the wireless communication latency *Rl* required for implementing such collision avoidance mechanism. More precisely, *Rl* would need to be related to *Dm*, the safe distance among UAVs; the UAV cruise speed Cs; and the collision processing time Pt. Assuming that all UAVs groupcast their position + vector data at exactly the same time, and that—in the worst-case scenario—these UAVs are heading towards each other with a relative speed of 2**Cs*, the following simple equation can be used to determine the maximum communication latency:

$$2\,.\,Rl = \frac{Dm}{2\,.\,Cs} \; - \; Pt$$

As an example, let us consider that *Dm* is 15 m and *Cs* is 10 km/h (exceeding the parameters that would allow an operator to see a suspect situation, such as a robbery, from the air). In such a case, the total period of 2*Rl* + *Pt* is around 2.8 seconds, which is more than sufficient time to detect a possible collision with the communication latency used in the model.

This work uses a trivial method to avoid intra-swarm collisions. By setting distinct flight altitudes for each UAV, we do not require any inter UAV communication nor processing: However there is a disadvantage of this approach, which depends on the total number of UAV in use. As more UAVs are used the larger would be the difference of their flight altitudes. But, very distinct altitudes with equal or similar sensors could represent distinct data quality of the ground obtained by each UAV, e.g. for cameras, different image qualit would the obtained. Moreover this simple collision avoidance approachis not scalable.

3.6.4 Overcoming malfunctions

As both leaders or slaves can fail, both cases must be addressed to manage the problem of runaway slaves or a negligent leader. Slaves could stop functioning correctly in the swarm formation for several reasons, including component or software failures, and the same applies to leaders, which could stop acting as expected. To address these cases, a heartbeat strategy was adopted. In this approach, slaves that do not receive any messages from their leader within a period of β seconds reset their status to FREE and return to their previous missions in FREE mode. However, as this may result in a situation whereby a leader proceeds with a mission with a reduced, and insufficient number of slaves, a parallel stream of heartbeat messages from the UAVs in slave status was introduced. Thus, when a leader does not receive a heartbeat from any of its slaves within a period of φ seconds, it releases all current slaves (sending a message of type

beFree) and immediately starts a process for new recruitment. The approach to address malfunctions described above is quite simplistic but could be sophisticated under penalty of increasing more the number of messages. However, it would not present a significant contribution to the problem of swarm coordination, and therefore was not further exploited in this work.

As noted earlier, the focus is on improving the UAV swarm performance in open areas, utilizing the signal from GPS satellites. For this reason, the potential strategies are limited to those applicable to vast open areas. Thus, in this work, the vertical degree of freedom (have different altitudes) was removed from UAV, thus significantly reducing the complexity of the approach [21].

A further potential constraint imposed on this approach is attributed to the System Model, which does not address out-of-swarm collisions. However, this problem belongs to a family of collision avoidance problems typically addressed in Simultaneous Localization And Mapping (SLAM) approaches [22]. Such flying robots have been discussed in Richards *et al* [23].

3.6.5 Smartphone

A Smartphone is a mobile phone with processing and communication capabilities, added with embedded sensors [24]. In our approach a smartphone is used due to its 3G/4G interface, the embedded GPS and compass sensors, and its powerful processing resources, that makes it function as a local communicating, sensing and processing unit at the UAV. Moreover most of these devices are lightweight, powered by an own battery, well tested and widely available. In order to use the smartphone aboard, the wired interfaces between the Smartphone, the integration module and the fight control board have to be resistant to vibration, dust and moisture/rain. The Smartphone acts as main actor in a functional decoupling architecture proposed in section 3.4.

3.6.6 Communication layer

In order to use the mobile network connectivity provided by smartphones, a communication software is necessary that guarantees reliable message delivery in spite of handovers between the smartphone and mobile operator base stations, that may cause some messages to be lost. The SDDL middleware [10] is a communication layer providing with features do handle such problems and fits to our purposes due to some of

its keys features: the MR-UDP mobile protocol; its GroupDefiner capable of handling dynamic group memberships, and its ClientLib for Android.

MR-UDP is a reliable UDP that controls the basic message exchange between all UAVs and the SDDL core, the latter executing in a cloud or cluster or servers. MR-UDP handles packet/message losses by providing smart acknowledgments and automatic retransmissions until entire message reaches its destination. MR-UDP also guaranties FIFO message delivery order at application level.

However, MR-UDP cannot be used without some penalties in our approach. Low quality networks incur frequent packet losses and to handle that, MR-UDP retransmits lost packages until the messages has been correctly delivered at the destination. This retransmissions however, increase the latency of the overall communication.

SDDL also provides a built-in groupcast and group management support for mobile nodes through its GroupDefiner, which has been extensively tested [25]. It groupcast capability makes it possible to send messages to groups without knowledge of its members. Itis also possible to use groups where its members continuously enter and leave the group according to their state, such as their role in the swarm (free, slave or leader) or other context information (i.e.: battery level; distance from a point; etc.). These features are used in recruiting and coordination phases in our algorithm.

Unfortunately, FIFO message delivery order is guaranteed only for individual MR-UDP connections, but groupcast messages lack of such guarantee. Thus, as older groupcast messages could arrive at some UAVs after recent ones, this temporal mismatch could lead some UAVs to set an “old” position as its target. In order to mitigate this problem, a sequential, auto-increment counter was introduced in groupcast messages sent by the leader. Thus, slaves only process messages that are newer than the last one processed.

ClientLib is a library that defines an asynchronous communication API for mobile clients on the Android platform, and hides from the application programmer much of the MR-UDP’s details. Thus, this libabry rovides an implementation layer ready to use in smartphones and was used in our simulations presented in chapter 5.

4.
Implementation

The proposed approach was tested and analyzed using a virtual simulation environment comprising emulated UAVs, a world map environment provided by Google Maps API, the SDDL middleware executing in a private cloud and an emulation of WAN Internet connectivity within a LAN. This chapter presents the architecture of the implementation and details about emulated UAVs, group management through GroupDefiner, WAN simulation, the coordination algorithm itself and the measurement components.

4.1
Architecture

Figure 13 shows the simulation environment used for testing the proposed approach consisting of four parts.

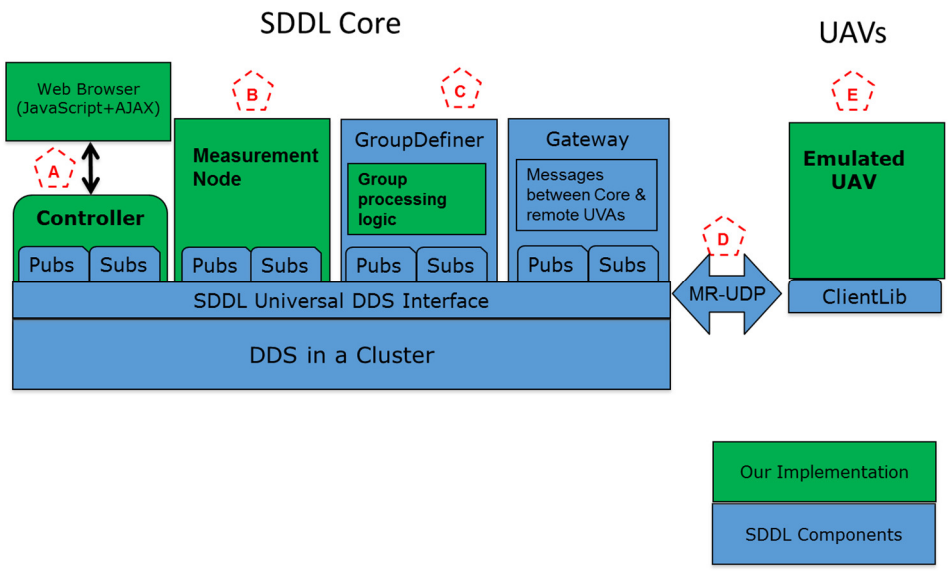

Figure 13 - Implementation parts: A: Core Node for command simulations; B: Measurement Node using CEP to verify swarm formation accuracy; C: Logic for group membership; D: Emulated MR-UDP with injected delays and message losses; E: Thread pool running emulated UAVs.

Part (A) of the testing environment is a collection of tools used to control and measure data yielded by the UAV pool. One of the tools is a map view interface showing the emulated UAVs in action, flying in a region, as well as a console to send commands to individual UAVs (such as set a Leader with n slaves and control its movements) and displaying basic data and UAV status information from the emulated UAVs. Figure 14 shows an example of the control map in use. This part also includes a Complex Event Processor (CEP) [26] engine and rules that are used to analyze and evaluate the swarm's formation accuracy from the stream of location update messages received from all UAVs. CEP rules process a geometric verification that checks if UAVa are correctly in swarm and how many of them are not in the swarm as shows Figure 19.

Figure 14 - Control Interface: (1) Shows six UAV flying in FREE status, patrolling around; (2) Shows four Slave UAV surrounding a central Leader before the swarm be in correct formation and (3) Shows a swarm correctly flying arround.

The second part (B) relates to the SDDL core executing in a LAN. This part contains a Gateway that handles all communication among UAVs in a WAN and all other processing nodes. The third component is the GroupDefiner, responsible for running the group logic and for controlling groupcasts.

The third part (C) is responsible for emulating different delays corresponding to different mobile networks. It implements configurable delays when any ClientLib routine is necessary to send a message.

The last part (D) refers to a thread pool. For each emulated UAVs a control thread runs other three threads regarding the flight simulation (Physical UAV layer), the communication layer (ClientLib layer), and the Coordination Algorithm.

The software necessary for the implementation of the proposed approach was programmed mainly in Java. Some web interfaces used JavaScript, while measurement rules were written in Java using the CEP library ESPER[27].

4.2
Simulation Control and Measuring Node

Those are two SDDL core nodes in charge of two main tasks: (A) it implements a web interface capable of providing information and control the simulation by a showing subset of context information from selected UAVs (such as its status or speed), and controlling a designed UAV. This web interface, shown on Figure 15, uses Google Maps API to present a world map; and (B) accept and process location updates from all emulated UAVs in order to evaluate the swarm's formation accuracy.

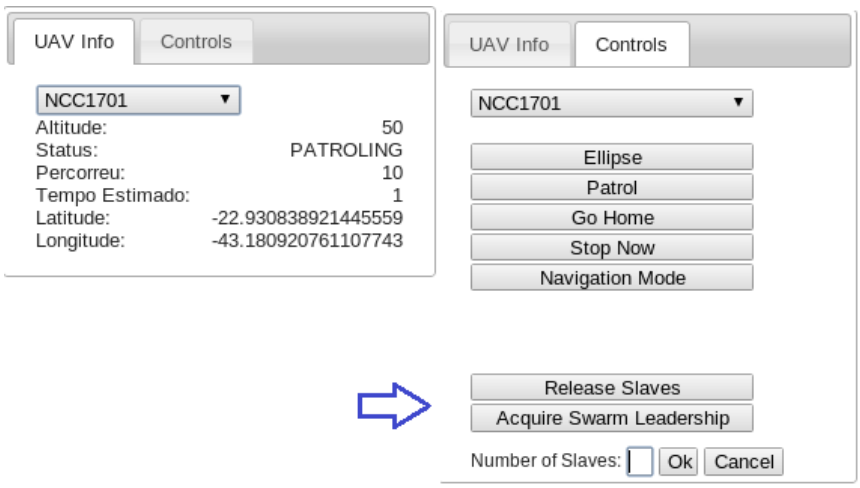

Figure 15 – Left side of figure shows internal UAV information such as state and coordinates. Right figure shows some controls at Simulation Control. Arrow shows recruitment specific options.

Those messages are duplicated, in a process that is part of the middleware's internal features and takes place with or without the proprietary interference. Consequently, it does not require any additional time or processing power from an emulated UAV. The aforementioned message stream is the input to a set of CEP rules and provide location information about the swarm formation. The scripts are capable of analyzing the entire swarm continuously by checking whether the swarm is correctly positioned. A UAV is correctly positioned if and just if it is Swarm Radius plus or less a *"tolerance"* from the Leader as shown on Figure 19 and CEP continually runs as briefly presented as follows:

```
1: String SwarmPair = "INSERT INTO SwarmPairs "                    ▷ CEP rule to select pairs of UAV
2:    + "SELECT luSlave AS slave, luLeader AS leader FROM"
3:    + "SlaveLocationUpdate.window AS luSlave, "
4:    + "LeaderLocationUpdate.window AS luLeader "
5: cepAdmin.createEPL(SwarmPair).setSubscriber(new PairSubscriber());   ▷ CEP thread to run above rule
6: function PAIRSUBSCRIBER(UAVLEADERSLAVEPAIR P)
7:     if (r − tolerance < distance(p.Leader, p.Slave) < r + tolerance) then
8:         return true
9:     else
10:        return false
```

Figure 16 –Fragment of the CEP rules used in measurement node from (B) on Figure 13, mainly regarding the detection strategy describe bellow.

Figure 16 shows a sample rule used in CEP. Lines 1 to 4 is a CEP query responsible to get location updates from each (Leader, Slave) pair and insert it in an event line. Line 5 set a thread that continually consumes such events and runs the Function PairSubscriber(n,p). Finally, function PAIRSUBSCRIBER checks if the pair generated is in the correct "tolerance" zone of the leader.

4.3 GroupDefiner

The GroupDefiner is a processing node of the SDDL core that defines - and manages group membership relations - one or more UAV groups, each of which with its own membership logic. These logics continuously update the group members of each group (depending on the messages sent by the UAVs) and inform all the Gateways of SDDL. Thus, when a UAV or a processing node sends a groupcast message, this message is transmitted correctly to all group members. Each time the SDDL Core receives a location update or any other groupcast message from an UAV, the GroupDefiner makes a copy of this message and sends an update to the group members.

The protocol presented in subsection 3.5 relies on three groups in order to coordinate swarms, namely FREE, SLAVES or LEADERS. As the name indicates, FREE group includes all UAVs that are not participating in any swarm and is the group used for recruitment purposes. SLAVES and LEADERS are used just for simulation purposes. Those groups are continually and check UAVs status contained in any messages that reach GroupDefiner.

Swarm formation maintenance is accomplished by the Leader sending groupcast with a Location Updates and Slaves re-calculating their desirable location. Each of its slaves can correct their absolute position that corresponds to its expected relative position (in respect to the Leader) in the swarm formation. As the group ***ID*** corresponds to that of

the leader, when a message arrives from an UAV, the GroupDefiner processes it in order to add the UAV to its respective group.

4.4
Mobile Network Emulation

As the entire simulation described here was performed without actually using Smartphones and telephony-based mobile Internet, workarounds were implemented in order to emulate the two main problems that might have impact on the wide-scale operating scenario—network latency and eventual message loss. These characteristics of mobile telephony were artificially inserted in the simulation environment to test their respective consequences on the accuracy of the swarm formation.

The mobile network latency was emulated by using an internal variable (Δt *per say*) in the message-sending functions. For this purpose, each time a UAV sends a message, rather than being processed immediately, it was sent into a ClientLib internal queue. The elements in this queue were processed at Δt ms intervals, with Δt varying according the mobile network assumed in test (2G/3G/4G), thus causing a delay. For example, assuming an average latency of 50ms for internet access in 4G networks, each time a UAV sends a message, 50ms would elapse before it is actually transmitted. So, for each message from a UAV *A* to a UAV *B* takes $2\Delta t$ (one Δt from UAV A to core and one Δt from core to UAV).

Message loss is the second network behavior that was emulated, and it was approached in a similar way as the introduction of network latency. Another variable denoted by ψ hold the probability (in percentage) of message-delivery success. Thus, each time a UAV tries to send a message, it uses ψ*'s* value to determine whether the message will or not be actually transmitted. For instance, if $\psi = 90\%$, every time an UAV requests the sending of a message, the network emulation code generates a random value between 1 to 100, and discards the message if the outcome falls in the interval [90-100]..

4.5
Emulated Unmanned Aerial Vehicles

The Emulated UAV implemented for this work, bases on three main layers as shown in Figure 17 and the first of which implements communication. It implements and uses ClientLib [28], allowing the SDDL middleware to be used over Mobile Internet Network with certain guarantees. It also implements methods for sending and receiving messages, through listeners with callbacks to provide received messages to second layer, the Coordination Algorithm layer.

The second layer comprises the Coordination Algorithm *per se*, running between the other two layers. It sends and receives messages from ClientLib layer, obtains simulated coordinates and directions from UAV Movement Emulation layer, calculates the (emulated) UAV's target positions and executes the coordination algorithm, as detailed in item 3.5.3.

The third layer, UAV Movement Emulation, is responsible for emulating an UAV. It controls internal variables, such as Altitude, Speed, Direction and Coordinates. These variables are continuously updated by a thread that runs indefinitely logics that describes the airship's movement physics and processes the external flight control commands received through Coordination Algorithm layer. This also relies on the shared compass concept. All emulated UAVs when are created and inserted in the processing pool has their virtual front pointed for north of the world map sited on subsection 4.2, take this north as universal and each emulated UAV compare their direction with that.

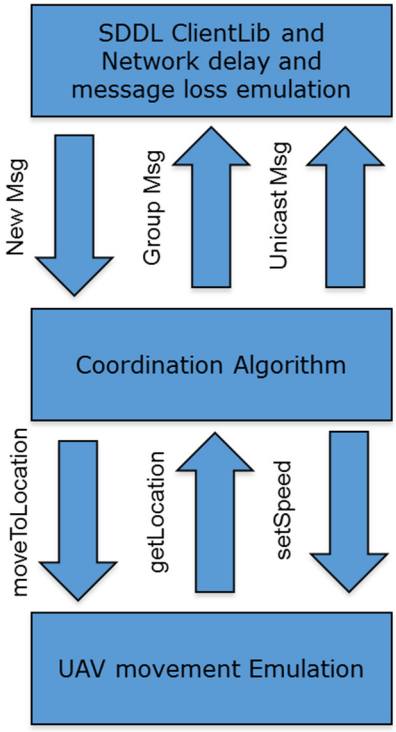

Figure 17 - This figure shows the UAV implementation performed by a three layers: communication, UAV Movement Emulation and a distributed algorithm.

The UAV Movement Emulation also implements routines defined by the coordination algorithm, such as GOTOPOINT(LATITUTE, LONGITUDE, HIGH) or GETACTUALLATITUDE(). These methods provide a modular environment for algorithm implementation, making possible to port the others two layers for a real UAV without significant changes.

4.6
Coordination Protocol

The coordination algorithm was implemented as a layer of the emulated UAV. Its main responsibility is to receive swarm-specific messages to process and to send commands to the UAV Movement Emulation. The coordination algorithm layer is a thread that runs the algorithm from section 3.5.2 and interacts with both other levels, in order to displace the UAV to its next desirable position, and exchanges messages using the communicating layer. It also has the callback functions that handle all arrived messages regarding the coordination.

Each Emulated UAV start runs as other three threads, one for each layer presented in section 4.5. It was code in Java and has more details than section 3.5.2 regarding overall error controls which we highlight as follows more detailed than section 3.5.2 as follows in Figure 18.

```
1: procedure NEWLEADERMSG_V2(SWARMCOMMAND MSG)                          ▷ Enhanced version
2:     if (msg.id ≠ myId)&&(myStatus = FREE)&&(not amIcandidating) then ▷ It may become a Slave
3:         candidacyLeader ← msg.id
4:         amIcandidating ← true
5:         newMsg ← newSlaveCandidate(myId, myPosition)
6:         UNICAST(msg.id, newMsg)                                      ▷ Candidating for Slave
7:     else                                                             ▷ It should became a Leader
8:         if (msg.id = myId)&&(not amIcandidating)&&(myStatus = FREE) then
9:             myStatus ← LEADER
10:            swsz ← msg.SwarmSize
11:            SlaveCandidates ← ∅
12:            GROUPCAST(FREEGROUP, msg)                                ▷ Invitation for Slave Candidates
13:            wait 2σ for all slave candidate's positions              ▷ In a message receiver Thread
14:            confirmSlaves(msg)                                       ▷ Starts with a callback from previous line
15: // swsz stands for the number of slaves / swarm size
16: procedure CONFIRMSLAVES(SWARMCOMMAND MSG)
17:     if (SlaveCandidates.size >= msg.SwarmSize) then
18:         mySlaves[] ← Choose(swsz, SlaveCandidates)
19:         i = 0
20:         for each mySlaves[] do
21:             newMsg ← new beSlave(myId, slaveAngle)
22:             newMsg.slaveAngle ← (360/swsz × ++i) + 30
23:             UNICAST(mySlaves[i].id, newMsg)
24:     else
25:         SlaveCandidates ← ∅
26:         newLeaderMSG_v2(msg)

27: procedure ADDSLAVE_V2(SWARMCOMMAND MSG)                             ▷ Enhanced version
28:     if (amIrecruiting) then
29:         synchronized (SlaveCandidates){
30:             SlaveCandidates.add(msg.sender, msg.pos)}
31: procedure BESLAVE_V2(SWARMCOMMAND MSG)                              ▷ Enhanced version
32:     if (amIcandidating)&&(candidacyLeader = msg.id) then
33:         myLeader ← msg.sender
34:         myStatus ← SLAVE
35:         myAngle ← msg.SlaveAngle
36:         mySpeed ← mySpeed + 60%
37: procedure LEADERUPDATE_V2(SWARMCOMMAND MSG)                         ▷ Enhanced version
38:     if (myStatus = SLAVE)&&(msg.sender = myLeader) then
39:         tmp.Pos.X ← r.cos(myAngle)
40:         tmp.Pos.Y ← r.sin(myAngle)                                  ▷ msg.position has Leader's position (xL, yL)
41:         destination ← tmpPos ⊕ msg.position                         ▷ Slave corrects its position to destination
42:         moveToPosition(destination)
43:         // ⊕ stands for the addition of coordinates
44: function CHOOSESLAVES(INTEGER N, UAVLIST SLAVECANDIDATES)
45:     synchronized (SlaveCandidates){
46:     Collections.Sort(SlaveCandidates)
47:     uavList slaves ← null
48:     for i = 0 to n do slaves.add(SlaveCandidates[i])
49:     }
50:     returns slaves
```

Figure 18 - Coordination algorithm with some implementation details.

4.7
Execution Flow

The execution runs starting Emulated UAV thread, than SDDL middleware (Gateway, GroupDefiner and MTD agent) starts and thereafter CEP node and Simulation Control node. After a successful start all activities can be controlled through the Simulation Control node and statists collected from CEP node.

Several parameters have to be set to run a simulation. The main set of variables used in the test scenarios are defined as follows:

- ***CS***: Cruise Speed: is the default speed shared by all emulated UAVs in the test scenarios. However, for UAVs in Swarm mode acting as Slaves, we used a slightly higher speed than *CS*. *CS* were set at 2, 5, 10 and 15m/s.

- ***SLS***: SLave Speed: All UAVs acting as slaves in swarm mode must fly a bit faster than the UAV acting as their leader. This condition is necessary to ensure that a swarm formation can be build, at first place. For example, let us assume that Leader A (La) recruited Slave B (Sb) while it is moving towards a target that is in the opposite direction of Sb's initial position. Or else, if La is moving and Sb needs to take a position in the swarm formation at the opposite side of La. In this case, if both UAVs were flying at the same speed, it would be impossible for Sb to overtake La and reach its desired position in the formation, as shown Figure 19. For this reason, in the evaluations of swarm accuracy for different emulated cruise speeds, CS , SLS was set to 120% of CS due the evaluations results explored and presented in chapter 5.

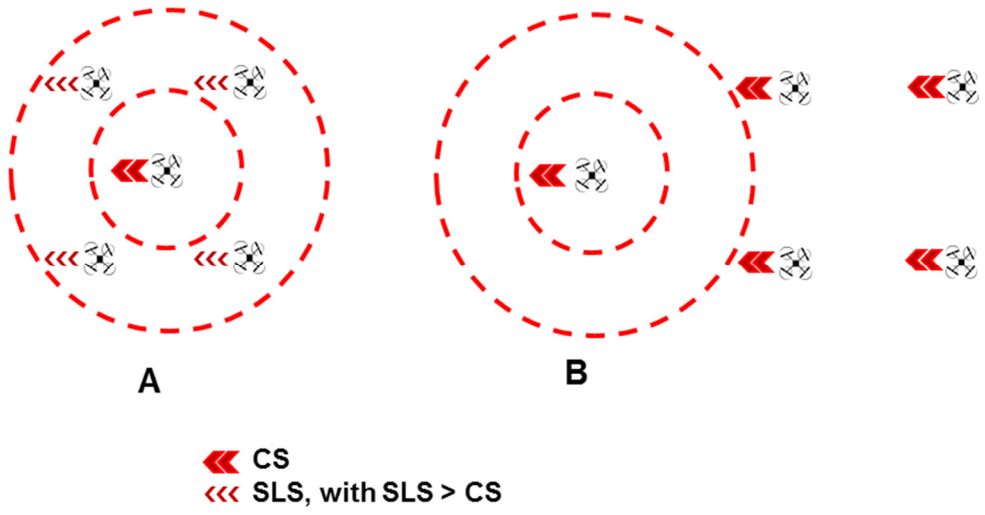

Figure 19 – (A) shows a correct Swarm formation. (B) shows all UAV going from right to left; in this case, the Slaves cannot reach their leader because all of them are moving with same speed and direction. The two circles represent the expected area that UAVs are supposed to be.

- ***UAV Height***: is the base flight height of all emulated UAVs and is set to 30 meters. However, each emulated UAV's flight height is *Height* plus the UAV's *id* (1, 2, 3, etc). Consequently, intra-swarm collisions are avoided, since individual UAVs are vertically separated by 1 meter, and always fly at different heights.
- ***Swarm radius*** or ***r***: It is the radius adopted for the swarm. By default its was set to 30-meter due our set of flight tests
- ***β***: Beta, candidacy timeout, a 3-second recruiting timeout waiting period was chosen, after which a slave offers itself to take part in a formation. If it does not receive a confirmation from the designated leader within this interval, it returns to its previous mission.
- ***α***: Alfa, recruitment timeout, set as 5 seconds. Leaders that do not reach their desired number of slaves during the recruitment phase will initiate their reattempt strategy after this time interval has lapsed.
- ***μ***: Mi, timeout for Leader heartbeats (in seconds). In our simulations, we set this parameter to 15 seconds. If a slave in a swarm does not receive a heartbeat message from its leader within this interval, the slave will leave the swarm and return to its previous task.
- ***φ***: Phi, timeout for Slave heartbeats (in seconds): If a Leader does not receive a message from one of its slaves during this interval, a it will release the entire swarm and recruit a new one. In our simulations, we set this parameter to 15 seconds.
- ***Ψ***: Psi, probability (in %) of a message loss, used in the mobile network emulation, to simulate bad network conditions. This value is discussed further in chapter 5.
- ***Δt***: Network latency (in *ms*), parameter used in the mobile network emulation. This work tests different values of network latency in chapter 5.
- ***ρ***: Rho, the frequency with which each Emulated UAV use as a time trigger to send location updates to the SDDL Core. Set at 6 Hz, and used to establish the position and obtain a set of UAV internal variables. Based on this parameter, the UAVs' positions are updated on the map.

5.
Evaluation

Evaluations were performed in order to test whether the coordination algorithm developed and implemented in this work performed as expected. This evaluation allowed us to identify some issues and to fine-tune some of the swarm's details (such as each member location, speed, status, etc. Moreover, it provided the opportunity to assess the impact of various details, such as the swarm's radius, UAV speeds and communication latency, on the protocol behavior. These scenarios are discussed in the context of the message-delivery guarantee provided by the SDDL middleware.

5.1
Experimental Setup

All experiments and tests were performed in the same environment, consisting of a single machine running Linux distribution, kernel family 3.x and the Java distribution OpenJDK 7. The machine had an Intel i7 quad- core (with Hyperthread Technology[1], enabling the execution of up to eight processes simultaneously), 8 GB of RAM memory.

A single machine was used in order to eliminate external network influence (such as switch or router delays, which could vary in different tests). Moreover, by running the evaluations on a single machine, we mitigated the potential of any other possible out-of-control interferences.

5.2
Analysis of Movement Accuracy and UAV speeds

In order to establish and maintain a swarm formation correctly, as shown in Figure 9, it is necessary for Slave UAVs to be faster than their leader UAV. For this reason, this section analyses different relative speeds among leader and slaves and discusses its relation to the accuracy of the swarm formation.

This first experiment was conducted, aiming to evaluate the dependence of the accuracy of the coordination between the leader and its slave UAVs—ignoring possible mobile network latencies that may occur in a real flight situation—on the relative speed among the UAVs. For this purpose, a simulation was initiated using a leader UAV, escorted by four slave UAVs, equally distributed around it in a circle with a 30 meter

radius (i.e., at positions 30°, 120°, 210° and 300° relative to the common North direction). The Leader was instructed to move along a pre-defined, irregular and closed trajectory. It was assigned three different speeds (LS), same with three fasters slave speeds (SLS), relative to the speed of the leader. Along the leader's trajectory—which included sharp turns in all directions—the difference between the current and the target position in the swarm formation of each slave UAV was measured. This difference was recorded each time a swarm member received a location update from the leader UAV (at a default of 6 Hz frequency). This difference in positions (current *vs* target) is thus the instantaneous measure of the swarm formation accuracy/imprecision. The average of these differences measured along the entire trajectory was calculated for all slave UAV, to estimate the overall imprecision of their movement coordination. The trajectory and the aforementioned measurements were repeated four times, yielding almost 500 data points for each UAV. For this measurements, we had to define at which point a UAV technically enters the swarm formation, since all of them approach the Leader from a distant position, after successful recruitment. In this respect, the beginning of the swarm formation was defined as the point at which a slave UAV reached twice of Swarm Radius distance for its actual target position. In other words, a slave UAV is getting in formation and its positions start to be measure when it at most two radius distance to its desirable position.

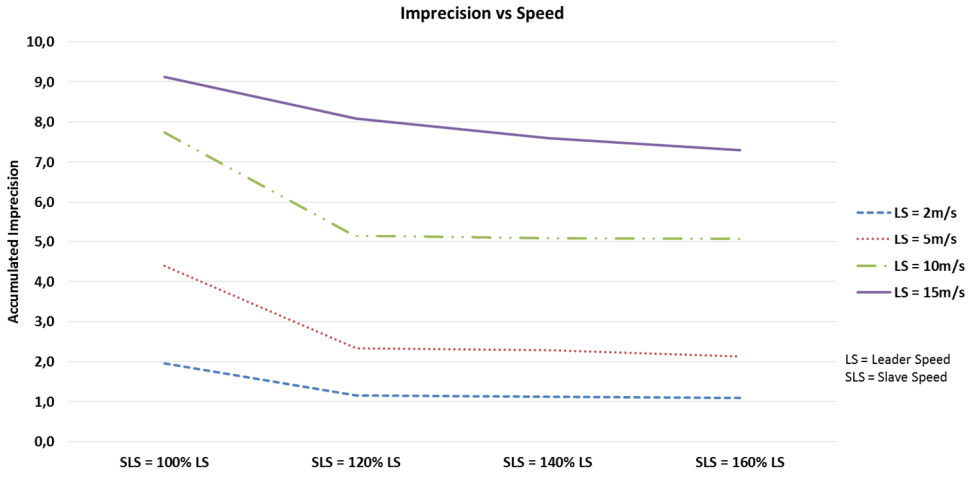

Figure 20 - Swarm formation accuracy versus proportions between leader and slaves speeds.

As can be seen from the graph in Figure 20, the swarm formation imprecision increases almost proportionally with the speed of the leader UAV, with the minimum value at 2 m/s. Another interesting finding that emerged from this simulation is that the swarm formation can be maintained quite well when the slave UAVs' speed is set just 20% above the speed of the leader UAV. Moreover, it could be observed that further

increases in the relative speed (of slave and leader UAVs) do not produce any significant improvement in the UAVs' swarm movement precision. The only exception was found for the high-speed scenario, where the leader moves at 15 m/s (54 km/h). Of course, in a real-world scenario, not only the 3G/4G communication latency, but environmental conditions (wind, etc.), will have significant impact on the movement coordination precision, and will thus influence the best possible speed ratio between the leader and the swarm UAVs. In the next set of simulation experiments, the effects of the emulated network latency on the precision of the UAV swarm movement precision is assessed.

5.3
Analysis of Movement Accuracy and Network Latencies

To evaluate how the network latency affects the swarm formation, we emulated eight network latencies and defined four different UAV speeds for the leader, and let the slave UAVs move at 120% of the leader's speed (SLS = 1.2 × LS). These simulations were run under the same physical conditions as of the previous experiment, and the swarm was also composed of four slaves equally distributed on a circle around the leader with 30-meter radius,

For latency tests, two distinct flows of message were produced by all UAV (and transmitted at a frequency of 6 Hz): (1) messages pertaining to the measurement itself, that were instantly delivered to the SDDL Core, without any artificial / emulated transmission delay; and (2) swarm messages between UAVs, with artificial delays produced by the SDDL layer displayed in Figure 17 (see section 4.5), in order to mimic communication latencies. For this second message flow, we used eight different delays: 0ms, 50ms, 100ms, 150ms, 200ms, 210ms; 225ms and 250ms. For each emulated delay, the leader speed was set at 2 m/s, 5 m/s, 10 m/s or 15 m/s. As previously, the measurement starts after all slave UAVs have reached half the distance to the swarm radius. From this point on, each time a measurement message (from any UAV) reaches the Measurement node in the SDDL core, it will compute the swarm's imprecision, in order to check whether the swarm is correctly structured.

Figure 21 shows the concept of proximity ratio or tolerance that consists in lower limit distance between a slave and its desirable position in a swarm. This is used as a trigger to start statistics about for evaluations, when a slave reach at first time in a swarm we consider that its finish the approach to its swarm. From this moment on, all location updates from a slave informs its distance from desirable position to be used in the CEP measurements.

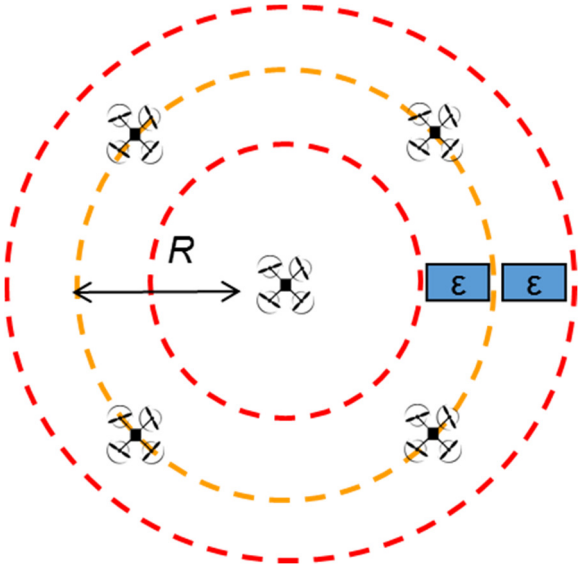

Figure 21 - The mid circle shows the Swarm radius *R*; (2) and (3) Most internal and external circles, shows the radius tolerance.

In this context, we consider a valid formation is reached when the distance of all slaves to their leader is within 90% to 110% of the swarm's radius *R*, as shown in Figure 21. Every message receipt and respective check (referred to as cycle) is repeated 2000 times for each UAV, resulting in 10000 messages being analyzed each cycle. The entire test cycle was repeated four times, and the average of 40.000 messages was taken to check slave locations *versus* its desirable locations. Therefore, the percentage of times that the swarm was correctly structured is used as a measure of the swarm formation accuracy as shown in Figure 22. After the four runs off the experiment for a particular speed and emulated network latency are completed, the average is computed.

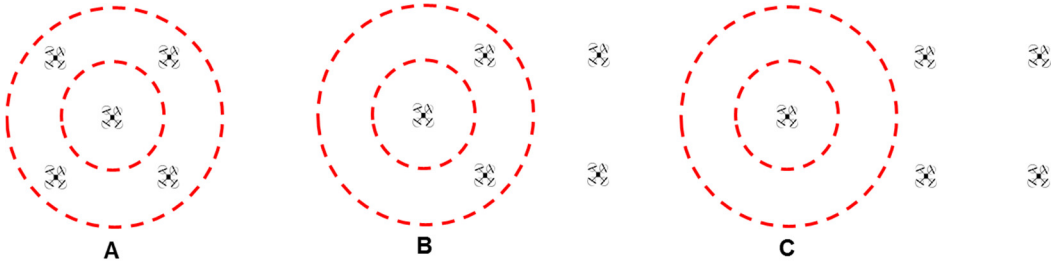

Figure 22 - (A) shows a swarm with 100% correct slaves (100% correctness); (B) and (C) shows a swarm with 0% correctness.

The simulation findings are summarized in Figure 23, which presents the data pertaining to all tested speeds and latencies, with a swarm radius of 30 meters. The graph

represents the relationship between latency and swarm correctness (vertical axis), with curves representing Leader Speed (LS) of 2, 5, 10 and 15m/s., The horizontal axis presents the emulated latencies, by 50ms increments. As can be seen from the graph, the best results are obtained at the (LS = 2 m/s, 100%) coordinate, while a point denoted by (LS = 15 m/s, 0%) is an example of a combination of leaser speed and network latency that is ineffective in controlling the formation.

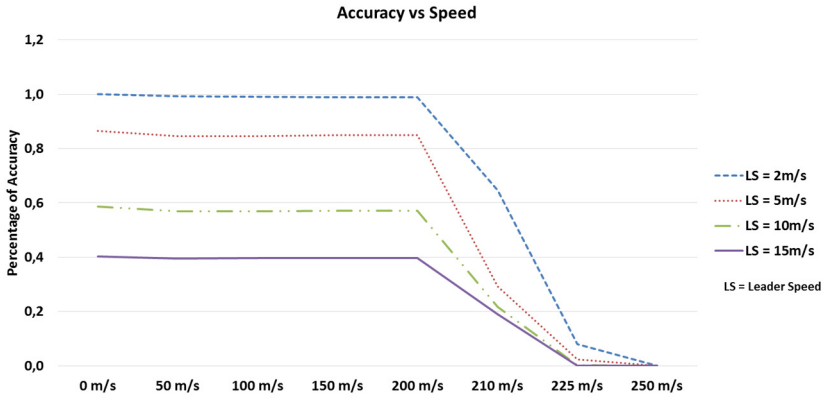

Figure 23 - Graph showing results of different leader speeds and network latencies and their combined impact on the accuracy for the swarm formation.

As can be seen from the graph of Figure 23, slower swarms can yield better results, as a greater number of "correct" states are identified. At LS = 2 m/s, 100% correctness can be achieved, even at 200 ms network latency (communication delay). On the other hand, with LS = 5 m/s, correct formation was reached in only 80% of cases, when the same latency were considered. This leads to the conclusion that the "*Leader Speed/Communication latency"* ratio is the key aspect that determines the swarm formation accuracy.

The experimental results also showed that all series exhibited a non-linear and non-proportional shape. For all the leader speeds, one can identify a maximum latency that can be supported, after which the swarm formation's accuracy decreases significantly. These limits appear somewhat close to each other on the graph, indicating the presence of other factors potentially contributing to this behavior as discussed below.

5.4 Analysis of Movement Accuracy and distance from the Leader

In the scope of the present analysis, these results indicate at least another parameter that influences the degree of accuracy of swarm formations that is the distance between the leader and the slave UAVs in the swarm, i.e. the swarms' radius R. In order to

evaluate how the swarm's formation accuracy depends on the radius R, we repeated the previous experiment, but now with R= 50 meters. Figure 24 shows the obtained results.

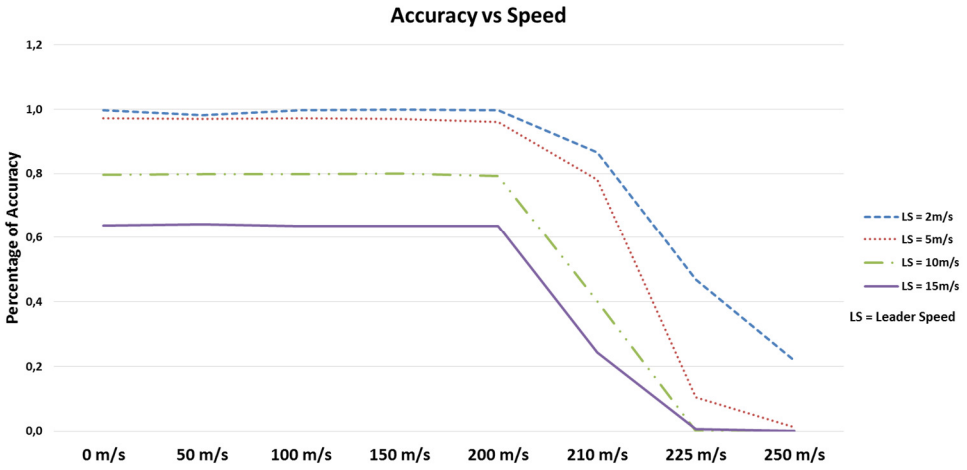

Figure 24 – Same experiment as of Section 5.3, but now with swarm radius equal to 50 meters.

When comparing this graph of Figure 19 with the one of Figure 23, it is clear that the overall percentage of formation accuracy increases for greater speeds. On the other hand, increasing the swarm's radius R does not result in an increase of the network latency tolerance. Similarly, the significant decrease in the swarm formation's accuracy that was observed in Section 5.3 did not change with the wider swarm radius. Nonetheless, overall, when the swarm radius is increased, all network latencies resulted in higher accuracy levels.

This improvement is due to the concept of formation accuracy adopted here. As previously noted, when the swarm radius increases make easier for all slaves reach their proximity to their desirable position between $\boldsymbol{r}$ - *10%* and $\boldsymbol{r}$ + *10%*. Hence, the precision (margin +/- 10% used) is also an important variable to be taken into consideration when assessing aerial swarm formation characteristics and setting parameters.

This evaluation does not intend to explore all possible factors that can affect the swarm formation accuracy. However, we identified the three factors that seem to be more influential on the swarm's behavior, at least in a simulation environment. As illustrated in chapter 3, the swarm formation's accuracy can be tuned by balancing three main factors: Leader Speed, Network Latency and Desired Accuracy (the swarm's radious).

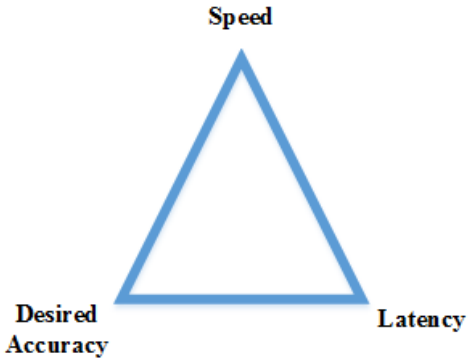

Figure 25 – Main factors that jointly influence the swarm formation accuracy: Leader Speed, Network Latency and Accuracy.

5.4
Message-Loss Analysis

The SDDL middleware, includes services and protocols for handling message loss over the mobile Internet link. For example, it provides a persistent service that keeps reattempting the delivery of non-delivered messages until a certain limit is reached. Once the threshold is reached, the ClientLib layer the application layer about the outcome. As for our analysis it is important to emulate a percentage of non-delivered messages and analyze the impact of this to the swarm formation and maintenance, this message-loss handling feature of SDDL was turned off, and all previous experiments were repeated. In other words, when we were running another set of simulations, all SDDL message delivery controls were manually disabled. Such scenarios produced the expected behaviors as a consequence of each type of failed message delivery. The main behaviors of the leader and the slaves – all of which were detected in the simulations- are summarized as follows:

- **Recruitment message from the Leader:** Without this initial message, no free UAV will present itself as a candidate for swarm formation, making the swarm unfeasible, thus terminating the process in the recruitment phase. After a period of $\boldsymbol{\alpha}$ seconds without receiving any answers from the slaves, the Leader starts a new recruitment.

- **Candidacy message from each invited Slave:** In the recruitment of a swarm of $\boldsymbol{n}$ UAV, 1 to n messages of candidacy messages from the invited slaves could be lost. In this case, after a period of α seconds without the

arrival of sufficient number of answers, the Leader will restart the recruitment procedure.

- **Recruitment confirmation message from the Leader to selected Slaves:** While confirming the recruitment of slaves for a swarm of $\boldsymbol{n}$ UAVs, 1 to n confirmation messages could be lost. In this case, two different outcomes can happen: (1) after β seconds have lapsed, all slaves—whether or not selected by the leader—that did not receive the confirmation message timeout and return to their previous tasks; or (2) the leader starts a new selection after a period of $random(\alpha) + \beta$ seconds, due to the timeout expecting slaves.

- **Keep-alive messages from slaves to their Leader:** The outcome of this scenario is similar to the one described in the previous case. If a leader timeouts after $\boldsymbol{\mu}$ seconds without receiving a heartbeat from some slaves, it releases all its slaves and starts a new recruitment.

- **Location update from leader to its slaves:** After a period of φ seconds has elapsed without receiving any messages from the leader, a slave returns to its previous task.

6.
Related Work

In this section, we summarize and compare four relevant related works. The first two studies are similar to our work, and address the coordination of swarms of quadcopters in order to perform coordinated swarm in movement. Two further works are discussed because they focus on swarms of UAVs performing autonomous monitoring tasks with their own communication approach.

6.1
UAV Collaborative Swarm

Vijay Kumar and his group at the GRASP Laboratory have a long-standing tradition of working with UAV swarms [29] [30] [31] [32]. Their work focuses on swarms of UAVs flying together in different formation shapes. The authors have achieved significant swarm movement accuracy, as confirmed by several of their tests. A video with the performance of the swarm can be found on YouTube at www.youtube.com/watch?v=YQIMGV5vtd4. Their swarms are capable of performing several flight maneuvers together, and thus can be deployed for different tasks. An example of UAV swarms successfully performing tasks together, although in an indoor, controlled environment, can be found in [29].

These achievements are accomplished by employing a set of identical UAVs, aided by a central controller and client-server communication between the UAVs and the controller. The UAVs are built with dedicated hardware and communication employ conventional client-server paradigm using IEEE 802.14.5 radio links [32]. Once the UAVs are turned on, each one establishes a link with a central processing node (e.g. the controller) that uses another IEEE 802.14.5 link to communicate with all other UAVs in the swarm.

The environment in which those UAVs fly is a special room equipped with motion capture cameras. Each UAV has a unique 3D marker that can be captured with these cameras, making the room itself work as an internal, precise location system for each UAV. This information feeds a central processing node that is connected to the location system and all UAVs. The entire process is managed by a mathematical model in Mathlab, which simultaneously controls all the UAVs with precision and robustness.

Despite some apparent similarities, this approach differs from our work in several aspects. Probably the biggest difference between the two stems from the location service approach. Vijay Kumar et al. have developed a precise indoor solution using cameras, markers and a previously prepared environment (e.g. walls and ceiling in specific color), while our approach focuses on outdoor, unprepared areas. Hence, while swarms flying in Kumar's setting can reach a precision of a few in centimeters, the GPS-based approach used in our work will have precision in the scale of meters. However, for most practical purposes, civilian and open GPS might be the only positioning service available. On the other hand, our approach does not require previous preparation of the environment, use of cameras and marking, and avoids problems related to the camera's line-of-sights. Moreover, the radio link used in Kumar et al. is, IEEE 802.14.5 that is used for Wireless PAN and has a range of 10-75 meters, which is clearly unsuitable for wide-area applications. On the other hand, our approach advocates the use of mobile Internet (3G/4G) networks as the means for communication with UAVs, even if it brings the problem of a higher communication latency.

Another relevant difference stems from the swarm control paradigm. More specifically, while Vijay Kumar et al. use a central controller to process tasks and movements of all UAVs, the paradigm adopted here aims at a distributed control through all UAVs, backed up by a scalable communication middleware with efficient groupcast. Thus, the potential scalability of the coordination model developed in the present work appears to be a significant advantage for practical deployments.

6.2
Flying Machine Arena

Rafaello d´Andrea et al. from the ETH-Z group have long been analyzing UAVs [33][34][35][36]. Of particular interest is their work entitled The Flight Machine Arena [37], which also has some similarities to the work by Vijay et al. [38]. The authors describe their use of prepared rooms with motion capture cameras to provide a precise location services to centrally control the UAVs. This work, which revisits previous studies from Vijay et al[38] in this field, also introduces more advanced approaches to flight control of individual UAVs, enabling quadcopters that have lost one of their propellers to land without problems. For the communication this work also uses IEEE 802.14.5 .

The similarities and differences between our work and the work of Rafaello d´Andrea et al. [39] are essentially the same as those to the work of Vijay et al.

6.3
Human Immune System-Based Algorithm

Weng *et al*. designed and implemented a distributed algorithm [40] inspired on the human immune system [41], which bears some similarities to the surveillance scenario envisaged in the present work. Their model bases on fixed-wing UAVs, locally processing their tasks and a fully distributed ad hoc coordination paradigm. Initially, the UAVs of a swarm fly individually along some routes within a delimited area of surveillance. Each UAV has radars capable of finding a Point Of Interest, such as an intruder, for example. As soon as one UAV—V_1, for instance—detects a moving intruder, it requests assistence from other UAVs within its radio range and starts following the target. All UAVs that receive the assistence request from V_1 respond by changing their routes in the direction of V_1, thus also moving towards the intruder, and with the goal to form a swarm around the leader. In their approach, all UAVs can assume the role of V_1, *i.e.*, any UAV that detects an intruder starts chasing it and sends messages to nearby UAVs. To avoid having all UAVs chasing a single intruder, and in order to have other, available UAVs to patrol the perimeter, Weng et al. devised a technique (based on the human immune system) that uses a predetermined threshold to limit the number of UAVs deployed in a single task [40].

Unfortunately, in their work [40], the authors do not explain how the UAVs obtain their location. However, a UAV is capable of sharing its position with others, in order to request assistence. As it is also capable of randomly patrolling a pre-determined wide-area region, it is likely that the UAVs of this work also relied on an outdoor, GPS-based localization method.

In this related work, as well as in our approach, all helper UAVs keep receiving Location Updates from the leader V_1, and proceed towards their target position relative to the leader, in the swarm. Any change in the leader's flight path, while following the intruder's movements, will generate Location Updates that are instantly shared with the "helping" UAVs. The work by Weng *et al*. is similar to ours in that it proposes a scalable system that is fully dynamic with respect to the number of UAV. It ensures that the swarm size is flexible and can accommodate potential arrival or leave of aircraft. Unlike our work, the approach developed by Weng *et al*., uses a networking swarm rather than a communicating swarm, according to Çelikkanat and Şahin [4], where the inter-UAV communication range is limited. Therefore, the behavior of individual UAVs cannot be altered based in others UAVs intra swarm, or even a leader, in order to enhance their

coverage, UAVs following a leader does not spread their presence over an area. Finally, it is not possible to obtain data on the overall system state, errors or achievements.

6.4
Perimeter Patrol Algorithm

Kingston et al. [42] proposed an approach that can be implemented to patrol a perimeter (e.g., a frontier) of a region using a dynamic team of UAVs, but which do not move in a rigid swarm flight formation. Their strategy involves a dynamic group of n UAVs hovering along a perimeter, where each UAV is responsible for patrolling an area equal to Length of a perimeter divided by number of UAVs. The authors also created a decentralized algorithm for coordinating the UAV movements, allowing the system to control an unknown and dynamic number of UAVs working together adapting itself in order to maintain perimeter vigilance. The communication adopted in this approach is based on an ad hoc paradigm, allowing information to be exchanged when two UAVs are approaching each other from opposite directions. It also uses geo-localization mechanisms, as does our work.

Similar to the work of Weng et al., this system offers to handle dynamic number of UAVs, instead to the approach proposed in the present work. However, it also uses an ad hoc communication paradigm without a central system for overall control or monitoring. Here, we find also some notable similarities with the work of Weng et al. [40], although this approach limits the UAV operations to a pre-determined perimeter, unlike our work, where the UAVs in Patrol mode (and in the Leader role) may exhibit any movement pattern.

7.
Conclusion

Our work presents an approach to use a hardware architecture and a message-loss tolerant protocol to create swarm of UAVs aiming surveillance above urban areas using mobile network as communication channel. This approach bases on simple integration hardware, an off-the-shelf Smartphone technology and it is telecom carrier independent. Consequently, the proposed approach is simple to reproduce and can be improved in subsequent studies.

We present a distributed Coordination Algorithm and its respective tests by a set of simulations using emulated UAVs. The algorithm was successfully tested for correctness. We also present simulations that identifies speed relations between Leaders and Slaves in a swarm, tendering an efficient relation between them.

Finally, latency variation experiments have revealed an important tradeoff between network latency, UAV speeds and formation distances between UAVs on the swarm formation.

7.1
Future Work

This work does not close itself, further works could follow from that, and this section presents some of them. First of them could be to build prototypes and conduct live tests based on the achievements presented here, in order to revalidate the overall correctness and latencies tests using real mobile networks instead of simulations.

Secondly, regarding the swarm formation itself, which forms a circle around the Leader, such pattern could be modular and work with other patterns. Delta patterns could be useful while using fixed wings UAVs instead quadcopters in any task (due their flight characteristics) and straight lines formation would be useful to provide surveillance above convoys on a road.

Following, our Coordination Algorithm could be enhance its usage by incorporating some of the capabilities seen in the related work, such as a dynamic number of Slaves UAVs.

Finally, a hybrid approach, mixing the Internet-based communication model and an ad hoc local communication system, could be investigated. Such approach could combine the advantages of both short-range networks and mobile-cloud usage. It could provide lower latencies with short-range radios and long-range message exchange by using the Internet.

8.
Bibliography

[1] R. Austin, *Unmanned Aircraft Systems*, First Edit. Chichester, UK: John Wiley & Sons, Ltd, 2010.

[2] G. Prencipe, "The Effect of Synchronicity on the Behavior of Autonomous Mobile Robots," *Theory Comput. Syst.*, vol. 38, no. 5, pp. 539–558, Jun. 2005.

[3] G. Prencipe, "Corda: Distributed Coordination of a Set of Autonomous Mobile Robots," 2001.

[4] H. Çelikkanat and E. Şahin, "Steering self-organized robot flocks through externally guided individuals," *Neural Comput. Appl.*, vol. 19, no. 6, pp. 849–865, Mar. 2010.

[5] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Auton. Robots*, vol. 35, no. 4, pp. 287–300, Jul. 2013.

[6] V. Kumar and N. Michael, "Opportunities and challenges with autonomous micro aerial vehicles," *Int. J. Rob. Res.*, vol. 31, no. 11, pp. 1279–1291, Aug. 2012.

[7] S. Bhattacharya, R. Ghrist, and V. Kumar, "Multi-robot Coverage and Exploration on Riemannian Manifolds with Boundary," *Int. J. Rob. Res.*, 2013.

[8] J. Fink, A. Ribeiro, and V. Kumar, "Robust Control of Mobility and Communications in Autonomous Robot Teams," *IEEE Access*, vol. 1, pp. 290–309, 2013.

[9] Wikipedia, "Unmanned aerial vehicle," 2014. [Online]. Available: http://en.wikipedia.org/wiki/Unmanned_aerial_vehicle.

[10] L. David, R. Vasconcelos, L. Alves, R. André, and M. Endler, "A DDS-based middleware for scalable tracking, communication and collaboration of mobile nodes," *J. Internet Serv. Appl.*, vol. 4, no. 1, p. 16, 2013.

[11] E. M. Arkin, M. A. Bender, S. P. Fekete, J. S. B. Mitchell, and M. Skutella, “The Freeze-Tag Problem: How to Wake Up a Swarm of Robots,” *Algorithmica*, vol. 46, no. 2, pp. 193–221, Jul. 2006.

[12] Y. U. Cao, A. S. Fukunaga, and A. Kahng, “Cooperative Mobile Robotics: Antecedents and Directions,” *Auton. Robots*, vol. 4, no. 1, pp. 7–27, Mar. 1997.

[13] M. Erdmann and T. Lozano-Pérez, “On multiple moving objects,” *Algorithmica*, vol. 2, no. 1–4, pp. 477–521, Nov. 1987.

[14] Y. Kuniyoshi, N. Kita, S. Rougeaux, S. Sakane, M. Ishii, and M. Kakikua, “Cooperation by observation: the framework and basic task patterns,” in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 767–774.

[15] L. E. Parker, “Designing control laws for cooperative agent teams,” in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pp. 582–587.

[16] M. Penttonen and E. M. Schmidt, Eds., *Algorithm Theory — SWAT 2002*, vol. 2368. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002.

[17] A. Pal, A. D. Kshemkalyani, R. Kumar, and A. Gupta, Eds., *Distributed Computing – IWDC 2005*, vol. 3741. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[18] A. Restivo, S. R. Della Rocca, and L. Roversi, *Theoretical Computer Science*, vol. 2202. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001.

[19] I. Suzuki and M. Yamashita, “Distributed Anonymous Mobile Robots: Formation of Geometric Patterns,” *SIAM J. Comput.*, vol. 28, no. 4, pp. 1347–1363, Jan. 1999.

[20] M. YAMASHITA and I. SUZUKI, “AGREEMENT ON A COMMON X-Y COORDINATE SYSTEM BY A GROUP OF MOBILE ROBOTS.” Dasgsthul Seminar on Modeling and Planning for Sensor-Based Intelligent Robots., 1996.

[21] D. Thakur, M. Likhachev, J. Keller, V. Kumar, V. Dobrokhodov, K. Jones, J. Wurz, and I. Kaminer, “Planning for opportunistic surveillance with multiple

robots," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 5750–5757.

[22] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006.

[23] A. (MIT) Richards, J. (MIT) Bellingham, M. (MIT) Tillerson, and J. (MIT) How, "Coordination and Control of Multiple UAVs," *AIAA Guid. Navig. Control Conf. Monterey, CA*, 2003.

[24] S. Sense, "Smartphone definition," *Wikipedia*, 2015. [Online]. Available: http://en.wikipedia.org/wiki/Smartphone.

[25] M. Endler, G. Baptista, L. D. Silva, R. Vasconcelos, M. Malcher, V. Pantoja, V. Pinheiro, and J. Viterbo, "ContextNet: Context Reasoning and Sharing Middleware for Large-scale Pervasive Collaboration and Social Networking," in *Proceedings of the Workshop on Posters and Demos Track - PDT '11*, 2011, pp. 1–2.

[26] A. Buchmann and B. Koldehofe, "Complex Event Processing," *it - Inf. Technol.*, vol. 51, no. 5, Jan. 2009.

[27] E. is an open source event series analysis and event correlation engine (CEP)., "EsperTech - Complex Event Processing," 2015. [Online]. Available: http://www.espertech.com/products/esper.php. [Accessed: 15-Jun-2015].

[28] R. O. Vasconcelos, L. David, and M. Endler, "Scalable Data Distribution Layer (SDDL)," 2012. .

[29] A. (University of P. Kushleyev, D. (University of P. Mellinger, and V. (University of P. Kumar, "Towards a Swarm of Agile Micro Quadrotors," *Robotics: Science and Systems*, 2012. [Online]. Available: https://fling.seas.upenn.edu/~dmel/wiki/index.php?n=Main.Publications.

[30] Q. Jiang, D. Mellinger, C. Kappeyne, and V. Kumar, "Analysis and Synthesis of Multi-Rotor Aerial Vehicles," in *Proceedings of the International Design Engineering Technical Conference*, 2011.

[31] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar, "Design, Modeling, Estimation and Control for Aerial Grasping and Manipulation," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[32] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction with Quadrotor Teams," *Auton. Robots*, 2012.

[33] F. Augugliaro, A. Schoellig, and R. D'Andrea, "Dance of the Flying Machines: Methods for Designing and Executing an Aerial Dance Choreography," *Robot. Autom. Mag. IEEE*, pp. 96–104, 2013.

[34] R. Ritz and R. D'Andrea, "Carrying a flexible payload with multiple flying vehicles," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, pp. 3465–3471.

[35] F. Augugliaro and R. D'Andrea, "Admittance control for physical human-quadrocopter interaction," in *Control Conference (ECC), 2013 European*, 2013, pp. 1805–1810.

[36] M. Hehn and R. D'Andrea, "An iterative learning scheme for high performance, periodic quadrocopter trajectories," in *Control Conference (ECC), 2013 European*, 2013, pp. 1799–1804.

[37] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The Flying Machine Arena," *Mechatronics*, 2014.

[38] F. Augugliaro, A. Mirjan, F. Gramazio, M. Kohler, and R. D'Andrea, "Building tensile structures with flying machines," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, pp. 3487–3492.

[39] A. Schoellig, F. Augugliaro, and R. D'Andrea, "A platform for dance performances with multiple quadrocopters," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)-Workshop on Robots and Musical Expressions*, 2010, pp. 1–8.

[40] L. Weng, Q. Liu, M. Xia, and Y. D. Song, “Immune network-based swarm intelligence and its application to unmanned aerial vehicle (UAV) swarm coordination,” *Neurocomputing*, no. 0, p. -, 2013.

[41] V. Roberge, M. Tarbouchi, and G. Labonte, “Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning,” *IEEE Trans. Ind. Informatics*, vol. 9, no. 1, pp. 132–141, Feb. 2013.

[42] D. Kingston, R. W. Beard, and R. S. Holt, “Decentralized Perimeter Surveillance Using a Team of UAVs,” *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1394–1404, Dec. 2008.