Opportunistic data collection and routing in segmented wireless sensor networks

Juliette Garcia Alviz

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *07 Décembre 2020* **par :**

JULIETTE GARCIA

Opportunistic Data Collection and Routing in Segmented Wireless Sensor Networks

---

JURY

NATHALIE MITTON	Inria Lille-Nord Europe	Rapporteure
FABRICE VALOIS	Inria Agora / INSA Lyon	Rapporteur
KHALED BOUSSETTA	Université Paris 13	Examinateur
THIERRY GAYRAUD	Université Paul Sabatier	Examinateur
OANA IOVA	INRIA / INSA Lyon	Examinatrice
THIERRY VAL	IUT Blagnac	Examinateur
ALAIN PIROVANO	ENAC Toulouse	Directeur de thèse
MICKAËL ROYER	ENAC Toulouse	Co-directeur de thèse

École doctorale et spécialité :
EDSYS : Informatique 4200018
Unité de Recherche :
École Nationale de l'Aviation Civile (ENAC), Toulouse
Directeur(s) de Thèse :
Alain Pirovano et *Mickaël Royer*
Rapporteurs :
Nathalie Mitton et *Fabrice Valois*

Acknowledgments

My deepest gratitude to God for giving me this great opportunity, along with life and health to achieve the success on it. Thanks to His support, this thesis became a reality. For God is all the glory.

I also thank my advisors, Alain Pirovano and Mickaël Royer, for all the time they dedicated to helping me with the execution of my thesis, and for all their useful directions.

Special thanks to Professors Fabrice Gamboa, Thierry Klein and François Bachoc for all the timely help you gave me at crucial moments throughout the thesis and my time in France. Thanks for your advices and also for your determination to take action and provide support. For you, my best wishes.

I also express my gratitude to my family and to my husband's family, for all your constant support. What a privilege I have to have meet such wonderful and lovely people. Your patience, knowledge and love is priceless.

Huge thanks to my husband, José, my life partner. I'm grateful for your unconditional support, encouragement, care, and listening along this journey. You are the most intelligent person I know. Wish you the best in everything as you deserve nothing less.

Finally, I want to thank all the people who contributed in some way and helped me complete this research successfully.

Résumé

La surveillance régulière des opérations dans les aires de manoeuvre (voies de circulation et pistes) et aires de stationnement d'un aéroport est une tâche cruciale pour son fonctionnement. Les stratégies utilisées à cette fin visent à permettre la mesure des variables environnementales, l'identification des débris (FOD) et l'enregistrement des statistiques d'utilisation de diverses sections de la surface. Selon un groupe de gestionnaires et contrôleurs d'aéroport interrogés, cette surveillance est un privilège des grands aéroports en raison des coûts élevés d'acquisition, d'installation et de maintenance des technologies existantes. Les moyens et petits aéroports se limitent généralement à la surveillance de quelques variables environnementales et des FOD effectuée visuellement par l'homme. Cette dernière activité impose l'arrêt du fonctionnement des pistes pendant l'inspection. Dans cette thèse, nous proposons une solution alternative basée sur les réseaux de capteurs sans fil (WSN) qui, contrairement aux autres méthodes, combinent les propriétés de faible coût d'installation et maintenance, de déploiement rapide, d'évolutivité tout en permettant d'effectuer des mesures sans interférer avec le fonctionnement de l'aéroport.

En raison de la superficie d'un aéroport et de la difficulté de placer des capteurs sur des zones de transit, le WSN se composerait d'une collection de sous-réseaux isolés les uns des autres et du puits. Pour gérer cette segmentation, notre proposition s'appuie sur l'utilisation opportuniste des véhicules circulants dans l'aéroport considérés alors comme un type spécial de nœud appelé Mobile Ubiquitous LAN Extension (MULE) chargé de collecter les données des sous-réseaux le long de son trajet et de les transférer vers le puits. L'une des exigences pour le déploiement d'un nouveau système dans un aéroport est qu'il cause peu ou pas d'interruption des opérations régulières. C'est pourquoi l'utilisation d'une approche opportuniste basé sur des MULE est privilégiée dans cette thèse. Par opportuniste, nous nous référons au fait que le rôle de MULE est joué par certains des véhicules déjà existants dans un aéroport et effectuant leurs déplacements normaux. Et certains nœuds des sous-réseaux exploiteront tout moment de contact avec eux pour leur transmettre les données à transférer ensuite au puits. Une caractéristique des MULEs dans notre application est qu'elles ont des trajectoires structurées (suivant les voies de circulation dans l'aéroport), en ayant éventuellement un contact avec l'ensemble des nœuds situés le long de leur trajet (appelés sous-puits). Ceci implique la nécessité de définir une stratégie de routage dans chaque sous-réseau, capable d'acheminer les données collectées des nœuds vers les sous-puits et de répartir les paquets de données entre eux afin que le temps en contact avec la MULE soit utilisé le plus efficacement possible. Dans cette thèse, nous proposons un protocole de routage remplissant ces fonctions.

Le protocole proposé est nommé ACME (ACO-based routing protocol for MULE-assisted WSNs). Il est basé sur la technique d'Optimisation par Colonies de Fourmis. ACME permet d'assigner des nœuds à des sous-puits puis de définir les chemins entre eux, en tenant compte de la minimisation de la somme des longueurs de ces chemins, de l'équilibrage de la quantité de paquets stockés par les sous-puits et du nombre total de retransmissions. Le problème est défini comme une tâche d'optimisation multi-objectif qui est résolue de manière distribuée sur la base des actions des nœuds dans un schéma collaboratif. Nous avons développé un environnement de simulation et effectué des campagnes de calculs dans OMNeT++ qui montrent les avantages de notre protocole en termes de performances et sa capacité à s'adapter à une grande variété de topologies de réseaux.

Abstract

The regular monitoring of operations in both movement areas (taxiways and runways) and non-movement areas (aprons and aircraft parking spots) of an airport, is a critical task for its functioning. The set of strategies used for this purpose include the measurement of environmental variables, the identification of foreign object debris (FOD), and the record of statistics of usage for diverse sections of the surface. According to a group of airport managers and controllers interviewed by us, the wide monitoring of most of these variables is a privilege of big airports due to the high acquisition, installation and maintenance costs of most common technologies. Due to this limitation, smaller airports often limit themselves to the monitoring of environmental variables at some few spatial points and the tracking of FOD performed by humans. This last activity requires stopping the functioning of the runways while the inspection is conducted. In this thesis, we propose an alternative solution based on Wireless Sensor Network (WSN) which, unlike the other methods/technologies, combines the desirable properties of low installation and maintenance cost, scalability and ability to perform measurements without interfering with the regular functioning of the airport.

Due to the large extension of an airport and the difficulty of placing sensors over transit areas, the WSN might result segmented into a collection of subnetworks isolated from each other and from the sink. To overcome this problem, our proposal relies on a special type of node called Mobile Ubiquitous LAN Extension (MULE), able to move over the airport surface, gather data from the subnetworks along its way and eventually transfer it to the sink. One of the main demands for the deployment of any new system in an airport is that it must have little or no interference with the regular operations. This is why the use of an opportunistic approach for the transfer of data from the subnetworks to the MULE is favored in this thesis. By opportunistic we mean that the role of MULE will be played by some of the typical vehicles already existing in an airport doing their normal displacements, and the subnetworks will exploit any moment of contact with them to forward data to the sink. A particular characteristic of the MULEs in our application is that they move along predefined structured trajectories (given by the layout of the airport), having eventual contact with the set of nodes located by the side of the road (so-called subsinks). This implies the need for a data routing strategy to be used within each subnetwork, able to lead the collected data from the sensor nodes to the subsinks and distribute the data packets among them so that the time in contact with the MULE is used as efficiently as possible. In this thesis, we propose a routing protocol which undertakes this task.

Our proposed protocol is named ACME, standing for ACO-based routing protocol for MULE-assisted WSNs. It is founded on the well known Ant Colony Optimization (ACO) technique. The main advantage of ACO is its natural fit to the decentralized nature of WSN, which allows it to perform distributed optimizations (based on local interactions) leading to remarkable overall network performance. ACME is able to assign sensor nodes to subsinks and generate the corresponding multi-hop paths while accounting for the minimization of the total path length, the total subsink imbalance and the total number of retransmissions. The problem is defined as a multi-objective optimization task which is resolved in a distributed manner based on actions of the sensor nodes acting in a collaborative scheme. We conduct a set of computational experiments in the discrete event simulator OMNeT++ which shows the advantages of our protocol in terms of performance and its ability to adapt to a variety of network topologies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sensor-based monitoring has been gaining significant attention over the last two decades, both in academic and practical dimensions. Survey studies like [1], [2] and [3] show a sharp positive trend in the number of research publications per year, while the pool of specialized ventures grows fast in this domain. The range of applications attended so far is remarkably broad, including those in military operation, industrial safety, precision agriculture, environmental monitoring, health-care, home automation, among several others (see e.g., [2, 4, 5]). This thesis is framed on the implementation of sensor networks for monitoring of environmental, safety-oriented and operational data at airports.

1.1 Background and motivation

This work is inspired by the process of Airport Surface Area Surveillance (ASAS), which encompasses the set of strategies and techniques implemented to monitor and control operations in the airports, seeking for safety and sustainability. Those include the continuous supervision of movement areas[1] for the timely identification of foreign object debris (FOD), non-movement areas[2] for the register of operational statistics (e.g., usage of different parking spots) for managerial purposes, and the record of environmental data (e.g., noise, concentration of air pollutants, temperature) for mid- and long-term intervention and control. All these actions seek to mitigate risk of accidents during the operation of aircraft on the airport, protect the health and comfort of people living in its surroundings, and ensure its environmental sustainability. The large-scale adoption of these actions worldwide helps to gradually reduce the negative effects of aircraft operations, making it a long-term reliable solution for long-haul transportation.

[1] Movement areas of an airport include the taxiways and runways.
[2] Non-movement areas of an airport include the aprons and aircraft parking spots.

At the early stages of this thesis we conducted a set of interviews oriented to identify the specific needs of the airports in terms of data monitoring and determine the current practices in this activity. The interviewees were an airport manager and two traffic controllers, who pointed out the following interests:

- ***Environmental data:*** helps to determine if the airport is compliant with the current regulations and formulate strategies for further improvement. Environmental data collected at the airports is often also exploited by local and regional governments concerned by sustainable development of populated areas, research groups and weather information providers. Indeed, the weather conditions reported by many apps and websites are based on the reports of the weather station placed at the closest airport.
  - Targets: temperature, pollution, wind, noise, water/snow height, cloud cover.
- ***Safety-oriented data:*** contributes to keep the risk of accidents in airports as low as possible. While environmental data is primarily utilized for mid- and long-term decision making, safety-oriented data often implies some threat to the aircraft operating in the airport and thus, merits immediate reaction.
  - Targets: presence of foreign objects, state of runway safety lights, water/snow height around taxiways and runways.
- ***Operational data:*** pointed out by the interviewees as of great potential impact on the efficient management of airport physical resources, but currently not being systematically collected on them. Statistics on the use of physical resources could help airport managers to prevent overloads, reduce failures, systematically increase the efficiency of the operations and constitute effective maintenance programs.
  - Targets: record of taxiway, apron and parking spot visits.

According to our interviewees, the integral monitoring of environmental and safety-oriented data is nowadays a privilege of the biggest airports due to the high acquisition, installation and maintenance costs of the required technologies. Thus, most mid- and small-size airports limit themselves to human-based foreign object debris[3] (FOD) detection and sometimes the monitoring of a limited number of environmental variables. This thesis seeks to bridge this gap by proposing an alternative low-cost monitoring solution suitable for the above listed interests.

[3]Foreign object debris refers to any entity located somewhere over the movement area of the airport, which is not supposed to be there, and whose presence at the wrong place puts the aircraft at risk (e.g., pieces fallen from moving aircraft and wildlife).

1.2 Monitoring solutions currently in use

Human-based FOD detection: consists of regular inspections of runways and taxiways performed by ground personnel to ensure that these areas are free of FOD. The task typically involves 5 to 10 workers [6] which cross the runway from end to end, some by walk and others by car (see Figure 1.1a). The International Civil Aviation Organization (ICAO) recommends to make at least four inspections a day [7], which has been confirmed as the standard by some airport managers [8]. A major drawback of this method is that it is time consuming and no take-offs or landings can be conducted in parallel [9], which explains why the frequency is usually kept low. On top of that, many studies (e.g., [10], [9] and [6]) have pointed out the vulnerability of the approach to human failures. Even fully-conscious, reasoning humans are prone to errors due to fatigue, stress, limited visibility, among other factors that can prevent the opportune identification of a piece of FOD. Despite this, human-based inspection is the technique adopted by most airports worldwide due to lack of resources to access more sophisticated solutions [6].

Mounted-on-tower FOD detection: consists of automated system of sensors mounted on fixed towers by the side of the runways, allowing to detect FOD (see Figures 1.1b to 1.1d). Multiple solutions of this type have been proposed, but up to now only three of them have been approved by the Federal Aviation Administration (FAA) and also the only ones installed in airports around the world [6]: ***Tarsier*** by QinetiQ [11, 12], ***FODFinder-XF*** by Trex [13, 14] and ***iFerret*** by Stratech [15]. Tarsier and FODFinder use radars, while iFerret uses high resolution cameras instead. The number of towers typically varies between 2 and 6 per runway. Normally, these systems require a wired installation for power supply and a communication mechanism (either wired or wireless) to transmit the information to the control center. All three systems continuously monitor the area, making it superior to human inspection but little accessible for some airports due to its high overall cost (acquisition, installation and maintenance) [6].

Mounted-on-truck FOD detection: instead of using fixed towers like in the previous solution, this one uses vehicles that carry on the sensors along the runway during regular inspections (see Figure 1.1e). Up to now, the commercial implementation of this solution is offered by Trex in the ***FODFinder-XM*** [13, 14], which uses radar technology as its analogous fixed version FODFinder-XF described above. Although the inspections are conducted by crossing the runway from end to end while checking for FOD along the way, this solution is considered much more reliable and stable than human-based inspection since it removes the errors due to human factors from the process. The mounted-on-truck solution is more economical than the mounted-on-tower one since the former only requires one radar unit while the later often needs at least four; however, truck rides along the runway cannot

(a) Human-based

(b) MoTower Tarsier

(c) MoTower FODFinder-XF

(d) MoTower iFerret

(e) MoTruck FODFinder-XM

(f) Weather station

(g) Wired sensor network FODetect

Figure 1.1: Monitoring solutions currently in use. Pictures (a), (b), (c), (d), (e) and (g) retrieved from [17], [12], [14], [15], [14] and [17], respectively.

be performed very often since they require landings and takeoffs to be stopped, which is a downside of the truck-based solution in front of the tower-based one. Each single radar unit, however, is expensive enough to put this solution out of the scope of many airports [6, 16].

Weather Station: contrarily to all the above mentioned systems which provide monitoring of safety-oriented data, this one addresses environmental data. The system involves a physical facility equipped with a number of sensors allowing to gather and report meteorological observations (see Figure 1.1f). Like mounted-on-tower FOD detection systems, a weather station needs a wired installation of power supply and some communication mechanism (either wired or wireless) to report data to the control center. The installation and maintenance of this system is prohibitively expensive for many airports [18]. Part of the expense can be amortized by trading the collected data with interested governmental, scientific and commercial entities; however, even with this possibility on the table, the cost is sometimes too high and small airports opt for attaching themselves to stations placed elsewhere in the city [18].

Wired sensor networks: this solution employs fixed-location sensors like the tower-based FOD detection systems, but relies on a larger number of sensors endowed with shorter sensing range (see Figure 1.1g). The cost of each sensor unit is thus naturally cheaper in this system [19], but a greater number of nodes (at least a few tens) is often needed. The system requires an extensive wired installation providing power supply and communication means to every sensor node. So far, this type of system has been implemented in airports by XSight under the name of ***FODetect*** [17]. In this version, the nodes are mounted on the runway lighting fixtures to resolve the need for power supply. Yet, communication often poses a major issue since network cables are not expected to be already available for connection at the lighting spots. Thus, the implementation might require to either: (i) throw a new wire through the pipe used for power supply, which might not be possible if the pipe is already at maximum fill or has prohibitive bends; or (ii) install a new pipe, which might be considerably expensive due to the proportion of the works to do at the airport. The overall cost of this system is still out of the scope of most mid- and small-size airports [6].

Synthesis: five different types of solutions have been implemented so far; four of them for safety-oriented monitoring and one for environmental monitoring. To the best of our knowledge, no solution has been proposed to address operational monitoring. Several observations can be made about the attributes and scope of the available alternatives:

- The human-based FOD detection method seems to be the only solution broadly cost-affordable by airports of any size. Unfortunately, this approach is inefficient, considerably unreliable, and leaves the system vulnerable for long periods. All the other FOD detection solutions are superior in these three aspects, but also much more expensive.
- The MoTower and MoTruck FOD detection solutions rely on a few units of costly specialized equipment, which makes them little fault tolerant. Wired sensor networks are superior in this aspect since backup units could be stored and easily put into operation in case of failure;

- Weather stations are the only environment-oriented solution so far. These are composed of multiple specialized mechanisms which lead to high acquisition and maintenance costs. They present the same issue of lack of resilience as the MoTower and MoTruck FOD detection solutions since they rely on a single set of specialized sensors which are not cheap to replace in case of failure;
- Most technology-based solutions are difficult to scale and take to some places of the airport due to the need for power and/or communication wire installation, which considerably limits their scope;
- Based on the previous observations, while some of the available solutions certainly provide significant value, there is still a clear place to improve in terms of cost, resiliency, versatility and comprehensiveness.

1.3 Proposed WSN-based monitoring solution

The system devised in this thesis seeks to integrally fulfill most of the gaps found in the solutions currently in use at the airports. Our proposal relies on a wireless sensor network (WSN) with the following characteristics:

- Fully automated monitoring, which prevents any operational risk linked to human error;
- Continuous non-intrusive sensing without the need to stop the regular functioning of the airport, which ensures safety without compromising efficiency;
- Induced resilience by means of a relatively large number of small-size, moderate-cost sensor nodes which are easy and fast to replace in case of failure;
- High versatility and easy deployment supported on battery-powered sensor nodes which do not require any wire installation.

The nodes are expected to be deployed over the airport covering the main areas of interest for monitoring. Every node must be provided with ZigBee technology for communication of non-critical data with nearby peers, and only those gathering safety-oriented data must also be provided with LoRa technology for the exclusive transmission of critical data to the sink (i.e., the control center). We provide more details on the selection of these technologies in Section 1.3.1. Due to the large size of the airports and the limited communication range of the sensor nodes for non-critical information, it could be expensive or impractical to establish a connected WSN[4] covering the entire area. Thus, we rather consider a segmented WSN constituted by multiple isolated subnetworks strategically distributed over the airport. Diverse approaches can be followed in order to keep possible the communication between

[4]By connected network, we refer to one where any node is reachable by any other node either by direct or multi-hop communication. We call the contrary case, where some nodes are not reachable by others by the name of segmented network.

the subnetworks and the sink. In this thesis we opt for the opportunistic use of the airport service vehicles in regular duty as relay nodes. This type of mobile emissary node is better known in the telecommunications field as Mobile Ubiquitous LAN Extension (MULE). By opportunistic we imply that the service vehicles will keep their normal schedule and trajectory, and the subnetworks will need to exploit any moment of contact with them to forward data to the sink. A particular characteristic of the MULEs in our application is that they move along predefined trajectories indicated as service traffic roads in the airport. This implies the need for a data routing strategy to be used within each subnetwork in order to transfer the data collected by each sensor node to the ones located by the side of those roads. In this thesis, we call those nodes *subsinks*. Note that the communication scheme based on MULEs fulfills one of the main demands for the deployment of systems in airports, which is to keep the interference with the regular operations little or null. Figure 1.2 illustrates the proposed solution.

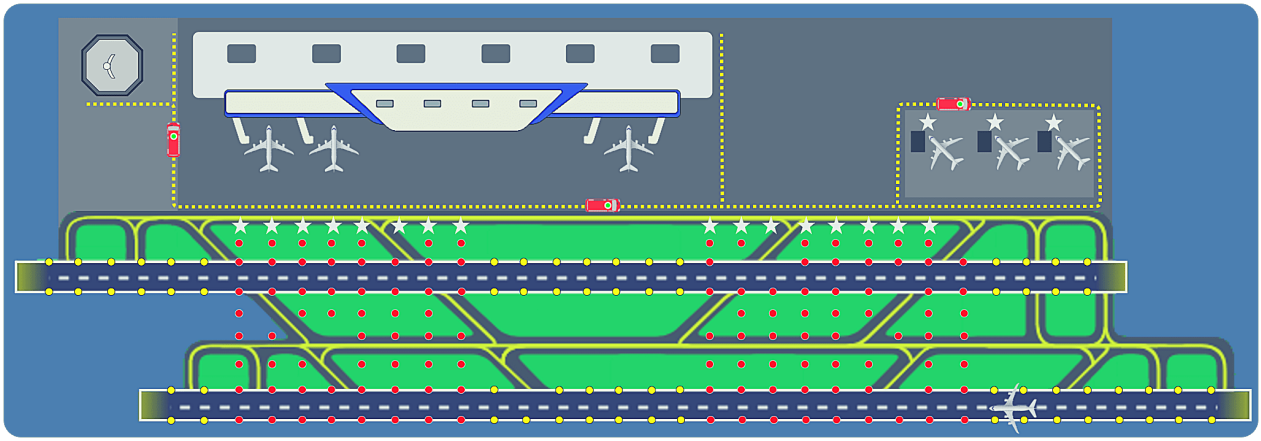


Figure 1.2: Proposed WSN-based monitoring solution. Sensor nodes (●,●) are suitably placed over green areas, by the side of runways and nearby parking spots to monitor environmental, safety-oriented and operational variables, respectively. Data MULEs (▬) doing regular tours occasionally pass by the side of the subnetworks making contact with the subsinks (☆). Along their trajectory and without stopping, the MULEs collect data from the subnetworks and move on with their duties. The data is delivered by the MULEs at the sink (e.g., the control tower ⬡) also opportunistically during the journey.

It is important to remark that this research idea was submitted for evaluation at four airports of different size (Castres-Mazamet, Montpellier, Perpignan, and Toulouse-Blagnac). All of them confirmed the pertinence of the proposed monitoring system and acknowledged a clear operational value on it.

1.3.1 Supporting communication technologies

In this section we explain our selection of short- and long-range communication technologies for the transfer of non-critical (environmental and operational) and critical (safety-oriented)

data in the proposed system. We start by discussing the predominant communication technologies nowadays and then we proceed with our selection.

Prevalent short-range technologies. Currently, the most popular alternatives for short-range communication are the IEEE 802.11, 802.15.1 and 802.15.4 standards, often referred to as WiFi, Bluetooth and ZigBee, respectively [20, 21]. IEEE 802.11 is intended for devices transmitting at relatively high data rate which have either wired power supply or access to frequent battery reload [20]. IEEE 802.15.1 and IEEE 802.15.4 are better suited for cases where the nodes have limited energy supply. The energy spent both during transmission and reception is about 1/4 of that of 802.11 [22]. IEEE 802.15.1 allows a higher data rate than IEEE 802.15.4, but the latter has a considerably greater communication range. More precisely, the maximum communication range of IEEE 802.15.1 is only 10 m compared to 100 m for IEEE 802.15.4 [20]. The maximum communication range of IEEE 802.11 and IEEE 802.15.4 is similar.

Although the three technologies have been implemented in a number of studies, IEEE 802.15.4 has been notably the preferred choice for WSN [23]. This is thanks to the fact that it offers a relatively large communication range at a low power demand, characteristics that fit well most WSN applications [20].

Prevalent long-range technologies. Currently available long-range technologies can be classified into two main groups named Cellular and Low-Power Wide Area Network (LPWAN) [24]. LPWAN technologies can in turn be subdivided into two groups: the ones using licensed mobile frequencies (LPWAN-L) and the ones using unlicensed ISM[5] bands (LPWAN-ISM) [26].

Cellular technologies (e.g., 2G, 3G, and 4G) provide high data rate long-range communication at relatively high energy spent [24]. A practical example of use of this technology is in mobile phones, which allow transfer of documents, web navigation and real-time video transmission, among other various functionalities, mostly supported on 3G and 4G, but require the user to recharge the battery about every day. The operation of this technology is conditioned to the availability of cellular coverage in the target region, which is often the case in most urban areas, but rather unusual in many rural, isolated or less populated places [27, 28].

As their name suggests, LPWAN technologies are specially conceived for applications requiring long-range communication at low energy spent. The moderate energy consumption is achieved at the expense of lower data rate than in Cellular technologies. In particular, LPWAN-L technologies (e.g., LTE-M, NB-IoT) ensure that the frequency band is used only by authorized devices making part of the network, thus preventing external devices from interfering in the communication. These technologies are conditioned to the availability of

[5]ISM stands for Industrial, Scientific and Medical. ISM bands refer to a portion of the RF spectrum originally reserved by the International Telecommunication Union (ITU) for those sectors. Over the years, these have become more crowded by short-range, low power systems like ZigBee, Bluetooth and Wi-Fi [25]

base infrastructure such as LTE antennas, infrastructure typically owned and managed by mobile operators [29], providing coverage in the target region of implementation. Since the expansion of this kind of infrastructure is costly, adaptations and tailoring of the network is rarely driven by the needs of one single customer, but rather by the trends in geographical distribution of the demand [30].

LPWAN-ISM technologies, on the other hand, use unlicensed parts of the spectrum to support communications. Unlicensed bands can be used by everyone without any specific authorization; thus, systems operating on these are prone to interference caused by external devices [31, 32]. However, they have the advantage that the user is not dependent on third party infrastructure, but is allowed to build his own private network and tailor it as desired. Sigfox and LoRa fall under this category. Apart from both using relatively low data rates, each adopts particular strategies to keep moderate the energy consumption. LoRa, for instance, implements an operational policy where the maximum data rate is decreased as the transmission distance is increased [33, 34]. Sigfox, for its part, imposes greater limitations on the packet size [24]. On top of that, these technologies must respect duty cycle regulations that strongly limit the number of packets that can be sent per hour [34]. The above makes these technologies more suitable for scenarios in which the traffic is considerably low.

Selected technologies

• ***For non-critical data: IEEE 802.15.4 standard (ZigBee) + MULE.*** Non-critical data in the ASAS application comprises environmental and operational variables. Environmental data in our system is expected to be gathered by most of the nodes in each subnetwork. Contrary to current monitoring practices based on a single high-cost weather station, we rely on several measurements performed at a collection of nodes spatially distributed over the airport. This monitoring approach is less sensitive to local factors such as variations in temperature or pollution due to the presence of clouds or the proximity to motorways, respectively. In addition, it allows for a wider scope of exploitation of the sensed data and provides greater degree of detail (e.g., in the construction of spatial analysis maps for the airport). Measurements of environmental variables are typically performed in a periodic fashion at rates up to 1 record per minute at some weather stations [35]. Operational variables on the other hand are planned to follow an event based monitoring scheme, checking events such as the use of different taxiways and parking spots. The frequency of registers would then depend on the intensity of the activities conducted at each airport.

Among the short-range technologies, IEEE 802.15.4 seems advantageous for us since it allows for a relatively large communication range at a low power demand. These two features fit well our ASAS application where the sensing field typically has an area in the order of tens of squared kilometers and the nodes are battery powered. Direct sending from each sensor node to the sink using a long-range technology seems also an interesting solution that would

simplify data routing. LoRa, in particular, appears more interesting since it allows a greater packet size which better fits the needs for environmental and operational sensing which might involve multiple variables of interest. However, the policy of reduction in maximum data rate as a function of the transmission distance is a high impact factor in the ASAS application, where the farthest nodes would be placed at various kilometers from the sink. On top of that, various recent studies [36, 37, 33] have pointed out scalability issues in LoRA linked to a weak prevention of collisions. For instance, [38] reported less than 14% of successful receptions in experiments involving LoRa-based systems with 250 and 5000 devices. Thus, although LoRa's official specifications indicate thousand of devices [39], the consequent compromise in quality and efficiency suggests that, at least for now, this technology might be reserved for applications with a moderate number of devices.

Based on the above discussion, our choice for non-critical data is to make the transfer in two phases: (i) sending of data from the sensor nodes to the subsinks at each subnetwork; and (ii) opportunistic collection of the data stored at the subsinks and further delivery at the sink by means of the MULEs. For the first phase, we opt for the IEEE 802.15.4 standard as short-range communication technology. This standard is most widely used in the 2.4 GHz (ISM) band at a data rate of 250 kbps and has a maximum packet size of 127 B [21]. Those characteristics fit well to the periodicity and type of information to be transmitted in our ASAS application. The data relay approach based on MULEs for the second phase is well suited for the non-critical data which is tolerant to delays.

• ***For critical data: LPWAN (e.g., LoRa).*** Critical data in the ASAS application corresponds to safety-oriented variables which are reported based on events. Examples of those are the presence of FOD on the runways and the surpass of a threshold in the temperature of the pavement of a runway. This type of register takes place with relatively low frequency in the system, but each time it requires quick notification of the control tower seeking for immediate reaction in order to prevent any incident.

Our choice for critical data is to perform direct sending through an LPWAN technology, which allows the fast notification of any threatening situation in the runways to the control tower. In particular, at this point, we favor LoRa based on the advantages mentioned in the discussion above. However, other LPWAN technologies may also suit the needs of the proposed monitoring system. Although we discarded LoRa for sending of non-critical data, it results suitable for sending of critical data since the transmission rate is considerably lower and just a few nodes are expected to be notifying a safety issue simultaneously.

▷ ***Remark:*** neither IEEE 802.15.4 which operates at 2.4 GHz, nor LoRa, which operates at 433/868 MHz in EU, 915 MHz in US and 430 MHz in Asia, interfere with the airports' operations [6]. Radio communication between control facilities and aircraft takes place in the frequency range of 108 to 136 MHz [40].

1.3.2 Main challenges for WSN-based monitoring in ASAS

The materialization of the WSN-based monitoring solution described above presupposes the assembly of technical and technological elements inherent to various disciplines, those being: (i) moderate-size sensor nodes able to perform for extended periods without need for battery replacement; (ii) reliable communication technologies for short- and long-range transmission; and (iii) a proficient routing protocol focused on the efficient use of the time in contact between the subnetworks and the MULE.

Element (i) has been proven feasible in previous studies which have implemented sensor nodes with the desired characteristics (see e.g., [41]). Element (ii) has been thoroughly discussed and supported in Section 1.3.1. Having verified that the proposal is feasible in terms of the first two elements, we focus on the third element, the routing protocol. In this application, the design of such a routing protocol must take into account the following particularities of the problem:

- ***Broad class of subnetwork topologies:*** as illustrated in Figure 1.2, the subnetworks located at different zones of the airport are likely to present heterogeneous topologies. Moreover, the presence of runways, buildings, fences, taxiway bridges, among other elements might contribute to this heterogeneity by prohibiting the deployment of individual nodes at some particular locations, leaving holes in the subnetworks. The routing protocol should be versatile enough to offer high performance independently of the topology formed by each subnetwork.

- ***Uncontrolled MULEs:*** the communication system is assumed to have no control on the behavior of the MULEs. The visits of the MULEs to the different subnetworks will then occur as part of the regular transition of service vehicles during ordinary operation. The MULEs are also not expected to stop near any subnetwork to facilitate the collection of all stored data. Hence, the routing protocol should implement a convenient strategy for the transfer of the data gathered at each node to the MULE, making efficient use of the time in contact with it.

- ***Nodes with limited lifetime:*** in order to remove the limitations derived from the need of wire installation discussed in Section 1.2, we rely on battery-powered sensing devices. Since most of the data is expected to be transferred using short-range communication, we envisage the use of application based sensor nodes such as TelosB [42] or MICAz [43] which come equipped with IEEE 802.15.4 (ZigBee) compliant RF transceiver. Nodes of this type run on two AA batteries with a nominal initial load of 18720 Joules [44], providing an expected battery lifetime of 1 to 5 years depending on the usage. The energy supply is thus not a critical issue for us as in certain (mostly military) applications where the nodes count with just 1 Joule as initial battery load.

Nonetheless, the elongation of the lifetime of the nodes translates into maintenance cost savings for the airport and thus, must be pursued. The routing protocol should then pursue an efficient use of the sensor nodes in their role as data receivers and forwarders.

- ***Nodes with limited communication range:*** as a way to preserve the battery of nodes and prevent congestion issues, we privilege short-range communication for the transfer of all non-critical data (which as said before, is expected to be the large majority of it). The routing protocol should be able to integrate these communication constraints into the data transfer scheme.

- ***Need for self-organization:*** the proposed system is expected to embrace multiple subnetworks located at diverse areas of the airport. A straight way to guarantee the functioning of every subnetwork, regardless of its remoteness to the control center is to provide them with the ability to self-organize. This avoids the need for intense data transfer to a central coordinator node, and helps to keep low the acquisition cost of the solution by allowing some nodes (those not involved in safety-oriented monitoring) to be equipped with just short-range communication technology. The routing protocol should thus allow the configuration of each subnetwork for data transfer, based only on local interactions of the sensor nodes.

- ***Connectivity based on failure-prone devices:*** communication in WSNs inherently depends on the proper functioning of the nodes, which permanently perform as data forwarders. Hence, the loss of one node (e.g. due to failure or battery depletion) might leave multiple reliant nodes unable to send their data to the sink. Under this type of situation, the routing protocol should be able to opportunely react and provide alternative paths to keep the network functioning. The protocol should be able to achieve this in a decentralized manner, following the ideas presented in the previous paragraph that motivate self-organization.

1.4 Organization of the manuscript

The remainder of this manuscript is structured as follows:

Chapter 2. Routing protocols for WSNs: state of the art

As explained in Section 1.3.2, this thesis focuses on the construction of the routing protocol that would serve as the base of the proposed monitoring system. This chapter presents a literature review on routing protocols for WSNs with emphasis on four families of protocols that might result of particular relevance in the development of our method: (i) flat routing, (ii) cluster-based routing, (iii) routing in settings involving MULEs, and (iv) routing based on advanced optimization techniques. A detailed analysis of the protocols in those categories

allow us to identify useful techniques that could be exploited within our method and build up a list of research gaps, various of which are bridged in this thesis. This chapter extends our work published in the proceedings of the International Workshop on Communication Technologies for Vehicles (2018) [45].

Chapter 3. WSN for ASAS: modeling and initial solutions

This chapter presents our first steps in the modeling and resolution of our WSN routing problem. There we implement and evaluate the two routing protocols from the literature, devised for settings involving MULEs, that better match our routing context and our selection of technologies. We compare the two protocols through computer simulation in OMNeT++. The results lead us to conclude on the ability of one of the two protocols to provide optimal solutions when the subnetwork has a regular grid-like topology, and open the discussion on the need for the development of a more sophisticated method able to deal with a more general class of subnetwork structures. This work was published in the proceedings of the International Conference on Ad-Hoc Networks and Wireless (2019) [46].

Chapter 4. Advanced routing based on Ant Colony Optimization

Following the conclusions of Chapter 3 and motivated by the large class of subnetwork topologies that may arise in the ASAS context, in this chapter we undertake the construction of a protocol able to provide high quality routing solutions for a general class of subnetwork topologies. The proposed protocol is founded in the Ant Colony Optimization technique, which provides a general framework for the optimization of multiple objectives of performance of the subnetwork in a distributed manner. We name this protocol ***ACME***, standing for *ACO-based routing protocol for MULE-assisted WSNs*. ACME is designed to select the subsink that each node must use as gateway to reach the MULE during an opportunistic visit to the subnetwork, and also the multi-hop path that each packet must follow from its source node to its gateway subsink. In this chapter, we model the routing problem as a multi-objective optimization task where the conflicting objectives of minimization of total path length and minimization of total subsink imbalance are considered as main targets. The minimization of total number of retransmissions is considered as a secondary, optional target. A thorough description of ACME and several technical implementation details are provided along the chapter.

Chapter 5. ACME's performance assessment

This chapter presents the computational experiments carried out to validate our Ant Colony based routing protocol, ACME. The experiment is split into three parts addressing the following aspects: (i) the ability of the protocol to provide high quality solutions for a variety

of subnetwork topologies when only the total path length and the total subsink imbalance are optimized; (ii) the impact on performance of adding the minimization of total number of retransmissions as a third objective of optimization; and (iii) the comparison between the efficiency in the interactions with the MULE achieved by ACME and that offered by the best performing routing protocol from Chapter 3. Our computational experiments in OMNeT++ show ACME's ability to provide high quality solutions for a general class of subnetwork topologies and its suitability for application in real life ASAS instances.

Chapter 6. Conclusions and future research

This chapter summarizes our work, contributions and scientific production, provides our general conclusions and presents some promising research lines for future development.

Chapter 2

Routing protocols for WSN: state of the art

The chapter in brief

This chapter presents a literature review on routing protocols for WSN. The review is divided in two main sections, each corresponding to one family of protocols of particular relevance for the development of this thesis. In Section 2.1, we first discuss *flat routing* protocols where all the nodes have equal roles and functionality [47, 48]. While simplistic, those protocols might be valid solution approaches for our problem, or at least, serve as the base of a more sophisticated approach, if required. In Section 2.2, we discuss *cluster-based routing*, where the nodes are given responsibilities for the data forwarding process based on a specified hierarchical structure [49, 50]. Although most of the protocols found in this category do not explicitly consider MULEs, it is possible to find a strong affinity between them and our problem if we see the subsinks as a special type of node used to convey the data from groups of nodes to the MULE. Then, in Section 2.3 we move on to *routing in settings explicitly involving MULEs*. This section paves the road to a variety of concepts relevant to handle MULE specificities, but are naturally neglected by most of the studies discussed in Sections 2.1 and 2.2. The major part of studies covered up to Section 2.3 propose systems of rules which are relatively simple from the perspective of current optimization techniques. In Section 2.4 we study more sophisticated *routing protocols based on advanced optimization* methods such as Ant Colony Optimization [51], Bee Colony Optimization [52] and Genetic Algorithms [53]. Section 2.6 wraps the section out with a synthesis on the topics discussed, the key ideas to keep in mind for next chapters and the open challenges for routing in S-WSN.

Contents

2.1 Flat routing

• ***Flooding:*** this is one of the simplest and oldest routing protocols one could find in the literature. Some of the earliest allusions to it could be found in [54] and [55]. Differently from most modern approaches, Flooding does not involve routing tables. Indeed, it departs from the idea that nodes are unaware of identities and locations. As its name suggests, what this strategy proposes is that every time a node has some information to share, it is propagated through the whole network until it reaches the interested destinations. To achieve that, the sender node broadcasts packets to its neighbors which repeat this process by broadcasting the packet to their neighbors as well; the process is repeated until all nodes in the network have received the packet. Any node will drop a packet that has reached a predefined maximum number of hops. The procedure is illustrated in Figure 2.1.

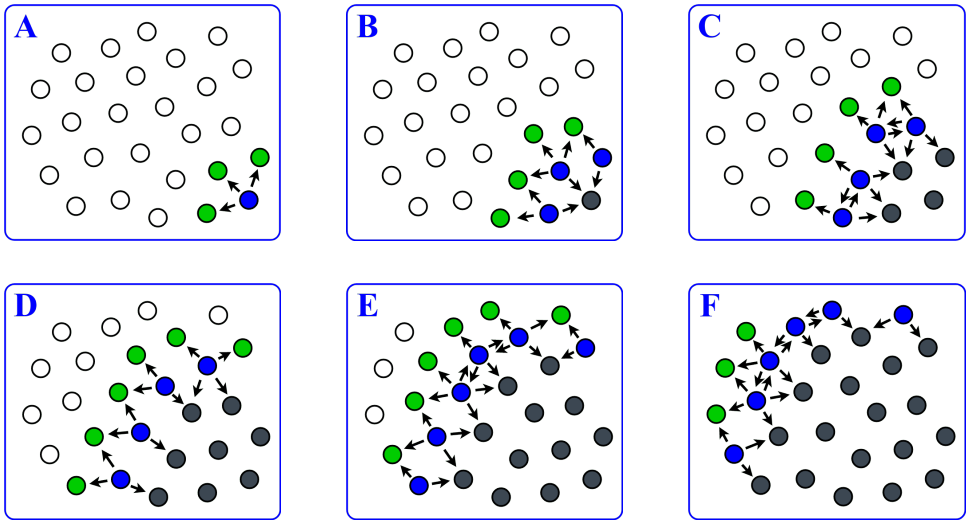


Figure 2.1: Flooding protocol.

The main strengths of this protocol are its simplicity and the low expectancy of a packet not reaching its destination. Those come at the cost of three major drawbacks: (i) most of the nodes receive several copies of each message, (ii) some nodes sense overlapped areas and thus send similar information to their neighbors, and (iii) energy consumption is disregarded. These three issues are well known in the literature by the name of *implosion*, *overlap* and *resource blindness* [47, 56]. As will become manifest throughout this survey, implosion and resource blindness have been addressed by almost every routing protocol proposed afterwards. Overlapping, on the other hand, is not addressed by various popular routing protocols (e.g., GAF [57] and RPL [58], which will be examined later). However, one could argue that some data aggregation scheme could be used on top of most routing protocols. Most of the protocols that *explicitly* take this issue into account lie in the category of cluster-based routing, explored in detail in Section 2.2.

• ***Gossiping:*** this protocol is an adaption of flooding where the nodes forward data to only one of their neighbors, which is selected randomly, instead of broadcasting the data to all of

them [54] . As a direct positive effect of this change, the implosion is reduced. However, it has been discussed that this strategy implies an important increment in the latency and a relatively high chance of the packets not reaching a suitable destination [59] as a consequence of the random selection of the next hop. The procedure is illustrated in Figure 2.2.

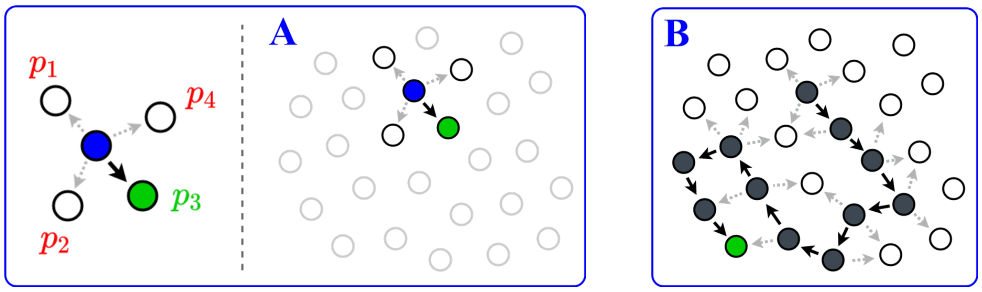


Figure 2.2: Gossiping protocol. The values $p_1, \ldots, p_4$ denote probabilities of selection as a next hop for the packet.

• ***Sensor Protocols for Information via Negotiation (SPIN):*** instead of a single protocol, SPIN [54, 56] embraces the family of protocols SPIN-PP, SPIN-EC, SPIN-BC and SPIN-RL. Briefly said, all four protocols depart from the idea of Flooding and work upon the concept of negotiation to overcome the issue of implosion. Negotiation consists of each node notifying its neighbors about the availability of some data before actually sending it, and then moving forward with the transfer only to those neighbors which reply with a data request. This mechanism, in turn, indirectly mitigates resource blindness by preventing much of the communication overhead related to duplicate packets.

The authors of SPIN also claim to resolve the issue of overlapping. To support this, they assume that the nodes are able to form a compact representation of any sensed data composed of named elements, that could be then used during negotiation instead of the original data in full representation. Thereby, upon receiving an advertising packet containing this compact description of the available data, a node must be able to indicate which pieces of the data it is interested in obtaining. However, in practice, it is not clear how most forms of sensed data sensed (e.g., sound signals or images) could be represented by a collection of named elements for use during the negotiation. This practical limitation of SPIN was also noted by [60].

Despite the aforementioned limitation, SPIN offers improved performance than Flooding and Gossiping thanks to the mitigation of implosion and resource blindness.

• ***Directed Diffusion:*** in contrast to the three previous protocols, Directed Diffusion [61, 62] considers a sink which occasionally requests specific data through query messages called *interests*. The interests are propagated through the network using flooding until they reach a node with matching event records. Such a node is called *source*. A characteristic that differentiates this from most other WSN protocols is the assumption that interests are

persistent, meaning that if a source has information relevant to the sink, then this last will be interested in repeated measurements from that source for some period of time. The network exploits this extended communication scheme by progressively improving the path while the sending of records is performed. To achieve this, the sink puts a fixed period of validity to each interest it delivers, and keeps extending the lifetime of interests only on the paths through which the event records are taking less time to be received. The interests thrown on the longest paths are thus left to expire and the frequency of traffic is progressively raised upon request of the sink.

In Directed Diffusion the network progressively learns how to efficiently route messages between a particular source-sink pair. This type of strategy is especially useful when the interaction between a source and a sink is expected to last for a sufficiently long time such that the network is able to amortize the cost of finding efficient paths during their period of use. The next protocol in this revision presents an alternative for situations where such an interaction is expected to be brief.

• ***Rumor Routing:*** this protocol has basically the opposite assumption than Directed Diffusion. In Rumor Routing [63], the sink is presupposed to require just a small amount of data from a given source; thus, the authors of Rumor Routing argue that flooding every query throughout the whole network in order to find the best path may be more inefficient than delivering it by a non-optimal one. In rumor routing, the nodes observing an event dispatch a small number of a special type of packet called *agent*. Those advance through the network tracing a random path while updating the routing tables of the nodes they visit in order to let them know how to reach the source of that event. From a metaphoric perspective, the agents *spread the rumor* about the availability of information on some particular event and how to reach it. When a sink becomes interested in an event, it generates a query which starts to explore the network using a random walk as well. As soon as the query finds a node which already knows the path towards the matching event (thanks to the action of the agents), it redirects the query to the event directly through this path. The full sequence of links covered by the query from the sink to the source constituted the path finally used to transfer the information on the event.

• ***Geographic and Energy-Aware Routing (GEAR):*** in Flooding, Gossiping and SPIN, the information transfer is triggered by the record of new information at some sensing device. In Directed Diffusion and Rumor Routing it starts with a sink exhibiting interest for some type of event through queries. GEAR [64] addresses the case where the sink is able to specify the specific region of the sensing field for which it is interested in obtaining information. It was proposed as an improvement to Directed Diffusion where the interests are routed directly to the target region instead of flooding them through the whole network [64]. The main assumptions of this protocol are that the nodes are aware of the location and energy

level of themselves and their neighbors. To back up these assumptions, the authors rely on the availability of some positioning system (e.g., GPS, Galileo, GLONASS) and the occasional exchange of information between neighbors informing their energy level.

The routing of queries is done in two parts. In the first part, the query is routed from the sink to any node in the target region. To do so, each time a node receives an interest, it selects a neighbor as the next hop by minimizing the weighted sum of (i) normalized distance to the target region and (ii) normalized residual energy. The second part starts when the query reaches a node belonging to the target region. From that point and on, the query is disseminated using either a Restricted Flooding (RF) or a Recursive Geographic Forwarding (RGF) algorithm. RF works similarly to the regular Flooding strategy explained above, with the difference that only nodes within the target region forward the query. In RGF, the first node who receives the interest divides the target region into subregions and forwards the interest to one node in each of them. Each of the nodes that receive the query repeat the process of dividing and forwarding within their corresponding subregion. The process is repeated until every subregion contains only one node. At that point, every node in the target zone has received the query. Data dissemination to the sink is done identically as in Directed Diffusion. The authors recommend the use of RF when the nodes on the target region are sparsely deployed, and RGF for the contrary case of a dense coverage.

Certainly, the approach for query routing in GEAR might provide substantial energy savings compared to Directed Diffusion. Nonetheless, the way of identifying and specifying the shape of the subregions might be a difficult task that is somehow overlooked by the authors. The whole methodology and the simulations consider only the simple case of a rectangular sensing field and rectangular subregions which could be easily represented by a set of four coordinates. The general case with more complex shapes of both sensing fields and subregions is left unattended in the article.

• ***Geographic Adaptive Fidelity (GAF):*** this protocol is very similar to GEAR in many ways, including that GAF [57] also assumes that the nodes are provided with a positioning system, and also that it splits the sensing field into squared subregions. In this case, an important condition is that the nodes in a given subregion are able to reach all the nodes within the adjacent subregions with a single hop. Under these two assumptions, the functioning of GAF is quite simple; within each subregion, the radio system of only one node is kept on at each time (*active node*), and the radio system of the others is turned off (*inactive nodes*). In addition, GAF considers nodes positioned at the same subregion as equivalent in terms of cost of packet routing. Thus, if the nodes within each subregion alternate the role of active node, the lifetime of the network results prolonged compared to a setting where the radio system of all nodes is kept on permanently. The role of active node within each subregion is periodically rotated, selecting at each time the node with highest residual energy.

Although the idea of strategically switching off some nodes from time to time might be beneficial, the approach proposed in GAF reduces to deploying redundant nodes over the network in order to obtain a longer average node lifetime. This, of course, comes at considerably larger acquisition costs, which might limit the scope of the protocol.

• ***Routing Protocol for Low-power and Lossy Networks (RPL):*** inspired by the classical Dijkstra's algorithm [65], RPL [58] provides a method to build a shortest path tree, commonly named, a Destination Oriented Directed Acyclic Graph (DODAG) [66, 67] which is typically rooted at the sink. The protocol works in a purely distributed manner and does not depend on positioning systems. Moreover, RPL is standardized by the Internet Engineering Task Force (IETF) group since 2012 [58]. Naturally, it is an ideal choice in terms of path length for networks with a single, fixed sink. Thereby, it has been pointed by some authors as a predominant routing protocol for the Internet of Things (IoT) [68].

In RPL, the paths from all the nodes to the sink are built simultaneously as follows. The whole procedure is supported in a special type of packet called DIO (standing for DODAG Information Object). Each time a node sends a DIO, it specifies its current lowest cost to reach the sink. For the sake of exposition, let us consider a cost expressed in terms of number of hops. The algorithm functions for any generalized cost function which might include the energy, distance or any other feature of interest. The procedure starts with the sink storing a cost of 0 and all the other nodes storing a cost of ∞. The sink broadcasts a DIO which is received by its neighbors. Then, each neighbor compares its current cost with the one it would have if it adopts the sink as its next hop. If the proposed cost is convenient, the neighbors of the sink adopts it as their parent (i.e., as next hop to reach the sink). Next, each of the neighbors of the sink repeat the process by broadcasting a DIO with their new cost to reach the sink. Their neighbors, in turn, compare their current cost to see if it is convenient for them to adopt the corresponding sender of the DIO as a parent, and if so, they proceed accordingly. The procedure is repeated multiple times until all the nodes are attached to the DODAG. The resulting graph is a tree where every node has a single path to reach the sink, which happens to be the shortest path connecting them. Thereafter, each node uses the routes obtained by the algorithm to transfer data to the sink. The only information each node needs to store is the ID of its parent. The procedure is illustrated in Figure 2.3.

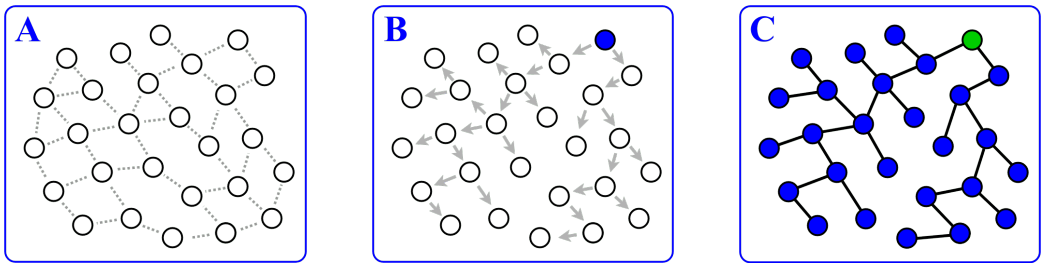


Figure 2.3: RPL protocol.

2.2 Cluster-based routing

Cluster-based protocols constitute one of the main streams in WSN routing. Several surveys have consistently appeared over the last 15 years to report progress in this specific direction (see, e.g., [69, 70, 71, 72, 73]). The base idea of clustering is to arrange the sensor nodes by groups and then split the process of communication with the sink into three phases: (i) transmission from the cluster members to a cluster-specific leader node called cluster head (CH); (ii) aggregation of the data from the different cluster members at each CH; and (iii) transmission of the aggregated data from the CHs to the sink. Both, phases (i) and (iii), can be done either by single-hop or multi-hop routing (see Figure 2.4).

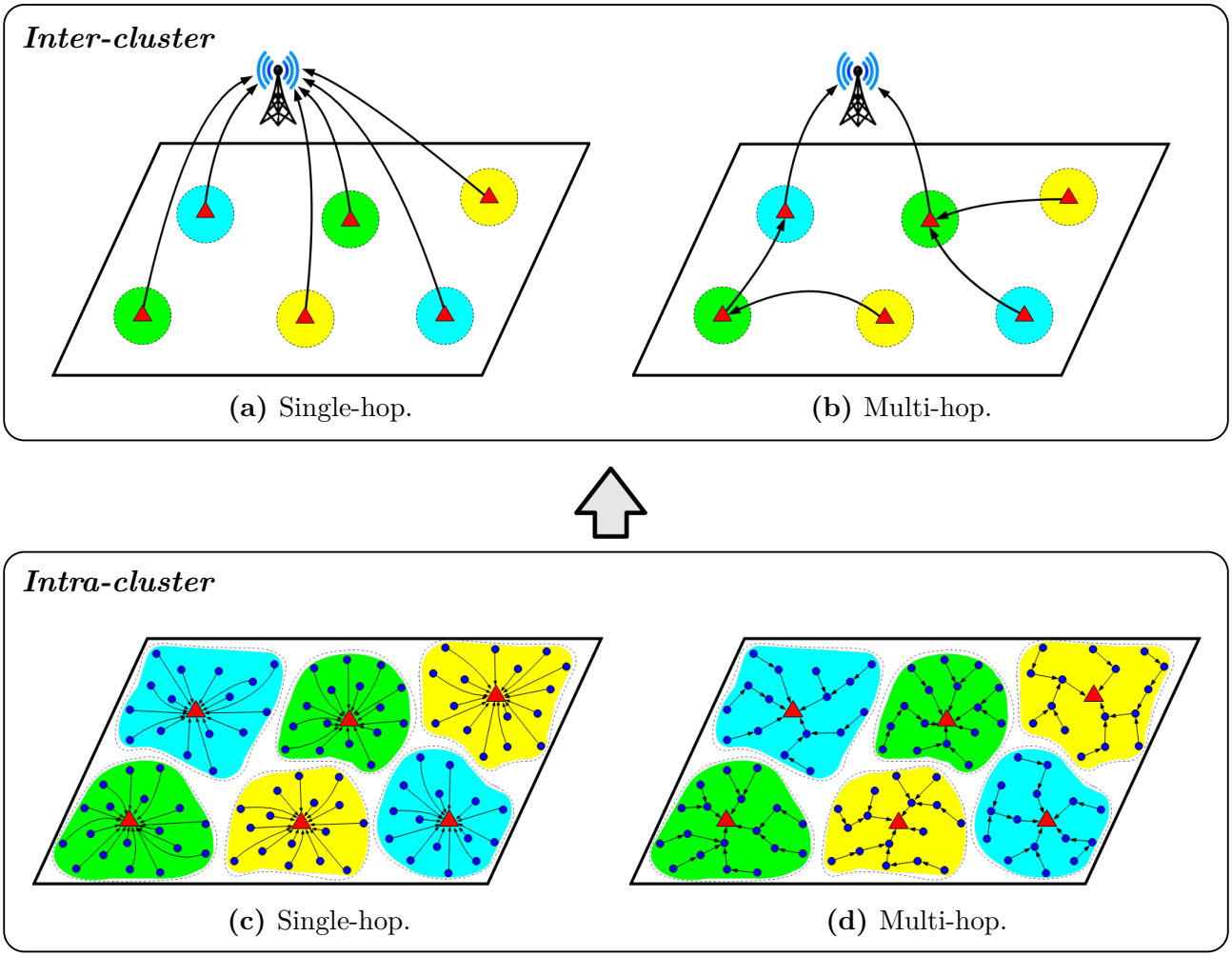
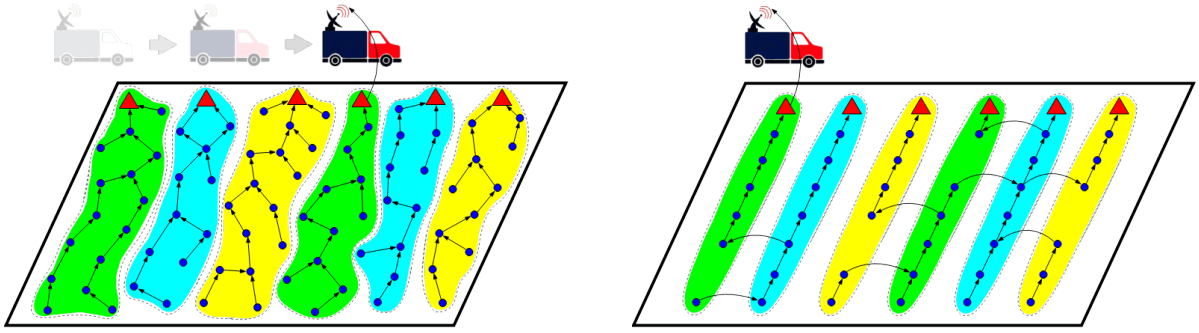


Figure 2.4: Cluster-based routing schemes. Clusters specified by the colored shades. Bottom frame: ways to transfer data from the sensor nodes (●) to the cluster heads (▲). Top frame: ways to forward the data to the sink. Both (a) and (b) are are compatible with (c) and (d).

Clustering-based routing emerged as an alternative to direct transmission (single-hop communication between the nodes and the sink) and multi-hop routing, seeking for an extended network lifetime [74, 75]. The postulate is that clustering might help to achieve a more

homogeneous power consumption over the network than in multi-hop routing and, strongly relying on significant reduction of the total transferred data thanks to the aggregation at the CH level [76], it might also lead to lower energy consumption per node than in direct transmission.

As briefly mentioned in the introduction of this chapter, there are some similarities between the arrangement of sensor nodes adopted in this thesis and the one typically used in cluster-based routing. Following our decision to work with multi-hop routing (see Section 1.3.1), each of our subnetworks could be seen as a cluster-based system where the subsinks play the role of CHs and groups of sensor nodes are assigned to them (see Figure 2.5). Of course, some differences between the two settings are immediately noticeable. It stands out, for instance, the fact that our subnetworks need to communicate with a mobile device while the sink in cluster-oriented studies is regularly set fixed. In addition, in our problem the role of subsink comes implicit with the relative position of each sensor node to the pathway followed by the MULE, while cluster-based protocols involve the selection of the CHs. In spite of these differences, it could be worth taking a look at the cluster-based protocols proposed in the literature and check if we can find some ideas for our method. Thus, the remaining of this section is dedicated to the examination of various papers in cluster-based routing.



(a) Random structure. **(b)** Grid-like structure with holes.

Figure 2.5: Parallel between our ASAS setting and that of cluster-based routing. In analogy to Figure 2.4, the groups of sensor nodes (●) assigned to each subsink (▲) are indicated by colored shades. Each subfigure represents **a single subnetwork**. The random and grid-like structures are used for the sake of exposition.

Overview of existing protocols

• ***Low Energy Adaptive Clustering Hierarchy (LEACH):*** borrowing ideas from clustering algorithms in various contexts (see details in [77]), LEACH [78] is acknowledged by various authors [79, 80, 81] as the first energy-efficient clustering protocol for WSNs. Its base idea is to periodically rotate the nodes that play the role of CH, seeking for a homogeneous energy dissipation over the network in the long term. The protocol involves a set-up phase where the clusters are formed, and a steady-state phase where the data is collected by the sensor nodes and forwarded to the sink in a periodic way (see Figure 2.6). Multiple iterations

of data collection and forwarding to the sink (so-called frames) might take place between each pair of set-up episodes. The protocol performs based on single-hop routing both at the intra-cluster and inter-cluster levels (frames (c) and (a) of Figure 2.4, respectively). In order to prevent packet collisions during the steady-state phase, it relies on a TDMA based MAC protocol to schedule the transmissions from each cluster member to its corresponding CH.

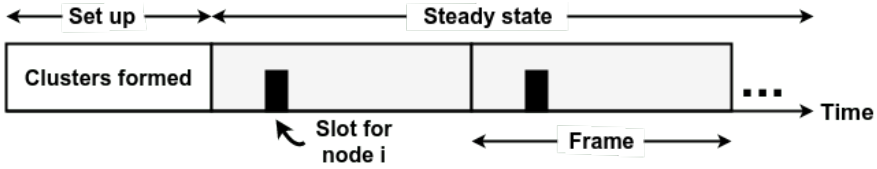


Figure 2.6: Timeline for LEACH's operation

At each set-up phase, each node decides whether or not to be a CH for the next steady-state phase based on a random decision rule which takes as input the target percentage of nodes to be CHs; a parameter that must be defined by the network administrator. Due to the randomness of the decision and the distributed nature of the procedure, this percentage is not precisely attained at each round, but is expected to be met on average over multiple set-up rounds. LEACH is illustrated in Figure 2.7.

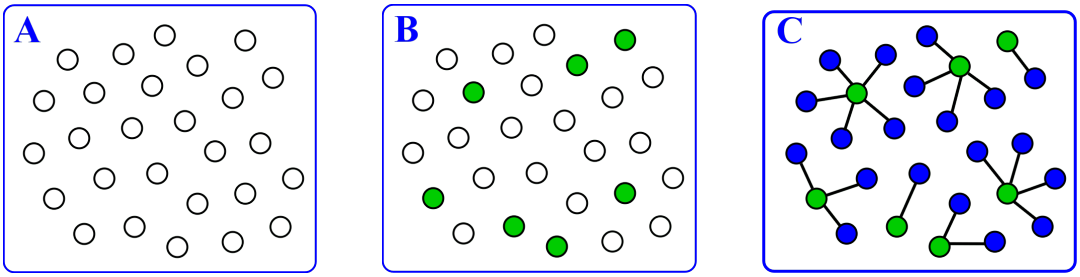


Figure 2.7: LEACH protocol.

Some of the main assumptions in LEACH are that all the nodes [74, 82]: (i) have the necessary processing capabilities and the ability to coordinate intra-cluster transmissions; (ii) are able to adjust their transmission power; (iii) are able to directly communicate with the sink. Two further assumptions that are often disregarded in the literature but which are highly relevant as well are: (iv) data aggregation is a suitable procedure for the application of interest; and (v) all the nodes possess the processing capabilities to perform data aggregation. Since most of these assumptions are adopted by the next reviewed protocols, we leave a more thorough discussion of them for the concluding section of this chapter (Section 2.6).

• ***LEACH-C:*** this centralized version of LEACH is proposed in [83], seeking to ensure the desired percentage of nodes functioning as CHs and improve their distribution over the sensing field. To achieve this, LEACH-C relies on the ability of the sink to resolve the

problem of grouping the sensor nodes into k clusters such that the sum of euclidean distances between CHs and corresponding cluster members is minimized. The computation of distances requires the nodes to keep the sink informed about their current location at each set-up phase, which implies the need for a relative positioning method (e.g., trilateration or triangulation [84, 85, 86]) or an absolute positioning system (e.g., GPS, Galileo, GLONASS). In addition, the stated optimization task corresponds to the well-known k-medoids problem [87] classified as NP-hard [88]. Thus, a significant processing power in the sink is implicitly assumed in order to be able to find high quality solutions to realistic instances (in the order of 10^2 to 10^4 nodes [89, 90]) in reasonable time. If these technical requirements are met, LEACH-C certainly has the potential of outperforming the traditional LEACH.

• ***LEACH-EP:*** this protocol proposed in [91] turns back to random CH selection as in the basic version, but incorporates the residual energy of the nodes into the random decision rule. The aim is to prevent nodes with relatively low residual energy to be selected as CHs since those are likely to not be able to make it until the end of the upcoming steady-state phase. A short centralized step is required, where the CHs communicate to the sink their residual energy and the number cluster members linked to them. That information is used for the selection of CHs in the next set-up round. Based on the results presented in [91], LEACH-EP is able to achieve a more homogeneous energy consumption over the network than LEACH.

• ***LEACH-R:*** in the same line as LEACH-EP, LEACH-R [82] takes the residual energy of the nodes into account when selecting the CHs. In addition, this protocol adds another level of hierarchy to the scheme by introducing a so-called *node R* responsible for the aggregation of data from the CHs and further transfer to the sink. Thus, in LEACH-R, unlike LEACH, LEACH-C and LEACH-EP, the CHs do not communicate directly with the sink. The reasoning behind the addition of a node R lies in the assumption that the sink is considerably distant from the network. Hence, some energy savings could be made if only one node aggregates the data from the CHs and performs the energy-intensive task of communicating with the sink a relatively low number of times. At each set-up round, the node R is selected among the CHs as the one with the largest ratio of residual energy to distance to the sink.

• ***LEACH-DT:*** as in LEACH-R, the relative position of the distance with reference to the network is considered in LEACH-DT [92], however, this last only relies on regular nodes and CHs. In LEACH-DT, it is the random decision rule for selection of CHs what takes into account the distance separating the nodes from the sink. Based on the assumption that the sink is significantly distant from the network, the authors offer a mathematical derivation for the probability of selection of node i that should be used within the random decision rule for CHs in order to obtain a perfectly homogeneous energy consumption over these last.

• ***Threshold sensitive Energy Efficient sensor Network protocol (TEEN):*** although the name does not make it evident, TEEN [93] is an extension of LEACH as well. As its main differentiating feature, TEEN is tailored for event-based data collection. The adaption to achieve this is straightforward and consists of sending data only when important changes in the sensed variables occur. More specifically, the nodes only send data in two situations: (i) when the sensed variable suddenly overpasses a predefined threshold, and (ii) when the variable had already exceeded the threshold in a previous sensing and now it has a significant change showing either a return to regular condition or further distancing towards a critical condition. The authors also propose the use of two levels of CHs instead of only one as in the previously reviewed protocols; however, they do not indicate any procedure for the selection of the CHs belonging to each level.

• ***Adaptive Threshold sensitive Energy Efficient Network protocol (APTEEN):*** briefly said, APTEEN [94] is a generalization of LEACH-C which accounts for both, periodic and event-based data collection. Similarly as in TEEN, two levels of CHs are proposed but no indications are provided regarding the selection of the CHs for each level.

• ***Power-Efficient Gathering in Sensor Information Systems (PEGASIS):*** strictly speaking, PEGASIS [95] is not a cluster-based protocol. However, we find it relevant to include it in this category since it preserves the notion of hierarchy present in all cluster-based protocols. On the other hand, it has consistently been compared to cluster-based protocols in the literature (see e.g., [96, 97]). PEGASIS departs from the idea that some energy savings could be made if: (i) a single node (so-called leader) is used to forward the data of the whole network to the sink, (ii) the communication of all the other nodes is kept limited to cheap transmissions to neighbors, and (iii) the data is systematically aggregated on its multi-hop way towards the leader. To materialize this concept, PEGASIS proposes to arrange the nodes on a chain-like structure (see Figure 2.8) and implement an aggregation scheme based on the distributed projection of the data collected by the n nodes in a space of lower dimension m, where each node must perform exactly m data transfers. Naturally, the whole proposal depends on the possibility to achieve significant data compression (i.e., $m << n$) so that the amount of short-range transfers between nodes and long-range transfers to the sink is kept fairly moderate. This condition immediately lowers the expectations for situations where the post-treatment requires the data exactly as collected or with a minimal error. Similarly, the application in contexts involving highly variable data (difficult to compress) might also be hard to justify in terms of energy consumption. Lastly, it is worth mentioning that this PEGASIS makes the role of leader periodically rotate among the nodes. Therefore, it retains the assumption made by all the other cluster-based protocols examined up to now, specifying that all the nodes are able to reach the sink through direct communication.

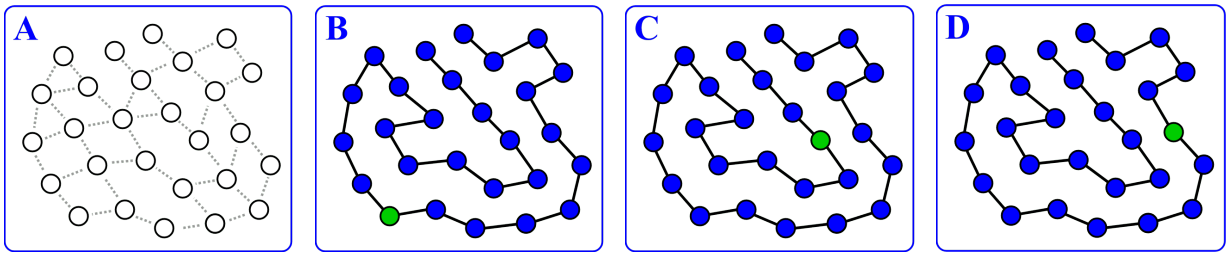


Figure 2.8: PEGASIS protocol.

• ***Energy Efficient Hierarchical Clustering (EEHC):*** the EEHC is proposed in [77] as a cluster-based protocol involving multiple levels of CHs. The motivation to use multiple levels comes from the idealism that significant data compression is doable at any CH, irrespective of its level. Unfortunately, in practice data can only be compressed to some extent before becoming useless; any further layer of CHs after that point might be useless or even detrimental in terms of data recovery or energy spent. For the selection of the CHs, each node generates a random value in $[0, 1]$ and compares it to a reference value p standing for the probability of a node becoming CH. In contrast to the other protocols revised up to this point, EEHC allows cluster members to be $k > 1$ hops away from the CH, favoring the multi-hop intra-cluster routing scheme (frame (d) of Figure 2.4). Most part of the technical development in [77] is dedicated to the derivation of analytic expressions for the optimal p and k values in terms of the total energy spent in the system during one round of data collection, in the specific case of a single level of CHs. All the developments are done under the assumption that the sensing field is a square region with the sink located at its center and all the other nodes homogeneously but randomly distributed over it. The authors attempt to generalize the results to $L \in \mathbb{N}$ CHs, but they only succeed with the expression for k. Numerical approximation is proposed as an alternative to estimate the optimal value of p.

• ***Fast Local Clustering Service (FLOC):*** as some of the other methods previously discussed, FLOC [98] works upon a deficiency found in most LEACH-like protocols. The authors argue that the CHs in LEACH are prone to experience unnecessary network contention induced by members of other clusters located within their range of communication. This condition is commonly referred to in the literature by *cluster overlap*. Thereby, FLOC seeks to select the most CHs in such a way that each node is within the communication range of only one CH. By doing that, the CHs only receive packets from the members of their cluster, which do not need to compete against members of other clusters to access the channel. The process to select the CHs and form the clusters is done in a distributed manner. Sensor nodes dynamically propose themselves as CHs and their neighbors store their proposal. If a node relatively close to an already announced candidate receives a new proposal from another node relatively close to it, the node replies with a conflict message indicating that its election

would cause an overlap. Following a conflict message, the node making the new proposal gives up on its candidacy. After some fixed time without receiving a conflict message, a node becomes elected as CH. Finally, the nodes which are not selected as CHs join the closest one, using as a measure of proximity the strength of the signal perceived for the received candidacy messages. As a side positive effect of the above described process, the CHs are approximately uniformly distributed over the sensing field, which prevents excessive energy spent by nodes located relatively far from their CH (as might occur in LEACH).

• ***Efficient Honeycomb Clustering Algorithm (EHCA):*** the EHCA [96] recently raised as a geometric approach to balance energy consumption over the network in a cluster-based scheme. As its name indicates, EHCA adopts clusters with the characteristic hexagonal shape of the honeycombs; the sensing field is supposed to be circular and multiple sinks are supposed to be uniformly distributed around it. The authors claim that much energy is wasted in most cluster-based protocols by periodically refreshing the clusters. Thus, they propose to define the clusters only once and then just rotate the CH role among the cells in the vertices of each honeycomb cluster.

The protocol performs in two phases. In the first phase, the assembly of honeycomb frames is placed over the sensing field. The exact position and rotation of this structure determines how the sensor nodes are grouped in clusters. Each node is assigned a coordinate based on a relative positioning frame with origin at the geometric center of the structure. An important assumption in this regard is the existence of a number of anchor nodes[1] necessary for triangulation to work. Although not explicitly stated in [96], a centralized approach is likely to be needed in this phase in order to achieve identifiability of the nodes in the proposed coordinate system.

Once the coordinates have been established, the second phase starts. For this phase, each cluster is split into hexagonal cells. Each cell is assumed to contain at least one node, and might contain more than one. This phase is conducted by rounds as in LEACH-like protocols, but each round only requires the rotation of each CH within the vertex cells of its cluster. Following the selection of the new CH, the network operates by collecting data and forwarding it to the sink until the new round. During this period, every cell keeps active only the node with highest residual energy and all the others are set to sleep (similarly as in GAF). EHCA implements multi-hop intra-cluster combined with single-hop inter-cluster routing (frames (d) and (a) of Figure 2.4, respectively). Intra-cluster routing is based on the recursive selection of the nearest neighbor, using the residual energy to break ties.

A set of simulations seems to demonstrate the superiority of EHCA over GAF [57], PEGASIS [95] and HEER [101] (all examined above) in terms of nodes alive, the latency of data delivery,

[1] Anchor node is a popular name used in WSN literature to refer to nodes aware of their absolute or relative position [99, 100].

and the percentage of successful data delivery to the sinks. However, scarce details are offered about the implementation of the benchmark protocols which makes it harder to determine the validity of the conclusions. On the other hand, the experiment performed seems to fit well the assumptions of the method about the shape of the sensing field and distribution of sensor nodes. The conclusions and the suitability of the methods are likely to change if those assumptions were removed.

• ***Sleep Scheduled and Tree-Based Clustering (SSTBC):*** the SSTBC protocol [97] is a combination of LEACH-C, where the sink makes all the decisions for the network in a centralized way, and GAF, where the nodes follow a sleep schedule to save energy and extend the network lifetime. Like most of the protocols we have reviewed up to this point, SSTBC performs over an iterative concatenation of set-up and steady-state phases. During each set-up phase, the sink: (i) virtually divides the network into square subregions and determines which nodes will be kept awake based on their residual energy (like in GAF); (ii) defines the number of clusters and groups nodes seeking for an even distribution over the clusters (the protocol does not specify how this assignment is conducted); (iii) selects the CH for each cluster as the member node with largest ratio of residual energy to distance to the sink; and finally (iv) builds the multi-hop paths from each node to its CH (frame (d) of Figure 2.4) by constituting a minimum spanning tree[2] at each cluster based on the Prim's algorithm [102]. The routing approach used to transfer the solution to the nodes is not specified.

Note that at least actions (i) and (iv) require the nodes to share their location with the sink. To this end, the nodes rely on the availability of a relative or absolute positioning system (details and references in the description of LEACH-C). The transfer of data from CHs to the sink is done in a single-hop manner (frame (a) of Figure 2.4). Since the clustering approach used in action (ii) is not specified, it is impractical to determine the technical requirements of the sink. A brief discussion for the case where the task is modeled as the k-medoids problem was offered in the examination of LEACH-C, where we concluded on the need for a node with significant processing power. The main advantage of this approach is the possibility to have a globally optimum clustering and routing scheme since the decisions are made in a centralized way. Unfortunately, centralization also represents a major vulnerability of the system since the whole network depends on the well functioning of the sink.

• ***Multi-hop Overlapping Clustering Algorithm (MOCA):*** contrary to FLOC (examined above), MOCA [103] aims at generating overlapping clusters on networks implementing multi-hop inter-cluster routing (frame (b) of Figure 2.4). Recall that two clusters are overlapping if they have at least one node in common. According to the authors, the interest in

[2]In graph theory, a minimum spanning tree is formed by the subset of links (edges) that connects all the vertices without any cycles while offering the minimum total generalized transfer cost (e.g., total number of edges or sum of their lengths).

creating overlapping clusters lies in the possibility of using the shared (or boundary) nodes as gateways for inter-cluster communication when the CHs do not have long enough communication range to send data directly to other CHs. Unlike the aforementioned protocols which perform over several set-up rounds, MOCA is executed only once. The CHs are selected similarly as in LEACH-like protocols; each node self-elects as CH with probability p by generating a random number in $[0, 1]$ and taking the role of CH if the number is less than or equal to p. Self-elected CHs advertise their role to their neighbors who, in turn, forward the message to their neighbors. These advertisement messages are forwarded at most k hops away from the originating CH, and each node receiving an advertisement becomes part of the cluster linked to the message. The nodes that receive multiple advertisements from different CHs become members of all the corresponding clusters and take the role of boundary nodes. Henceforth, the network operates based on the resulting clustering structure, which is never refreshed.

The intra-cluster routing in MOCA is multi-hop as well (frame (d) of Figure 2.4). At each cluster, the routes from cluster members to CHs are built by means of a shortest path procedure similar to RPL (revised above). The performance of this protocol, and the potential to exploit the boundary nodes in an efficient manner, strongly depend on the parameters k and p, which are supposed to be given by the user. If those two values are not well chosen, the network might end up with excessive or too few boundary nodes. Unfortunately, the authors do not provide any intuition or guideline on how to properly set them. Furthermore, no details are provided on how inter-cluster communication could be handled in order to exploit the availability of boundary nodes.

• ***Protocols for settings with prefixed CHs:*** up to this point, all the cluster-based protocols that we have examined involve a methodology for CH selection. Indeed, some of them exclusively focus on the selection of the CHs and further constitution of the clusters, offering very simplistic or no indication on how to build the paths for data transfer. We now focus on three proposals where the sensor nodes functioning as CHs are predefined, and most of the attention is put in the load balance among them and the multi-hop routing within each cluster: ***Largest-Traffic-First (LTF)*** [104], ***Partition-based Network Load Balancing (P-NLB)*** [105] and ***Controlled potential-based routing (CPBR)*** [106]. In LTF the nodes acting as CHs are the neighbors of the sink. The primary goal of that protocol is to use those nodes as evenly as possible in order to keep them alive, and thus keep the sink highly accessible, for an extended period. P-NLB and CPBR assume a network counting on multiple sinks distributed over the network, each of which acts as a CH. In this case, the load balance is intended to prevent congestion issues and extend the network lifetime by distributing the traffic over the network. As briefly stated above, all three proposals assume that the CHs are chosen before the protocol is called. Note that the term CH is not explicitly used in the introductory article of any of these protocols (i.e., [104], [105] and [106], respectively).

However, for the sake of comparison, in the remainder of this discussion we stick to the term CH to refer to the nodes functioning as leaders in the three studies.

In ***LTF*** [104], the clusters are formed by setting up multiple routing trees, each rooted at one CH. It is assumed that the nodes have heterogeneous amounts of data to share with the sink. The protocol operates by turns, each time letting the cluster less loaded attach a free node to its tree. More precisely, the cluster adopts as a member the node with the largest amount of data among those reachable by any current member of the cluster. The procedure is explained at a high level with no much detail on which interactions should occur between the nodes in order to build the clusters. Nonetheless, it is presumable that the nodes are required to centralize, at each turn, the information on the data load of their free neighbors. It is also presumable that some node must be responsible for selecting and announcing the cluster allowed to expand and the node to be added to it. Note that this type of centralized scheme, where the decisions to make involve simple logical comparisons, is not as difficult to put in place as the ones proposed in protocols like LEACH-C and SSTBC (both examined above), where complex optimization tasks are required to be performed by the sink. The main advantages of this method could indeed be the simplicity of the operations made by the nodes and the possibility to achieve load balance at the CHs (to some extent) using just local communication. Its main drawbacks might be the possibility of some trees blocking the growth of the others (limiting the ability of the algorithm to produce balanced clusters) and the lack of a feedback mechanism allowing the system to gradually improve and reach a better solution.

P-NLB [105] is also based on the constitution of routing trees. In this case, each node starts by joining the cluster of its nearest CH and making the corresponding CH aware of this decision. Then, each CH counts the number of nodes that joined its cluster and shares this information with the whole network. Each node uses the received information to decide whether to stay in the current cluster or switch to a nearby one. More precisely, the nodes decide on the possibility to join the lowest loaded cluster so far. As a decision rule, any node with at least one neighbor belonging to the lowest loaded cluster joins this last and picks one of such neighbors as its parent (e.g., the one with highest residual energy). Then, each node advertises itself to its current CH and the process starts again. This is repeated multiple times until some stopping condition is reached (e.g., degree of convergence or number of iterations). Similarly as in LTF, an important advantage of P-NLB is its ability to search for globally optimal solutions in terms of performance of the whole network without demanding any node to perform complex processing tasks. Another advantage is the feedback mechanism it uses to progressively improve the clustering solution. The main weakness of this approach is the way the correction is made each time, based on the results of the previous iteration; the action of moving to the lowest loaded cluster all its neighbor nodes is very unlikely to resolve the imbalance. In fact, the larger the number of neighbor nodes this cluster has, the more likely is

that it will become overloaded and one of its nearby clusters will be the one the lowest loaded in the next iteration. In some instances the algorithm will get stuck throwing the nodes back and forth from one cluster to the other without being able to reach convergence. In those situations, the algorithm would only be able to stop based on the number of iterations done, which might anticipate an unsatisfactory result in terms of distribution of nodes among the clusters.

CPBR [106] is a gradient-based protocol (a.k.a. potential-based protocol), meaning that the routes followed by the packets are specified by a coefficient maintained by each node, which estimates the cost to reach each CH from its location. The collection of gradients of all the nodes is often said to form a cost field. By design, the CHs are the nodes located at the bottom of such a field, holding the lowest cost coefficients. The cost field is systematically updated to progressively learn efficient ways to reach a CH. In this type of protocol, any node having information to share broadcast its data packet along with its gradient; then, only the neighbors with a lower gradient keep moving the message forward through the network. As a differentiating factor from other gradient-based protocols, CPBR seeks to balance the load of the CHs. To achieve this, the gradient of the CHs is updated after each round of data transfer to take into account the packets received by each CH. The protocol assumes that a higher-level node computes the gradients of the CHs in such a way that each CH gets a gradient proportional to the number of packets it received so that the CH that received less packets is assigned the largest gradient and vice versa. Once the gradient of the CHs has been updated, it is shared with their neighbors in order to trigger the updating process in them. Each node updates its gradient based on the value of its own current gradient and of that of all its neighbors. This process seeks to give lower gradients to the nodes leading to CHs that were scarcely used in previous rounds. Once the neighbors of the CHs have updated their gradient, they share this information with their neighbors in order to repeat the process until the whole cost field has been refreshed. This protocol shares with LFT and P-NLB the advantage of pursuing globally efficient solutions without requiring the execution of intricate tasks at any node. It also shares with P-NLB the advantage of implementing a feedback mechanism that allows the system to gradually improve the solution. An important drawback of this protocol is the excessive replication of packets over the network resulting from the forwarding rule where all the neighbor nodes with lower gradient than the sensing node transfer the packet. This results in multiple copies of the same packet reaching different CHs and even the same CH. A second major deficiency is the difficulty it has to maintain an ideal solution once it has been reached. This happens because the field update takes place even in a round where such an ideal solution has been obtained.

2.3 Routing in settings involving MULEs

• ***MobiCluster:*** this protocol, introduced in [107] proposes the use of public transport vehicles to collect information from an isolated network. They focus on vehicles following fixed routes (such as buses), which would have contact with the same set of nodes each time they visit the network. Under this setting, MobiCluster presents a methodology to systematically move collected data from all over the network to the nodes that will eventually communicate with the MULE.

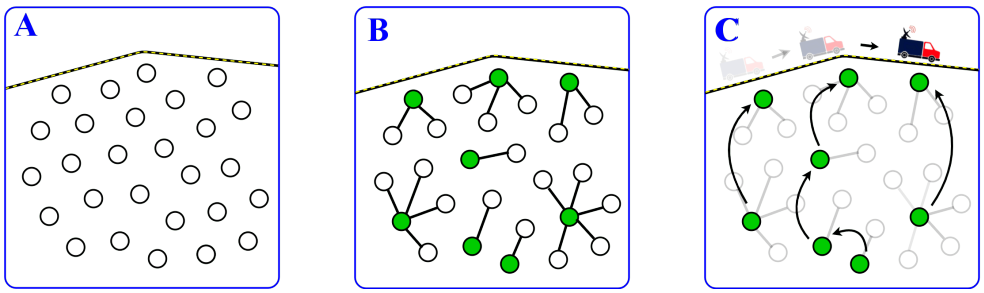


Figure 2.9: MobiCluster protocol.

Motivated by the possibility of balancing energy consumption and reducing the amount of traffic by means of data aggregation, the protocol starts by dividing the network into clusters as illustrated in Figure 2.9. To this end, the CHs self-elect based on a fixed probability (as in LEACH) and then, each node joins the cluster of its nearest CH. Every node is supposed to be able to reach its CH in a single hop. Once the clusters are formed, the protocol chooses the nodes that will be used as conveyors to forward the information aggregated at the CHs to the MULE. Those nodes receive the name of Rendezvous Nodes (RNs). It is assumed that the identification of potential RNs takes place before the activation of the protocol, during a first visit of the MULE to the network, where it moves along its fixed trajectory broadcasting periodic beacons. Then, the definitive RNs are selected by the CHs located in the periphery of the MULE's path, which pick as many RNs as they can while ensuring that their communication ranges do not overlap. After the RNs are chosen, the network enters a steady-state phase where data is collected by the sensor nodes and periodically transferred to the MULE. Inter-cluster communication is done strictly over the CHs in a multi-hop fashion, picking always as next hop the CH with lowest cost in number of hops to reach a cluster in the periphery of the MULE's path. The clustering structure is periodically refreshed to rotate the roles and prevent premature battery depletion at some nodes.

As most cluster-based protocols, MobiCluster strongly relies on the possibility to perform data aggregation at the CHs. However, as it is also the usual, it does not specify the mechanism of data aggregation and the synchronization scheme that would be used to support this proposal. Indeed, the simulations presented in [107] and [108] merely assume that the

data is reduced in some percentage when they pass through the CHs, at the cost of some given energy loss. Both quantities are arbitrarily fixed by the authors, which hinders the possibility to make conclusions about the performance of the protocol.

An important differentiating factor between the reviewed cluster-based protocols and MobiCluster is that this last introduces a new type of node responsible for data transfer to the MULE. As will be made evident in the remainder of this section, this is a usual feature in protocols considering MULEs. Indeed, most of them pick nodes in the periphery of the MULE's trajectory for this role, as in MobiCluster.

• ***Minimum Load Set algorithm (MLS):*** as MobiCluster, MLS [109] considers a MULE moving along a fixed trajectory and always getting contact with the network through the same set of nodes. In this case, MLS implements a tree-based structure instead of a cluster-based one, with the tree rooted at some arbitrary point over the segment of the MULE's trajectory where the contact with the network takes place. MLS implements a variation of Dijkstra's algorithm [65] to grow the tree until all the nodes have been attached. In this variation, one node is attached to the tree at each iteration by minimizing the energy demand of the most loaded node as a function of the addition of any candidate node through any candidate link. In short, the algorithm minimizes, at each iteration, the maximum energy demand of a node as a function of the link and node to be added to the tree. The result is a spanning tree[3] for the graph using the (dynamic) maximum energy demands as weights of the links.

The main advantage of this protocol is the simplicity of the computations that need to be made at each iteration; a property directly inherited from Dijkstra's algorithm. As mentioned in the examination of LTF, centralized schemes requiring simple computations at the central node are fairly plausible. Regrettably, this protocol also shares with LTF the deficiency that some trees might block the growth of the others, limiting the ability of the algorithm to minimize the maximum energy demand. This is a consequence of the fact that the decisions made at each iteration are made only based on the current state of the network and thus ignore the potential implications on the future development of the trees.

• ***Singh & Kumar, 2018 (SK18):*** the protocol introduced in [110] considers multiple isolated subnetworks, each attended by a dedicated MULE. At each subnetwork, the MULE holds a given position for some time and then switches to a new position where the nearby nodes have relatively high residual energy. This process is repeated periodically seeking to balance the battery consumption over the subnetwork. Each time a MULE takes on a new position, it triggers the creation of a routing tree rooted at itself, which once constructed is used by the nodes to forward the sensed information to it (see Figure 2.10). Although

[3]In graph theory, a spanning tree is formed by any subset of links (edges) that connects all the vertices without any cycles. Not to be confused with a *minimum* spanning tree, which in addition, offers the minimum total generalized transfer cost.

not explicitly said, the algorithm used for the constitution of the tree corresponds to the RPL protocol (examined above), which provides the shortest path tree or DODAG (see the analysis of RPL for details). After receiving information for a while at some point, the MULE implements an aggregation technique to compress the data and then forwards it to the sink through a multi-hop path formed by MULEs from the other subnetworks. Apart from this last component, SK18 could be seen as an application of RPL to a setting involving multiple isolated networks. Compared to MobiCluster and MLS, SK18 adopts a far more intrusive approach for the modeling of the MULE. Having a dedicated MULE per subnetwork and making it remain connected to it the whole time might not be a feasible solution in many real situations.

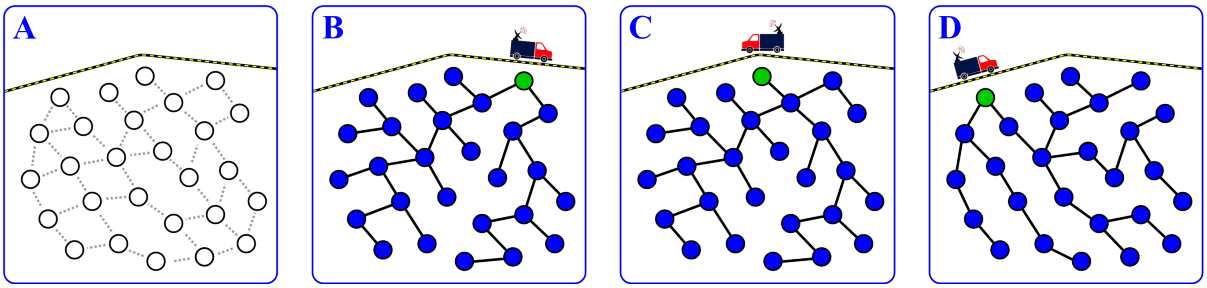


Figure 2.10: SK18 protocol.

• ***Somasundara et al., 2006 (Sea06):*** similarly to MobiCluster [107, 108] and MLS [109] (examined above), Sea06 [111] considers a MULE traveling along a fixed trajectory and always getting contact with the same set of nodes. In contrast to the mentioned protocols, Sea06 has the assumption that the MULE is able to modify its speed and stop along its path in order to better attend to the needs of the network. The authors describe their setting in a way that closely matches the analogy we described in the introduction to cluster-based routing (Section 2.2), where the nodes that systematically get contact with the MULE are seen as CHs and the remaining nodes group themselves forming clusters (see Figures 2.4 and 2.5). The protocol starts with the MULE doing a first tour. Along its way, the MULE broadcasts periodic beacons to let the nodes in the periphery of its path recognize themselves as CHs. This first visit also triggers the constitution of the clusters. To this end, each CH starts a shortest path tree rooted at itself which is completed by means of the RPL algorithm, described above. After the tree of every CH has been completed, each node picks a cluster by identifying the nearest CH reachable by the routes included in the RPL trees. This step defines the final clusters that are used for transfer of sensed data. Once the clusters are defined, each node sends a packet to its CH so that each CH can count the number of nodes linked to its cluster. Figure 2.11 illustrates an example of cluster formation using Sea06. During a second visit to the network, the MULE picks up the information about the size of each CH, which it then uses to determine the extent of time that it should spend parked

beside each CH. Data sensing and further transfer to the MULE is implemented from that moment onward.

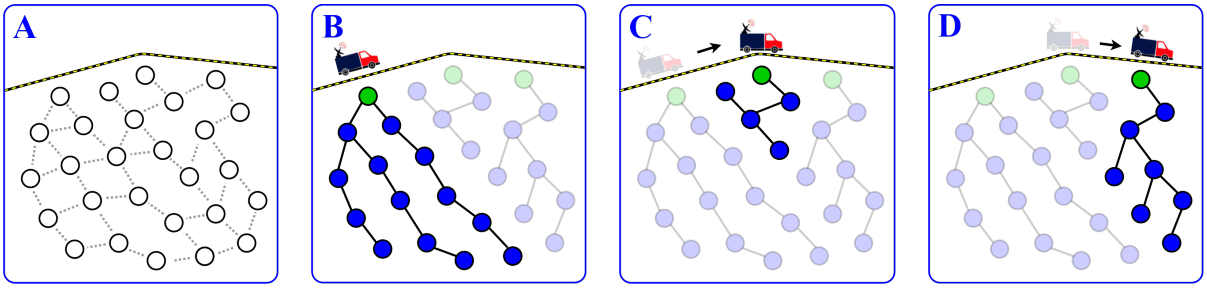


Figure 2.11: Sea06 protocol.

Sea06 shares the advantage with MobiCluster and MLS that the MULEs are not required to adjust their trajectory to better serve the functioning of the network. Although this corresponds more to a matter of assumptions rather than methods, the perspective of the mentioned protocols make them less restrictive for application in realistic scenarios. On the other hand, these protocols use multiple nodes as contact points with the MULE instead of only one as in SK18, which might be advantageous in at least two manners: (i) it helps to distribute the energy requirement over the network; and (ii) it helps prevent some contact nodes from getting overly loaded to the point of not being able to transfer the stored data to the MULE while the less loaded nodes even have idle time of contact with the MULE which is not used to transfer any data. Unfortunately, none of the strategies adopted in these three protocols ensure perfect load balance among the contact nodes, leaving place for improvement in these areas.

• ***Distributed Storage Management (DSM):*** the DSM protocol, introduced in [112], considers the case of a single MULE making opportunistic visits to the network, but always getting contact with the same single node. In addition, the contact time is assumed to be uncertain for the network and also to vary from one visit to the other. The authors further consider a setting where packets of sensed data are labelled by a priority index, and the aim is to deliver as many packets as possible during the time in contact with the MULE, starting from those with higher priority and moving towards the ones with lower priority. In addition, the nodes are assumed to have limited and fairly narrow storage capacity, making it impossible for the contact node to hold alone all the data collected by the whole network during the period between consecutive visits of the MULE. Seeking to alleviate the problem of limited storage, the protocol implements a Buffer Area (BA) located close to the contact node, enclosing a set of nodes allowed to store data until the next visit of the MULE. More precisely, the BA is defined as the region containing all the nodes located up to k hops away from the contact node. Each time a node generates a new data packet, it is sent to the BA and allocated at some specific node based on its priority index. The routing approach

followed to move the packet from the source node to the BA is not specified. On the other hand, the procedure for the selection of the node to store the packet inside the BA is defined by a set of basic decision rules which lastly put the packets with largest priority in the nodes closest to the contact node and vice versa. Once the storage capacity of the BA nodes gets exhausted, the BA nodes start dropping one of the packets with the lowest priority index each time a new packet reaches the BA.

DSM results an interesting proposal for situations involving priority indices, limited storage capacity and uncertain time in contact with the MULE. An interesting fact of the proposal is that any of the three assumptions could be easily removed without having to make changes to the methodology. However, as we remove those assumptions, it starts becoming more relevant the proper definition of the routing protocol using to move sensed data from the source nodes to the contact node, which as mentioned before, is a critical detail omitted by the authors. The assumption of a single contact node is also somehow limiting since in many situations as in our ASAS setting, the MULE actually gets contact with multiple nodes along its path, and by considering those multiple contact points one could reach a better energy distribution over the network and a more efficient use of the time in contact with the MULE.

2.4 Routing based on advanced optimization techniques

The routing protocols examined in the previous sections are primarily based on empirical decision rules which do not guarantee optimal performance. Most of these simplistic decision schemes arise from the intention of keeping the protocol decentralized. While relatively simple routing problems have been successfully adapted to the WSN domain (e.g., the shortest path three problem addressed by RPL), most variations proposed in the examined literature result in extensions of well known NP-hard routing, clustering and grouping problems such as the Hamiltonian path problem [113], k-medoids [87] and bin packing [114]. Those problems are indeed too complex to be optimally solved in plausible time for real size instances. Thereby, it is natural to find various empirically founded strategies in the literature. Nonetheless, consistent effort has been made for over 40 years to develop a whole set of metaheuristic optimization techniques able to provide high quality solutions to some of these difficult problems in reasonable time [115, 116]. In this last section of protocol review, we briefly explore *Genetic Algorithms* [117, 118, 119] and *Ant Colony Optimization* [120, 121], two of the most popular optimization techniques adopted in the WSN domain. Rather than conducting a detailed examination of the algorithm proposed by each author, we seek to identify the main advantages and drawbacks of each method and determine which could eventually fit better the needs of the ASAS application considered in this thesis.

Overview of existing techniques

• ***Genetic Algorithms (GAs):*** inspired by Charles Darwin's theory of evolution by natural selection [122], GAs [117, 118, 119] constitute one of the most renowned and widely studied metaheuristics [123] mainly due to its interpretability and remarkable record of successful performance in a variety of contexts (see e.g., [124], [125], [126] and [127]). According to Darwin's postulate, populations evolve over generations through a so-called process of natural selection, where the fittest individuals are more likely to survive, reproduce, and leave their hereditary traits to future generations. In the optimization context, each potential solution to the problem is regarded as an individual, whose fitness is given by a pertinent performance metric (e.g., the length of the route in a shortest-path problem). The algorithm performs in an iterative fashion, constituting each time a new set of solutions (population) which are systematically evaluated to retain the ones of highest quality. Each population is formed by four types of individuals: (i) *mutations*, denoting slight random tweaks of previous solutions; (ii) *crossovers*, resulting from combination of pairs of previous solutions; (iii) *elite* individuals, corresponding to the best k solutions explored so far; and (iv) new individuals. Thanks to this composition, the overall quality of the population is expected to improve from one generation to the other. The algorithm is typically left run until a resource-, or convergence-oriented stopping condition is reached. The best solution explored so far is taken as the approximate optimal solution to the problem.

In the WSN domain, GAs have been applied to some of the settings discussed in the previous sections. ***ROS-IGA*** [128] is a flat, multi-hop routing protocol, which encodes the path connecting each node to the sink as a vector of node IDs. A generalized cost involving the length of the path, energy consumption, sum of residual energy of visited nodes and total delay is used as a fitness function.

LEACH's philosophy also made its way up to this segment of the literature with ***LEACH-GA*** [129]. As in EEHC (examined above), the aim is to find the optimal value for the probability p of a node self-electing as a CH, in terms of the total energy spent during one round of data collection. In this case, every node is assumed to be 1 hop away from its CH (k=1). The authors first develop an analytic equation for p supported on the following assumptions: (i) the sensing field is square; (ii) the nodes are evenly distributed over it; (iii) the sink is placed at the center of the sensing field; (iv) all the clusters are circular; and (v) each CH is placed at the center of its cluster. Then, the authors propose a GA able to find an approximate optimal solution for p without resorting to any of the listed assumptions.

Other two cluster-based protocols, ***GABEEC*** [130] and ***EAERP*** [131], implement GAs to explicitly select the most appropriate CHs, instead of optimizing the probability of CH self-election as in LEACH-GA. GABEEC aims at minimizing the sum of distances from the member nodes to the CHs and from the CHs to the sink. EAERP minimizes the energy spent

in communication, instead. The solution of both algorithms is encoded as a binary vector where the i-th element takes a value of 1 if the i-th node is selected as CH and 0 otherwise. Every node is assumed in both protocols to be 1 hop away from its CH.

EHGUC-OAPR [132] is also a cluster-based protocol which, in addition to delivering the selection of CHs (as GABEEC and EAERP), matches each node to a CH, thus forming the clusters. The solution is provided by a GA which encodes the solution as an integer vector where the i-th element gets the ID of the CH assigned to the i-th node. Once the clusters are defined, the protocol uses a greedy algorithm to build the multi-hop paths connecting the CHs to the sink. In all the five cited protocols, the GA must be first resolved in a central node (typically the sink) and then the solution should be transmitted to the nodes.

A notable attribute of GAs is their adaptability to diverse optimization problems. The key of this versatility lies in the fact that plenty of individual encoding structures, mutation operators and crossover operators have already been explored in the literature (see e.g., [133], [134] and [135]), and it turns out relatively easy to find elements of this kind that fit well to almost any type of optimization problem. On top of that, the trade off between solution quality and processing time is often well managed by this type of algorithm thanks to its appropriation of the principles of exploration and exploitation [136, 137]. By these principles, the search of the solution starts with a fairly wide screening of the solution space and gradually converges to a reduced zone holding high quality solutions. The main drawback of GAs for application in WSN, is that they are inherently centralized. The mutation and crossover operations involve variations to previous solutions and combinations between pairs of those, which is not clear how to do in a distributed manner. The required centralization scheme is thus one where the whole algorithm should be run at a central node, and not just some basic operations as in LTF, P-NLB and MLS (examined above). This implies the need for a central node with particularly high storage capacity, battery autonomy and processing power. As mentioned before as a drawback of other centralized protocols of this type (e.g., LEACH-C and SSTBC), a system depending on such a special kind of node is fairly vulnerable since any failure on the node leaves the network inoperable.

• ***Ant Colony Optimization (ACO):*** just a couple years after the introduction of GAs, ACO [120, 121] appeared as a precursor of swarm intelligence, a branch of computational intelligence founded on the collective behavior of decentralized, self-organized systems [138]. This algorithm is based on the process followed by real ants while foraging for food. These insects use pheromone trials to let themselves and other members of the colony be aware of paths connecting the nest with food sources. The exploration starts with the ants randomly distributing themselves in different directions around the nest. Once a food source is found, the ants start taking pieces of it back to the nest. The ants using the shortest paths complete the trips faster, making stronger the pheromone trail in them. As the shortest paths start

getting considerably larger pheromone levels, a larger proportion of the ants start using them. More and more ants gradually switch to the shortest path until the whole colony converges to it. Although raised in this routing setting, ACO is by no means limited to resolution of routing problems. Independently of the nature of the problem, each potential solution is regarded as the path covered by one ant. The algorithm performs by iterations, each time releasing a group of ants that traverse the decision network until completing a new set of solutions to the problem. Once the whole group has made a feasible solution, the quality of those is evaluated and the k best ants are sent to increase the pheromone level in the links pointing out to the decisions they made. As in GAs, the whole procedure is repeated until a resource-, or convergence-oriented stopping condition is reached. The path with the highest performance metric is thus taken as the approximate optimal solution to the problem. A widespread variation of ACO in the WSN domain is one where each node releases an ant seeking to find a path to the sink, and a feasible solution is constituted by the whole set of paths built by the ants generated all over the network.

The ACO technique has gotten significant attention from the WSN community. On the side of flat routing, ***EEABR*** [139] and ***EPACOR*** [140] implement this algorithm to build the path from one source node to the sink while minimizing a weighting of path length and energy spent. Both protocols can be used recursively to generalize for multiple nodes. The pheromone trail is maintained at the nodes and the fitness is evaluated at the sink each time an ant gets to it. ***EBAB*** [141], ***ACALEACH*** [142] and ***LB-CR-ACO*** [143] are cluster-based protocols which first select the CHs using some empirical decision rule or greedy approach, and then use ACO to build the multi-hop paths connecting the CHs to the sink. ***MRP*** [144] treats the event-based data collection setting by borrowing ideas from Directed Diffusion and the cluster-based protocols examined above. Each time an event is captured by a group of nodes, a CH is elected among them by means of a greedy criterion. Then, ACO is used to build several multi-hop paths from the CH to the sink, which are used by the packets in a probability-based fashion. As in Directed Diffusion, it is assumed that once an event is detected, data gathering about it lasts for an extended period, allowing the network to amortize the cost of finding efficient paths. ACO-based CH selection is addressed in ***ACO-C*** [145], which aims at minimizing a weighting of the sum of intra-cluster distances and total energy spent. Every node is assumed to be one hop away from its CH. In contrast to all the other cited ACO-based protocols, ACO-C is executed in a fully centralized manner at the sink.

ACO is comparable to GAs in terms of versatility and exploration power. Both algorithms have been competing for years in a variety of domains (see e.g., [146], [147] and [148]). In the WSN context, the main advantage of ACO is its natural disposition to decentralization. The algorithm is designed to operate upon local actions to optimize for global performance. Indeed, among the above cited protocols, only ACO-C requires centralized processing at the

sink. The main drawback of ACO could be its relatively high energy demand compared to most of the empirical methods discussed in previous sections. This could be a limiting factor for (often military) applications where the nodes have extremely low battery autonomy (e.g., 0.25 to 1 J [95, 149]). Nonetheless, for most industrial and domestic applications the nodes can be provided with at least a regular pair of AA batteries with typical load of 18720 J [150, 151], ACO remains a perfectly valid method.

2.5 General synthesis and main takeaways

In this section we provide a brief discussion integrating the most relevant aspects of the four routing categories reviewed.

Flat routing: broadly speaking, these can be seen as the classical yet more simplistic group of protocols in WSN. Most of them are intended to just make the packets reach their destination, regardless of the efficiency of that process. Various protocols in other categories have used a flat routing protocol as basis and incorporated some notion of efficiency. This is the case, for instance, of SSTBC, which develops upon GAF and seeks energy efficiency and extended network lifetime. Among the reviewed flat routing protocols, RPL stands out with its ability to build shortest path trees in a distributed manner. Shortest path trees are valuable assets for routing in WSN, and the possibility of building them in a distributed manner keeps wide the scope of the protocol. It is thus natural to find various protocols in the other categories using RPL as part of their proposal.

Cluster based routing: protocols in this category involve cluster formation as a task in addition to that of routing already addressed by flat routing protocols. The motivation for clustering comes from the possibility of reducing energy consumption through data aggregation at the CHs. The revised articles present interesting techniques for CH selection and matching of nodes to CHs. Roughly, half of the protocols assume that the nodes are one hop away from their corresponding CHs, and also that the CHs are one hop away from the sink. Most of the remaining protocols perform routing based on shortest path trees like the ones delivered by RPL. Although decentralized schemes stand out as the favored choice for WSN, some protocols concerned about energy balance among clusters opt for centralization. We identified two ways of centralization: (i) *full centralization*, where the whole algorithm is ran at a central node; and (ii) *light centralization*, where only a simple action like the evaluation of the objective function is performed at a central node, and all the remaining processing functions are performed in a distributed manner by means of local actions of the nodes. Fully centralized protocols are more restrictive from the practical perspective. On the one hand, those require the central node to have superior memory and processing capabilities, together with greater battery autonomy. In addition, if the central node fails the whole network is

left inoperable. Lightly centralized protocols, in contrast, allow any regular node to act as a central node. Furthermore, the role of the central node can be assumed by any node at any time out of processing, and even periodic backups during the optimization could be done at the neighbors in order to secure the progress done in case the central node fails in duty.

Routing in settings involving MULEs: the protocols in this category consider the scenario where communication between the network and the sink is only possible by means of a MULE. Some of these protocols assume that the MULE is controlled by the network and thus, the nodes that have contact with it are selected at convenience. Some others study a case similar to the one we have in the ASAS application, where the MULE moves over a fixed trajectory, having contact with the same set of nodes at each visit. In these protocols, it is assumed that the contact nodes can be identified by the network during a first visit of the MULE where it passes beside it broadcasting beacons. On another issue, some protocols assume that the MULE has contact with only one node at each visit, while others consider the possibility of contact with multiple nodes along its trajectory. The assumption of a single contact node is considerably limiting since the strategic use of multiple contact points could help the network to transfer larger amounts of data to the MULE at each visit. The protocols that consider multiple contact nodes require every other node to choose one of them as gateway to the MULE. The simplistic rule of choosing the nearest contact node as gateway is used by all these protocols. Unfortunately, this strategy gives place to load imbalances among the contact nodes, which could end up in the most loaded nodes not being able to transfer the stored data to the MULE while the least loaded ones finish data transfer with time to spare. In addition, this imbalance makes the battery of the contact nodes drain asynchronously, causing the premature reduction of contact nodes.

Routing based on advanced optimization techniques: the routing protocols reviewed in the other three categories mostly apply empirical decision rules that do not guarantee optimal network performance. The ones reviewed in this last category implement two renowned metaheuristic optimization techniques which have gained considerable popularity in the WSN domain: Genetic Algorithms (GAs) and Ant Colony Optimization (ACO). Both methods have proven their ability to provide high quality solutions to a variety of routing problems in WSN. However, ACO has the advantage of being naturally disposed for distributed implementation, which allows optimizing for global network performance through local actions of the nodes, while GAs seem to require a (fully) centralized scheme.

◇ ***Taxonomic classification of the reviewed protocols:*** in Table 2.1 we provide a classification of the 41 reviewed routing protocols based on features which are relevant both, for this thesis and also for WSN routing in general. The last line of the table, in blue, indicate the characteristics that we seek to incorporate in our protocol. As can be seen, no protocol in the literature has yet covered that reference configuration.

N°	Routing protocol	Scheme				Nature			Multi-hop		Objectives		Tested network topology				Simulator		
		FL	CL	MU	OP	FC	LC	DC	Y	N	ER-MH	CH-LB	GR	RA	NS	OT	NS	SM	SP
1	Flooding [54]	x						x	x				-	-	-	-	-	-	-
2	Gossiping [54]	x						x	x				-	-	-	-	-	-	-
3	SPIN [54]	x						x	x					x					x
4	Directed Diffusion [61]	x						x	x		x		x	x					x
5	Rumor Routing [63]	x						x	x					x					x
6	GEAR [64]	x						x	x		x			x				x	
7	GAF [57]	x						x	x		x			x					x
8	RPL [58]	x						x	x		x		-	-	-	-	-	-	-
9	LEACH [78]		x					x	x					x					x
10	LEACH-C [83]		x			x				x				x					x
11	LEACH-EP [91]		x				x			x				x					x
12	LEACH-R [82]		x					x		x				x					x
13	LEACH-DT [92]		x					x		x				x					x
14	TEEN [93]		x					x		x				x					x
15	APTEEN [94]		x					x		x				x					x
16	PEGASIS [95]		x					x	x		x			x			x		
17	EEHC [77]		x					x	x		x			x			x		
18	FLOC [98]		x					x		x			x						x
19	EHCA [96]		x					x	x		x			x					x
20	SSTBC [97]		x			x			x		x	x		x					x
21	MOCA[103]		x					x	x		x			x					x
22	LTF [104]		x				x		x			x	x				x		
23	P-NLB [105]		x				x		x		x	x		x					x
24	CPBR [106]		x				x		x			x		x				x	
25	MobiCluster [107]		x	x				x		x					x		x		
26	MLS [109]	x		x			x		x		x			x					x
27	SK18 [110]	x		x				x	x		x			x					x
28	Sea06 [111]		x	x				x	x		x			x					x
29	DSM [112]	x		x				x	x					x				x	
30	ROS-IGA [128]	x			x	x			x		x				x				x
31	LEACH-GA [129]		x		x	x				x				x			x		
32	GABEEC [130]		x		x	x				x				x				x	
33	EAERP [131]		x		x	x				x				x					x
34	EHGUC-OAPR [132]		x		x	x			x		x			x					x
35	EEABR [139]	x			x		x		x		x			x					x
36	EPACOR [140]	x			x		x		x		x			x			x		
37	EBAB [141]		x		x		x		x		x			x			x		
38	ACALEACH [142]		x		x		x		x		x			x					x
39	LB-CR-ACO [143]		x		x		x		x		x			x					x
40	MRP [144]	x			x		x		x		x			x					x
41	ACO-C [145]		x		x	x				x				x					x
Target setting			x	x	x		x		x		x	x	x	x		x			x

Table 2.1: Taxonomy of revised WSN routing protocols. For the sake of exposition, the table adopts to the following convention: • Scheme: flat (FL), cluster-based (SM), involving MULEs (MU), based on advanced optimization (OP); • nature: fully-centralized (FC), lightly-centralized (LC), decentralized (DC); • Suitable for multi-hop: yes (Y), no (N); • Performance objective: efficient multi-hop routing (ER-MH), CH load balance (CH-LB); • Tested network topology: grid (GR), random (RA), not specified (NS), other (OT); • simulator: not specified (NS), self-made (SM), specialized (SP). The tested network topology and simulator are marked with a dash symbol (-) for Flooding and Gossiping since we did not find a seminal paper for those protocols. The reference [54], provided in both cases, corresponds to the seminal paper of SPIN, which offers a good description of these protocols. Similarly, the RPL is marked with a dash symbol in those categories since the official document introducing it does not show any simulation.

2.6 Open research challenges

Addressed in this thesis

In accordance with the findings of our literature review, below is the list of challenges addressed in this thesis.

1. ***Incorporate load balance between contact nodes in the problem with MULEs:*** none of the reviewed protocols considering MULEs accounted for this issue. Balance between contact nodes is expected to allow for better use of the time in contact with the MULE and a more homogeneous process of battery depletion. The solution to this problem is expected to be useful also for load balance in cluster-based scenarios not involving MULEs.

2. ***Integrate efficient routing and load balance in a joint optimization:*** in almost every instance, the ideal set of routes depends on the configuration of the clustering structure and likewise, the ideal clustering structure depends on the routes selected. Both problems are linked and thus, should be resolved together in order to achieve the best performance. None of the reviewed protocols, in any of the four categories, combine efficient routing and load balance in a joint optimization procedure. Depending on the distribution of nodes over the sensing field, one could think of instances where these objectives are in conflict (i.e., the optimal solution in terms of cluster balance does not match the optimal one in terms of routing). The solution approach should be able to deal with this trade-off.

3. ***Generalize the technique for a broader class of network topologies:*** almost all the revised articles either assume a random uniform distribution of nodes over the sensing field or use this type of distribution to evaluate the performance of the protocol. One might naively think that the random distribution is the most generic case and thus it proves the suitability of a method for any other topology. Nonetheless, if the distribution is particularly *uniformly* random, the clustering balancing problem might indeed turn simpler since a simple symmetric partition of the sensing field would provide a reasonable solution. For most of the proposed methodologies, the results and conclusions are likely to change if different network topologies like the triangular and hexagonal ones were used or if uneven distribution of nodes over the sensing field were considered (e.g., due to the presence of obstacles). In settings involving MULEs and where the contact nodes are fixed (e.g., MobiCluster or our ASAS application), this type of topology tends also to reduce the conflict between the cluster balancing and routing problems since the routing towards the nearest contact node already gives a reasonable clustering solution. The ability to deal with a more general class of network topologies should be incorporated into the protocol.

▷ ***Remark:*** *Overall,* ***distributed*** *or* ***lightly centralized*** *solutions must be favored in order to keep broad the scope of the protocol.*

Beyond this thesis

Addressing the above three challenges and accounting for the particularities of the ASAS application is already a serious duty. The following are two challenges that fall off the scope of this thesis but constitute interesting subjects for complementary research.

4. ***Consider heterogeneous data loads at the source nodes:*** apart from LTF, no protocol in our review considers heterogeneous data loads. These might appear, for instance, in scenarios of event-based collection or if the nodes use different collection rates or record different variables depending on their location on the sensing field. Load balancing in such a case is expected to be even harder than in the homogeneous case. The impact would be significant on the balance of energy consumption in cluster-based protocols, and the efficient usage of the contact time in problems involving MULEs.

5. ***Explicitly incorporate data aggregation in the protocols and their validation:*** the great majority of cluster-based protocols rely on the possibility of doing significant reduction in the amount of traffic by means of data aggregation at the CHs. Nonetheless, none of the reviewed articles went further than assuming that the number of packets was decreased in some (arbitrary) proportion at the CHs at the cost of an additional (also arbitrary) energy spent. The oversimplification of the aggregation process has led to ignore some of the practical issues to consider like the coordination between nodes necessary to make the aggregation process function properly once implemented in the network. This and many other important subjects linked to aggregation are totally neglected in the revised articles.

To the best of our knowledge, this thesis is the first study addressing challenges 1, 2 and 3 listed above. Although challenges 4 and 5 are not fully integrated in our proposal, in Section 4.5 we provide a thorough technical discussion on how our method could be extended in those two directions.

Chapter 3

WSN-based monitoring for ASAS: modeling and initial solutions

The chapter in brief

In this chapter, we present our initial steps in the modeling and resolution of the WSN routing problem raised from the ASAS context introduced in Chapter 1. This work was published in the proceedings of the International Conference on Ad-Hoc Networks and Wireless (2019) [46]. First, in Section 3.1 we offer a technical description of our data routing problem, resulting from the abstraction of the key features from the ASAS context. Having defined the problem, in Section 3.2 we move forward with the development of two routing protocols for its resolution, which are both founded on routing philosophies previously adopted in some of the studies reviewed in Chapter 2, involving systems assisted by MULEs. We evaluate the performance of these two protocols through simulation in Section 3.3. Our aim is to determine the potential of each method when applied for the specific parameters of the ASAS application (e.g., data gathering rate and time between visits of the MULE). The discussion provided in Section 3.4 closes the chapter with some ideas of improvement and extension which motivate the developments presented in the next three chapters of the thesis.

Contents

3.1 System modeling

In this section we move from the concept of WSN-based monitoring presented in Chapter 1 to the technical definition of the data routing problem derived from it. The scheme presented in Figure 3.1 illustrates the definition of a routing instance, departing from an ASAS monitoring setting comprised by following elements:

(i) ***A segmented WSN (S-WSN):*** each subnetwork is composed of static battery-powered sensor nodes with limited communication range. The expected battery lifetime is of 1 to 5 years depending on the usage; thus, energy must be managed efficiently but is not a highly constraining factor (see Section 1.3.2). Communication within each subnetwork is performed under a multi-hop scheme.

(ii) ***A set of fixed-trajectory MULEs with unknown schedule:*** the MULEs are service vehicles traveling over predefined trajectories in regular duty. The communication system has no control on their behavior. Visits of the MULEs to the subnetworks occur as part of the usual transit over the airport for support of aircraft operation. The contact between subnetworks and MULEs is done through the set of sensor nodes located closer to the path followed by these last. Those nodes receive the name of subsinks; they identify themselves during a preliminary discovery trip performed by the MULEs and maintain this role for the rest of the operation. The MULE never does partial visits where only some of the subsinks reach contact. The MULEs are not expected to stop by the side of the subnetworks to facilitate the collection of the stored data.

(iii) ***An isolated sink reachable by the MULEs:*** the communication range of the nodes in our routing setting is not enough to reach the sink directly. Thus, the subnetworks fully rely on the MULEs to forward gathered data to the sink.

For us, each routing instance is defined by a set of sensor nodes (sources) seeking to reach the MULE (destination) using a multi-hop path (bottom frame in Figure 3.1). The path for each node must pass through one subsink playing the role of ***gateway*** between the subnetwork and the MULE. We refer to the selected subsink by a node as its gateway subsink ***(G-subsink)***. Two different nodes are allowed to use different subsinks. Since contact with the MULE occurs in an opportunistic way, the routing protocol used at each subnetwork must emphasize on the efficient use of the time in contact for the transfer of data stored at the sensor nodes. We focus on a single subnetwork at a time; the same routing protocol is expected to be applied in every subnetwork of the system.

As specified in the description of the system offered in Section 1.3, safety-oriented data is transferred to the sink using long-range communication; thus, this data does not have to be considered in our routing process. The nodes exclusively devoted to this type of monitoring are omitted in Figure 3.1 for clearness on the definition of our data routing problem.

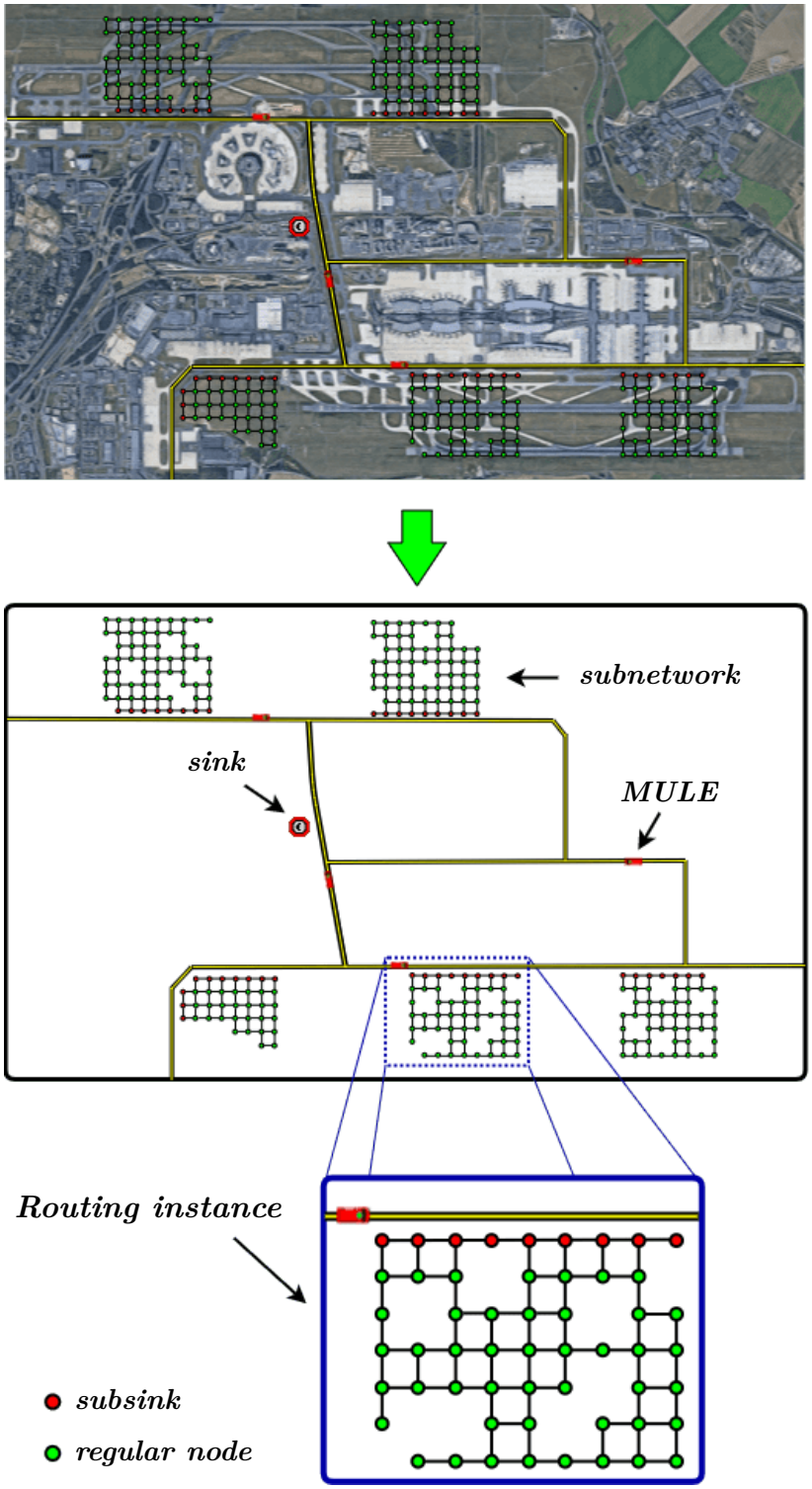


Figure 3.1: Definition of routing instances from an ASAS monitoring setting. Subnetworks displayed at scale over Paris Charles de Gaulle airport for illustration.

3.2 Initial routing solutions

In the spirit of progressive addition of complexity we start by testing two basic routing strategies previously adopted in the literature. The two Those differ in the way of selecting the subsink that each node uses as a gateway to reach the MULE. In the first strategy, the subsink currently in contact with the MULE is used as a gateway (see e.g., [110]). In the second strategy, each node uses the subsink closest to it, in terms of number of hops, as a gateway (see e.g., [111]). It is worth noting that these are the two routing strategies from the literature, devised for settings involving MULEs, that better match our routing context and our selection of technologies. Thus, we are interested in evaluating their potential for application in the ASAS context. The explanation of the two protocols offered in the literature is very little detailed; thus, in the remainder of this section we provide their formal definition as a routing protocol before proceeding with the analysis of their performance.

3.2.1 Data collection using subsink in contact as gateway

Protocol name: *MULE-directed data routing using subsink in contact as gateway (C-Sub)*

This protocol is designed to make the subnetwork operate in a reactive mode. As a first action, the system builds efficient paths connecting each sensor node to each subsink. Once the paths are done, each sensor node starts collecting and storing data. The system operates in steady state until the MULE appears. At that moment, the subsink that gets contact sends a packet to notify all the sensor nodes of its role as a temporary gateway to reach the MULE. As soon as any sensor node receives this message, it starts sending its stored data to the subsink in contact using the paths established previously. The subsink in contact, in turn, forwards as much data as it can to the MULE. As the MULE moves forward on its trajectory, it progressively reaches other subsinks, each of whom at the time of first contact notifies the subnetwork its role as the new temporary gateway to reach the MULE. Data transfer is redirected at each time to the new subsink in contact using the paths defined from the beginning. Even the subsinks which have already performed as gateway nodes are notified, enabling them to forward retained data to the current gateway. C-Sub is composed of the modules RPL, Reverse-RPL and React to Contact. These are described below.

Supporting modules

• ***IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [58]:*** this module builds the paths connecting each node to each subsink at the beginning of the operation. The paths delivered by RPL constitute shortest path trees rooted at the subsinks. Those are stored in a distributed manner over the network in the form of routing tables

that indicate each node the next hop to reach each subsink through the shortest path. RPL works for some generalized cost function that can be defined, for instance, in terms of time, distance, energy consumption, or number of hops. Our implementation optimizes for number of hops. Note that RPL in its original form only generates one shortest path tree, i.e., the routes from all the nodes to a single target point. In this module we execute RPL recurrently to obtain the shortest path trees rooted at all the subsinks. We refer the interested reader back to Section 2.1 for a more thorough explanation of RPL.

• ***Reverse-RPL (R-RPL):*** this module exploits the routes obtained from RPL to efficiently transfer messages from one subsink to the rest of the subnetwork. This includes short advertisement messages (ADVs) used to inform the subnetwork that a subsink either got contact with the MULE or lost the contact with it. The transfer of ADVs is achieved by applying a broadcast technique called Parent Flooding, proposed in [152]. The procedure starts from the root of an RPL tree (i.e., a subsink), which generates and broadcasts the ADV. Among the nodes that receive the ADV, only those which have the subsink as their parent in the RPL tree move on with the propagation and broadcast it as well. This rule is applied in a recurrent manner to keep moving the ADV deeper into the subnetwork. The process ends when all the nodes have received the ADV.

An alternative way to implement this module is by making each node send an ADV to its parent in the RPL tree. This way, every node becomes able to send messages to a specific child requiring confirmation from it. The implementation without confirmation, described in the previous paragraph, is favored in C-Sub since it is faster and there is a need to inform the subnetwork as quickly as possible about the subsink in contact with the MULE. The alternative implementation with confirmation, just described above, will be later exploited in Chapter 4.

• ***React to Contact (RTC):*** this module controls the dynamic assignment of G-subsink and determines when the nodes send and stop sending collected information. Sensor nodes start sending stored data when they receive an ADV notifying the presence of the MULE. This transfer is done using the shortest paths provided by RPL, rooted at the subsink in contact with the MULE. The nodes stop sending data when they receive an ADV notifying the loss of contact. A subsink, on the other hand, starts sending data to the MULE upon receiving a beacon message from it. It stops data transfer when a predefined period has elapsed without receiving any beacon. In both events, the subsink notifies the entire subnetwork its new status through R-RPL. If a subsink loses contact and still has information stored, it forwards it to the new subsink in contact. If the MULE has left the subnetwork, the subsink retains the remaining data until the next visit of the MULE.

3.2.2 Data collection using nearest subsink as gateway

Protocol name: *MULE-directed data routing using nearest subsink as gateway (N-Sub)*

In contrast to C-Sub, N-Sub seeks the subnetwork to operate in a proactive mode. The protocol also starts by defining efficient routes connecting every node to every subsink. Then, each sensor node selects a subsink as a gateway to forward its data to the MULE; in particular, the subsink selected by each node is the one closest to it in terms of number of hops. Note that this selection is made before the MULE gets in contact with the subnetwork. Once the G-subsinks are selected the nodes start collecting data. In N-Sub, this data is forwarded in advance to the relevant G-subsinks. When the MULE appears, the subsink in contact sends as much information as it can until losing contact. Any data retained by a subsink is stored by it until the next visit of the MULE. N-Sub is composed of the modules RPL and SNS, described below.

Supporting modules

• ***RPL:*** this module builds the shortest path connecting each node to each subsink using the RPL protocol. It works identically here as in C-Sub's RPL module. In N-Sub, the output of this module also provides the information of the number of hops required to reach each subsink from each node, which is then used in the SNS module to select the G-subsinks.

• ***Select Nearest Subsink (SNS):*** this module controls the selection of G-subsink for each node. Based on the output of RPL, each node chooses the subsink reachable in the least number of hops as its G-subsink. Each time a node gathers new information, it forwards it immediately to its G-subsink using the shortest paths provided by RPL.

3.3 Performance assessment

The experiment presented in this section pursues two main goals: (i) to assess the ability of C-Sub and N-Sub to resolve the routing problem arisen from our ASAS application, and (ii) to identify potential areas of improvement of these two protocols that could be addressed later in this thesis.

3.3.1 ASAS context

We consider a scenario with a single subnetwork composed of 50 sensor nodes, 10 of which are subsinks (see Figure 3.2). The subnetwork has a grid-like topology with 5 rows and 10 columns, covering a rectangular area of dimensions 600 m × 1100 m. We also consider a single service vehicle playing the role of MULE, and moving at constant speed of 30 $\mathrm{km\,h^{-1}}$. The experiment is framed on the interaction between the subnetwork and the MULE during

a single visit. At the specified speed, the visit covers a period of 2 minutes along which the MULE is always in contact with at least one subsink. We leave a window of 30 minutes between the beginning of data gathering at the nodes to the time of first contact between the MULE and one subsink.

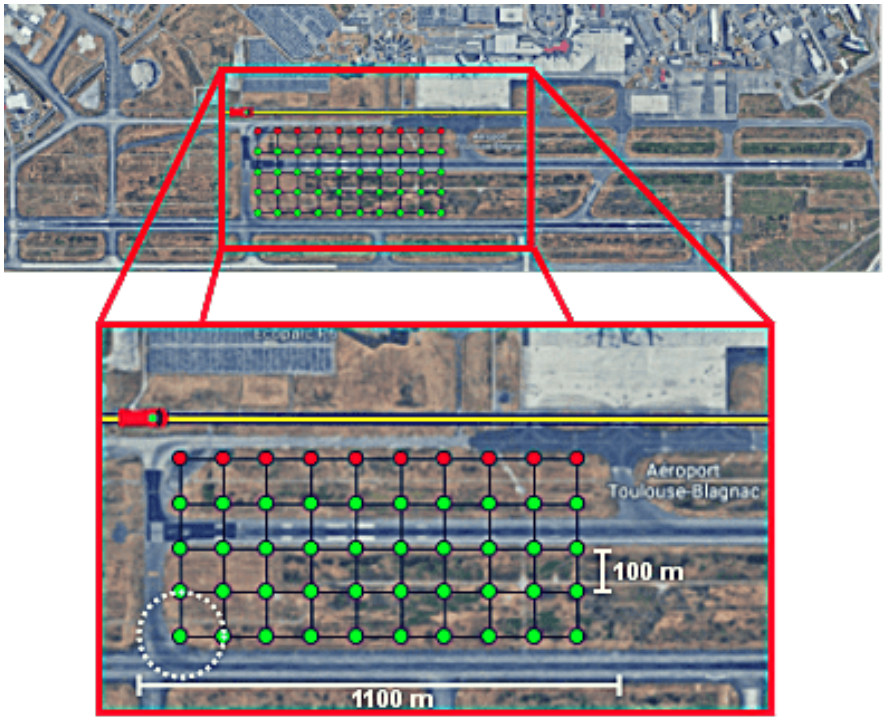


Figure 3.2: Grid-like subnetwork used in the experiment. Displayed at scale over a section of Toulouse-Blagnac airport for illustration.

The simulation of data gathering in this experiment is kept very simple. Each 60 s, every sensor node generates a data packet of 85 B containing the environmental and operational data collected during this period. As specified in our proposal, safety oriented data is assumed to be sent directly to the sink using long-range communication (see Section 1.3.1). The 85 B (payload of 16 B) constitutes a packet large enough to store five 2-digit variables along with the information added by the different layers of the protocol stack. For now we do not explicitly take into account the heterogeneity in the packet size between nodes resulting from event-based monitoring of operational data performed only at the nodes placed by the sides of the taxiways. In this regard, we assume that: (i) the weight of operational data is already accounted for in the packet size; (ii) operational data is sent along with environmental data at the end of the corresponding period; and (iii) nodes not collecting operational data still fill the corresponding fields of the packet with dummy values. The periodicity of 60 s for packet generation is based on the nominal sensing rate used in automatic weather stations at the airports [35]. This value is in turn consistent with the range of sensing rates typically used for environmental monitoring in various applications (one measurement every 10 seconds to some minutes [112]). The treatment of heterogeneous packet sizes is addressed later in Section 4.5.

3.3.2 Sensor node configuration

The architecture implemented in the sensor nodes is displayed in Figure 3.3. The Physical and MAC layers are based on the IEEE 802.15.4 standard. We use the maximum bitrate and communication range enabled by this standard, which are of 250 kbps and 100 m, respectively. In addition, we set the maximum queue length at 100 packets. This selection of parameters is consistent with the choices made in various previous studies [153, 154]. Acknowledgement messages are enabled and the maximum number of transfer attempts is fixed at 7. After that, a loss due to collision is registered and the message is dropped. The CSMA/CA procedure is implemented in the MAC layer to help prevent collisions. Depending on the protocol to evaluate, either C-Sub or N-Sub is used as control plane protocol next to IPv6 as data plane protocol. Transfer delays are used in C-Sub as a complementary action to mitigate collisions. ADVs (used in R-RPL) and data packets are sent with a delay uniformly distributed between 0.07 and 0.1 s and between 0.1 and 0.17 s, respectively. We use UDP in the Transport layer since the loss of an environmental or operational data packet is not critical in the ASAS application, and it omits the connectivity control procedures in TCP. Finally, in the Application layer we just simulate a data gathering process where five variables are collected each 60 s. The first measurement at each node is performed at a random instant uniformly distributed in $[1, 2]$ sec from the beginning of the simulation.

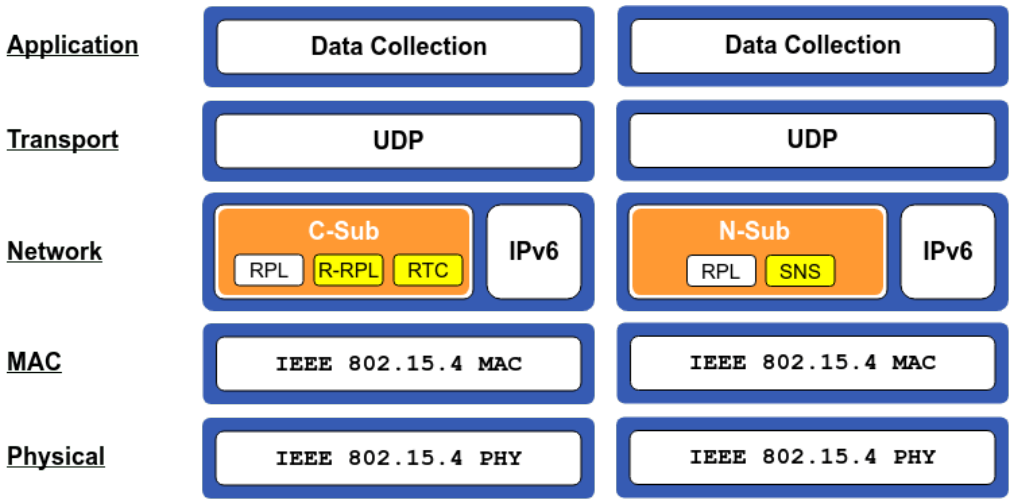


Figure 3.3: Protocol stack for both routing protocols: (i) using the subsink in contact as gateway; (ii) using the nearest subsink as gateway.

We take as reference a TelosB mote [42] frequently used in WSNs for the simultaneous monitoring of multiple variables (see e.g., [155, 156]). TelosB motes feature a 2.4 GHz Chipcon CC2420 IEEE 802.15.4 radio, a Texas Instruments MSP430 microcontroller and up to 1 MB of external flash memory. They have become popular in WSN applications mostly due to the relatively large amount of flash memory and the availability of the MSP430 microprocessor which provides several configurable ports that contribute to the versatility of

the device [157]. The radio transceiver of the node consumes 18.8 mA (56.4 mW at 3 V) in reception and listening mode, 17.4 mA (52.2 mW at 3 V) in transmission mode, and 1 μA (0,003 mW at 3V) in sleep mode [158, 159]. Each node uses two AA rechargeable batteries, whose energy load is typically set to 18720 Joules [151, 160].

3.3.3 MULE configuration

The MAC and Physical layers of the MULE implement the IEEE 802.15.4 standard with similar parameters to those used in the sensor nodes (i.e., bitrate of 250 kbps and communication range of 100 m). At the Network layer it only manages IPv6 as data plane protocol since no control plane protocol is required. The Transport layer is identical to that of the sensor nodes. On the other hand, the Application layer runs a particular protocol responsible for: i) the delivery of periodic beacons to alert the subnetworks about the presence of the MULE, and ii) the storage of data received from the subsinks. In addition, the sending of acknowledgement packets was enabled on the MAC layer to notify the subsinks of the successful reception of data packets. During the whole simulation, the MULE sends beacons at a constant rate of 1 message each 0.5 seconds. When a subsink receives a beacon it starts sending its stored data until it stops receiving beacons and realizes that the contact with the MULE was lost.

3.3.4 Simulation environment

We used the discrete event simulator OMNeT++ [161] for implementation and simulation. Some of the protocols required in the architecture of the nodes and the MULE were already available in libraries of OMNeT++, e.g., IEEE 802.15.4 PHY and MAC, IPv6 and UDP. Other protocols such as those that serve as base for C-Sub and N-Sub (RPL, R-RPL, etc), and those used in the Application layer of both the nodes and the MULE were not available. Therefore, we developed them.

3.3.5 Results

The performance of C-Sub and N-Sub was evaluated in terms of three performance metrics: packet delivery ratio, power consumption and subsink load. We ran 30 replicates (i.e., full simulations) with each protocol in order to take into account the noise due to the random selection of sending delays. The results are discussed below.

• ***Packet Delivery Ratio (PDR):*** percentage of data packets received by the MULE out of the total number of packets sent by all the sensor nodes. The average PDR for C-Sub over the 30 replicates was 52.72%, with a standard deviation of 1.26%. The causes of data packet loss were collisions, queue overflow and exceedance of the maximum number of backoff

attempts. C-Sub is highly susceptible to those problems since it makes all the nodes send data to the same subsink at the same time. This turns the nodes around the subsink in contact with the MULE into bottlenecks, as they are included in the shortest paths between several nodes and that subsink. Packet loss mainly occurred in two situations: (i) massive transfer of data from the sensor nodes to the first set of subsinks during the early moments of contact with the MULE, and (ii) transfer of data between subsinks. In N-Sub none of these problems took place since it did not require any communication between the subsinks, the data was sent to the subsinks in a gradual manner as it was collected and before the arrival of the MULE, and data was distributed among them. Thus, localized congestion spots were avoided. Table 3.1 shows the superiority of N-Sub over C-Sub in terms of PDR. The standard deviation of the PDR for N-Sub shows that it is also considerably more stable than C-Sub in this dimension.

Table 3.1: Comparison of C-Sub and N-Sub protocols in terms of PDR and PC.

	Performance metric	**Mean**	**Standard deviation**
C-Sub	PDR (%)	52.72%	1.26%
	Energy (J)	15.62	0.16
N-Sub	PDR (%)	98.19%	0.06%
	Energy (J)	10.63	0.07

• ***Power Consumption (PC):*** average over the 30 runs, of the total energy spent by all the sensor nodes. Results in Table 3.1 show that C-Sub causes in average greater PC. On the one hand, this measure is influenced by the length of the routes used to reach the destination (i.e., the total number of hops). In C-Sub, the subsink is assigned without considering the distance separating it from the sensor nodes. In contrast, N-Sub performs optimally in this aspect since it assigns each sensor node the subsink closest to it. On the other hand, the PC is inflated by congestion issues such as a high number of transfer attempts at nodes receiving substantial traffic; problems mitigated in N-Sub. Similarly as for the PDR, N-Sub shows a lower standard deviation than C-Sub in the PC, indicating a more stable performance.

• ***Subsink Load (SL):*** average over the 30 runs, of the percentage of packets received by each subsink out of the total number of packets received by all the subsinks. Figure 3.4 shows the SL for each of the 10 subsinks, for each protocol. The strong imbalance in SL obtained with C-Sub comes from the large number of packets collected by the subnetwork during the 30 minutes previous to the visit of the MULE, which are all suddenly transferred to the first subsink once the MULE is detected. Although the subsink exploits the time in contact with the MULE in continuous data transfer, this period is not long enough for it to deliver all the stored data. Thus, it has to redirect the remaining data to the second subsink which in turn delivers as much data as it can to the MULE and transfers the remainder to the next

subsink. The overload of the first subsink is so large that it only gets to transfer 57.79% of the data it receives to the MULE. The amount of retained data progressively shrinks as the MULE moves forward along the subsinks line. Two important consequences of the SL imbalance in C-Sub are: (i) considerably larger energy spent in the overloaded subsinks, specially because those are required to redirect retained packets to the other subsinks; and (ii) unnecessarily large number of collisions due to the packets having to cover unnecessarily large and congested paths to reach the MULE.

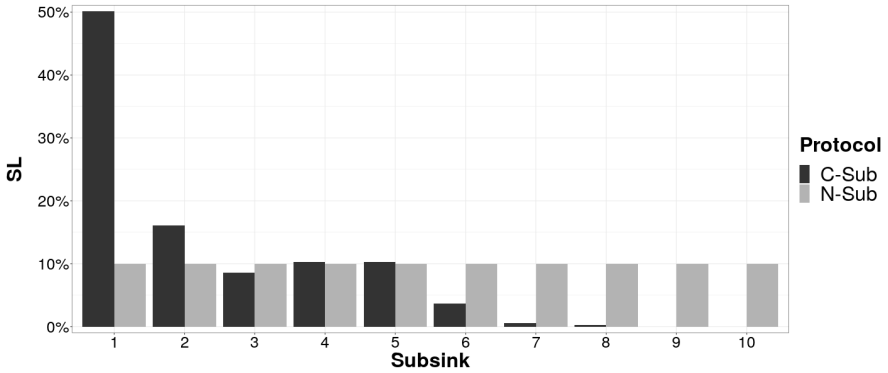


Figure 3.4: Subsink Load for each protocol.

In contrast to the results with C-Sub, N-Sub delivered a perfect balance in SL. This comes from the even assignment of sensor nodes to subsinks, which helps to reduce the number of bottleneck nodes, and thus, the amount of dropped packets by queue overflow, exceedance of the maximum number of backoff attempts and collisions.

3.4 Discussion and next steps

In this chapter we presented the formal definition of our routing problem departing from an ASAS setting, and then we evaluated two strategies previously adopted in the literature for data routing in scenarios involving MULEs; (i) data transfer to the MULE using the subsink in contact as gateway, and (ii) data transfer to the MULE using the nearest subsink to each node as gateway. We provided the formal definition of each strategy as a routing protocol by defining the necessary modules required for their functioning in the proposed system, thus giving rise to C-Sub and N-Sub (in corresponding order). We evaluated the performance of both protocols using realistic parameters from a typical ASAS instance like the data gathering rate, spacing between nodes and speed of the MULE. The results were quite conclusive, showing a strong superiority of N-Sub over C-Sub in terms of packet delivery ratio, power consumption and subsink load. These findings indicate a notable benefit in distributing gathered data among the subsinks and the early transfer of collected data to the subsinks before the arrival of the MULE. These two actions together help to mitigate the congestion and collision issues highly affecting the system operating under C-Sub.

Despite the clear superiority of N-Sub, there is still much place for improvement on it. The experiment presented in this chapter was performed using a subnetwork with regular grid-like topology, where this protocol indirectly provides perfectly even distribution of data packets among the subsinks. We expect the results and conclusions to change significantly if we try an irregular subnetwork topology, where the selection of G-subsinks based on the least number of hops might result in uneven subsink loads. This deficiency of the protocol is greatly important since the overload of some nodes could lead to various issues, including: (i) the overloaded subsinks prematurely getting out of battery; and (ii) the overloaded subsinks not being able to transfer all the stored data to the MULE while the less loaded ones having even spare time in contact where no data is transferred. Since the subnetworks in the ASAS application often present irregular topologies, it is a priority to eliminate this limitation from our routing protocol. It is worth noting that such an extension is not straightforward; the transition from regular grid-like subnetworks to irregular ones leads to a complex optimization task where the objectives of efficient routing and even assignment of loads to the subsinks might often be in conflict. The generalization of this method is the subject of Chapter 4.

Chapter 4

WSN for ASAS: advanced routing based on Ant Colony Optimization

The chapter in brief

In Chapter 3 we presented our modeling scheme and evaluated the performance of the C-Sub and N-Sub routing protocols, defined and implemented based on strategies previously used in the literature for scenarios involving MULEs. Our computational experience allowed us to note a clear superiority of N-Sub in terms of packet delivery ratio, energy consumption and subsink balance. Indeed, we concluded that N-Sub is able to give optimal results in terms of both efficient routing and subsink balance when the subnetwork has a grid-like topology. However, we also concluded about the difficulties that this protocol might have in terms of subsink balance for subnetworks with irregular (non grid-like) topology. The overload of some subsinks is a highly undesirable condition in our setting which makes it less likely that all the stored data will be effectively transferred to the MULE during its visit, and also causes the uneven consumption of the battery at the subsinks. Motivated by the wide variety of subnetwork topologies that could arise in the ASAS application, we concluded about the need for a more sophisticated routing protocol, able to deal with the trade-off between efficient routing and subsink balance in irregular subnetworks. In the present chapter we undertake the construction of such a protocol based on the well known Ant Colony Optimization (ACO) technique, which provides a general framework for multi-objective distributed optimization, conveniently well-suited to the needs of WSNs. We evaluate the performance of the proposed protocol later in Chapter 5.

Contents

4.1 Extended routing setting

Most of the modeling adopted in Chapter 3 remains valid in the present chapter. The only major change is the generalization of the subnetwork topology, which is intended to take into account the variety of subnetwork structures that might appear in the ASAS application. However, this apparently slight modification has two important implications for the routing protocol:

1. ***Efficient routing and subsink balance become conflicting objectives:*** in this chapter we retain efficient routing and subsink balance as our main objectives of performance. In contrast to Chapter 3, where the ideal routing strategy delivering optimal performance in both objectives was feasible, in the present chapter we must deal with a conflict between the two objectives induced by the asymmetries found in irregular topologies.

2. ***Most subnetworks experience greater collision issues:*** we anticipate that irregular topologies will in many cases induce a greater number of collisions. Thus, in the present chapter we integrate to the protocol a complementary, third objective, oriented to mitigate this type of issue.

In the light of these two new conditions, the N-Sub protocol is no longer well suited to the problem. If used in an irregular subnetwork it would voraciously concentrate on the minimization of the path lengths, leaving the subsink balance and collisions totally neglected. Therefore, in the present chapter we adopt a more sophisticated approach from the perspective of multi-objective optimization. To this end, we adhere to the Ant Colony Optimization (ACO) technique, revised in Section 2.4, which is not only well suited for multi-objective optimization settings, but is also naturally well conditioned for implementation in decentralized WSNs. In addition, it provides a general optimization framework where other objectives could be added to the protocol in the future to make it more versatile and comprehensive.

Throughout this chapter we explain the development of our ACO-based routing protocol for MULE-assisted WSNs. In Section 4.2 we start by providing an overview of the ACO technique, going through its biological inspiration, an introduction to the Ant Colony System (ACS), which is the version of ACO algorithm that we implement here, and finally the adaption of ACS to routing in WSNs. Then, in Section 4.3 we introduce and explain the proposed protocol. In Section 4.4 we provide further technical details related to the initialization of the protocol, the treatment of the objective functions and some useful simplifications when only efficient routing and subsink balance are optimized. Finally, in Section 4.5 we present a sketch for two potential extensions of our protocol which could be of great relevance as future research steps. The analysis of performance of our protocol is the subject of Chapter 5.

4.2 The Ant Colony Optimization technique

4.2.1 Biological inspiration

ACO algorithms seek to replicate the natural learning process of real ant colonies. These animals are capable of finding efficient paths between the nest and food sources via *stigmergy*, a mechanism for inter-agent communication through traces left in the environment [162]. This process can be explained with support on the frame sequence displayed in Figure 4.1. All the ants start located at the nest, with two possible pathways towards a food source, one significantly longer than the other (Frame 1). At the beginning, the ants have no information about the length of each path, thus they start randomly distributing themselves among the two alternatives in approximately uniform manner (Frame 2). As they walk, ants leave pheromone trails in the ground which are noticeable to their peers. As expected, the ants that take the shortest path find the food sooner and some of them even start their way back to the nest before the first ant traveling through the longest path reaches the food source (Frame 3). On the way back, the decision is already biased towards the shortest path since the pheromone load on it is larger and the ants feel more attracted to it. The shortest path keeps receiving pheromones at an incrementally higher rate than the longest one, which gradually reduces the number of ants traversing this last (Frame 4). The bias is reinforced by the fact that the pheromone trail evaporates, thus both paths lose some pheromones but only the shortest path gets recovered. After some time the pheromone load in the shortest path is stronger enough to dominate the selection of every ant and the colony reaches convergence over the efficient alternative (Frame 5). This phenomenon was observed by Goss et al. during the *double bridge* experiment conducted with real ants in the 80s [163].

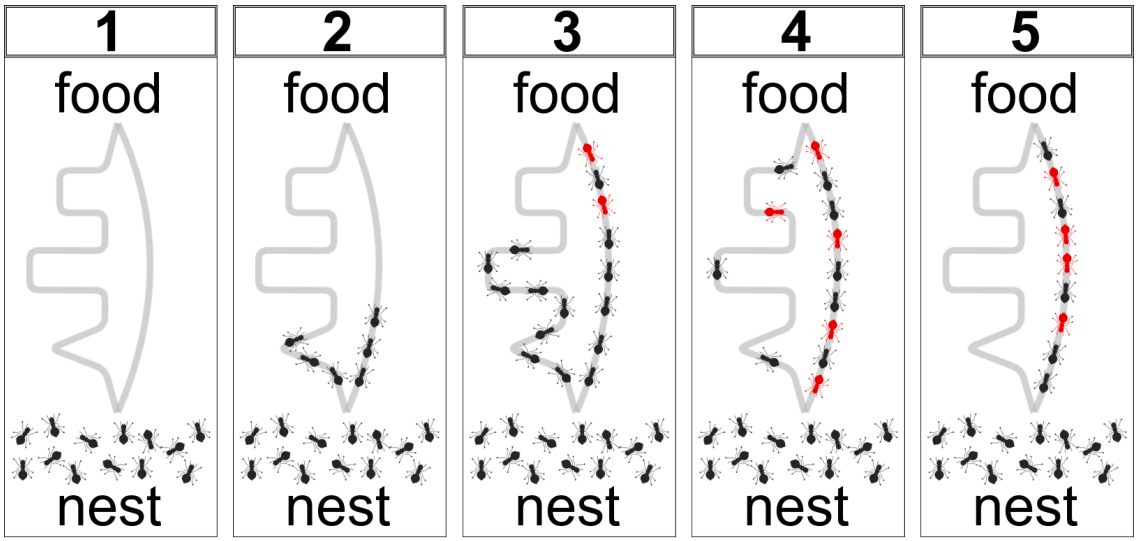


Figure 4.1: Ability of ant colonies to find efficient paths towards food sources. Red ants represent the ones going back from the food source to the nest.

4.2.2 The Ant Colony System

ACO encompasses a large variety of optimization metaheuristics inspired by the ability of real ants to find short paths between the nest and food sources in a decentralized but still collective manner. Indeed, ACO has been acknowledged as one of the most successful research lines in the area of swarm intelligence [164, 165]. Since the seminal work of Dorigo et al. in the early 90s [120, 121, 166], several authors have proved the pertinence of this technique in a wide range of optimization problems such as machine scheduling [167], assembly line balancing [168] and symbolic regression [169]. Several versions of the general ACO algorithm introduced in [120] were later developed seeking to improve its performance [51].

Most of the studies in the literature proposing routing protocols for WSNs based on ACO use the first version of the family of ACO algorithms called Ant System (AS) [121]. This fact was unexpected for us since an improved version, called Ant Colony System (ACS) [170], was developed by the same author more than two decades ago, and since then, it has become the most popular and widely implemented ACO-based algorithm in most other research fields (see e.g., [171, 172]). ACS induces a higher level of exploration of the solution space, thus preventing the algorithm from getting stuck at local optima and avoiding a premature convergence. Our proposal is developed upon ACS, contributing to the growth of the state of the art in WSN routing towards more sophisticated and better performing techniques. In the remainder of this section we briefly recall the functioning of the ACS and then, in the next section, we delve into the adaptations required to make it suitable for routing in WSNs.

The original ACS algorithm was conceived to resolve general optimization problems in a centralized manner. It was expected to be implemented in a single computer with general control of the information and the actions involved in the whole optimization procedure. Below we recall the functioning of the ACS with a written description. A complementary pseudocode is offered in Algorithm 1.

The following description applies for a generic optimization problem involving integer and/or binary decision variables. This includes most variations of routing, scheduling and sorting problems. The ACS operates over a decision network composed of links and nodes. This network is recursively traversed by artificial ants that build candidate solutions to the optimization problem by gradually adding nodes to their path based on a pseudo-random system of rules. Each time an ant traverses a link, a local pheromone update takes place and the pheromone load of that link is decreased. This action foments the diversity among the solutions built by different ants and prevents the premature convergence of the algorithm (principle of ***exploration***). Once all the ants have built a feasible solution, the objective function is evaluated on each of them and a global pheromone update is performed to increase the pheromone load of links that make part of the best solutions. This mechanism is intended to lead the exploration to the section of the solution space containing the best

solutions found so far (principle of ***exploitation***). The whole process is iterated with new groups of ants until some stopping condition is met (e.g., a given number of iterations).

Algorithm 1 Original ACS

```
1: while <stopping conditions remain unsatisfied> do
2:     create a new population of ants
3:     for <i=1:Psize> do
4:         locate ant i at its base spot
5:         tag ant i as partial
6:     end for
7:     while <there are still partial ants> do
8:         randomly pick a partial ant
9:         apply transition rule to select next node in its sequence
10:        reduce pheromone load of chosen link
11:    end while
12:    evaluate the solution made by each ant
13:    increase pheromone load of links in best solutions
14:    P* ← best solution so far
15: end while
16: return P*
```

4.2.3 Adaptation to routing in WSNs

As pointed out in Section 2.6, decentralized or lightly centralized schemes should be favored in WSNs to prevent limitations linked to the special needs of the central node in fully centralized systems. In this section we briefly explain how the ACS can be adapted to routing in WSNs by switching to a decentralized, or lightly centralized scheme. For the sake of exposition, let us first consider the case of a connected network with a single sink reachable by all the nodes via multi-hop, which corresponds to the typical setting where ACO-based routing protocols have been applied so far (see e.g., [139]). In Section 4.3 we adjust the framework to our MULE-assisted setting where a suitable G-subsink must be selected for each node.

In WSNs, ACS purely relies on local communication between the nodes and their neighbors. The system recursively builds candidate solutions by generating one packet at each node and finding a suitable route to the sink for it. The packets in this system play the role of ants, and the route for each ant is made based on probabilistic decisions that the nodes make to dynamically select each of its hops. Those probabilistic decisions depend on pheromone tables stored and maintained by the nodes. Each time a node forwards an ant, it performs local pheromone update by slightly decreasing the pheromone level of that neighbor. Every time a batch of ants from all the nodes reach the sink, the system obtains a candidate routing solution. The quality of this solution is evaluated by the sink in terms of a predefined performance indicator and then compared to the quality of the best known solution. After the comparison, the sink only retains the information on the best known solution. Every few solutions, the sink sends a set of ants through the network to perform global pheromone update. This is done by increasing the pheromone level of the hops covered in the best known

solution. As in the classic ACS implementation, in WSNs the algorithm runs until a given stopping condition is met, e.g., a predefined number of iterations. It is worth noting that, in WSNs, a candidate solution is composed of the routes from all the nodes to the sink, i.e., by the paths built by a whole batch of ants, and not by the path built by a single ant, as was the case in the original ACS implementation.

We remark that the evaluation of the quality of each solution is a simple task requiring only algebraic and sorting operations which do not require any special provision of battery, memory or processing power for the sink. Those special needs are avoided by decomposing the most intensive part of the optimization procedure into minor local actions performed by the nodes. This way, the overall network performance is optimized in a distributed manner.

4.3 ACME: an ACO-based routing protocol for MULE-assisted WSNs

4.3.1 The protocol in brief

Similarly to various other protocols in the literature, ACME works in two phases; a first one of construction of routes, and a second one of data gathering and transfer. Naturally, most of the essence of the protocol is concentrated on the ***first phase***, where potential routing solutions are tested by iterations while the system gradually learns how the best solutions are constituted. To produce a solution, every node generates an ant, picks a G-subsink and a next hop, and sends the ant through the subnetwork, where it is dynamically routed by other nodes until reaching the selected G-subsink. For each ant, the selection of the G-subsink and the next hop at each step are made based on pheromone tables stored at the nodes. The pheromone level of any node represents the expected benefit of choosing it as part of the routing strategy. Thus, the nodes make decisions in a probabilistic manner, giving larger probability of selection to the subsinks and neighbor nodes with larger pheromone level. Every time a batch of ants from all the nodes reach their corresponding G-subsinks, each subsink sends the information it has on the quality of the solution to a lead subsink call ***checkpoint***, which computes a general performance metric. Every few iterations, the checkpoint sends a feedback packet back through the network to perform global pheromone update by increasing the pheromone values of the G-subsink and hops selected in the best known solution. This way, the nodes are induced to repeat part of those decisions in the upcoming solutions (*exploitation*). To prevent the premature convergence of the algorithm at a local optimum solution, the nodes consistently decrease the pheromone level of the G-subsinks and neighbor nodes selected in each solution. This way, diversity in the solutions is promoted (*exploration*) and only the decisions which consistently appear in the best solutions get to maintain a relatively high pheromone value. Similarly as in nature, the solutions generated by the system progressively converge to the ideal decisions resulting in the best overall performance of the subnetwork. Once a given stopping condition is met, the protocol

enters in ***second phase***, where the subnetwork starts the operation and the nodes use as static routing solution the decisions of G-subsink and next hops included in the best solution found during the optimization.

In contrast to most ACO-based protocols proposed in the literature, ACME implements the local pheromone update included in the ACS which stimulates wider exploration and prevents premature convergence. In addition, ACME incorporates the selection of the G-subsink for each node into the optimization, an aspect that has not been considered by any other ACO-based protocol that we know. We want to stress the fact that, although some centralization of information occurs at the checkpoint, the operations performed by it remain fairly simple. Thus, as explained in Section 4.2.3, no special features are required for this node.

4.3.2 Detailed step-by-step explanation

The pseudocode for ACME is provided in Algorithm 2, followed by a detailed step-by-step explanation of the protocol.

Algorithm 2 ACME workflow

- **Nc:** number of colonies to run; **Ns:** number of solutions per colony
- **n:** number of ant packets per solution, corr. to the number of regular (non-subsink) nodes
- $\boldsymbol{T_{\text{ants}}}$: time allowed to ant packets to reach their subsinks
- $\boldsymbol{t_{\text{now}}}$`()`: function that returns the current time

```
1: procedure ACME
2:     perform network discovery
3:     for <i=1:Nc> do
4:         for <i=1:Ns> do
5:             generate n ant packets
6:             select a subsink for each ant
7:             perform local pheromone update for subsinks
8:             t0 ← tnow()
9:             while < tnow() − t0 ≤ Tants > do
10:                select the next hop for each partial ant
11:                perform local pheromone update for neighbors
12:                mark as complete ants that have reached their subsink
13:            end while
14:            if < all ant packets are complete > then
15:                store the ID and performance of each ant packet at its subsink
16:            else
17:                delete all complete ant packets from this solution
18:            end if
19:        end for
20:        evaluate the quality of each solution at the checkpoint
21:        perform global pheromone update for subsinks
22:        perform global pheromone update for neighbors
23:        R* ← best solution so far (subsinks and paths)
24:    end for
25:    return R*
26: end procedure
```

Steps

◇ **Step 1.** ***Network discovery:*** this step provides every node with the necessary information to start the optimization. It comprises the execution of five main procedures which are described below[1].

1.1. ***Identification of subsinks and checkpoint:*** when the MULE visits the subnetwork for the first time, the nodes that get contact with it become aware of their role as subsinks. The first subsink in getting contact is assigned as checkpoint. Alternatively, when possible, the subsinks and checkpoint could be assigned in advance by the system administrator.

1.2. ***Filling of cost vectors:*** the checkpoint launches the construction of an RPL tree that lets all the nodes become aware of the cost to reach it. The nodes use the RPL routes to reply with a packet indicating if they are regular (non-subsink) nodes, or subsinks. The subsinks also include the cost to reach them in the packet. This procedure lets the checkpoint know the total number of nodes and the number of subsinks, which it uses later to control the timing in the protocol. Then, the checkpoint uses R-RPL to reach the other subsinks and asks them, one by one, from the closest to the farthest, to launch an RPL tree as well. Only one tree is built at a time to prevent unnecessary congestion issues at this stage. The construction of each tree lets every node in the subnetwork know the cost of reaching the corresponding subsink. Thus, throughout this procedure each node gets to fill the cost vector that indicates its cost to reach any subsink.

1.3. ***Neighbor discovery:*** the checkpoint uses R-RPL to reach the nodes and triggers the neighbor discovery. Every node announces itself to its neighbors so that these last can consider it as a potential next hop for any packet they receive. The checkpoint also takes advantage of this message to indicate the nodes how much time to wait to perform pheromone initialization (Step 1.4) and launch the optimization (Step 1.5), and also how much time to leave between batches of ants. We provide more details on how these times are fixed in Appendix A.

1.4. ***Pheromone initialization:*** once the neighbor discovery is complete, the nodes perform pheromone initialization. In its most general form, ACME considers a pheromone vector and a pheromone table stored and managed by each node. The pheromone vector is intended to reflect the expected benefit of selecting each subsink as G-subsink for the packets generated at that node. Similarly, the pheromone table

[1]We recall that the RPL and R-RPL methods, used in Steps 1.2 to 1.5, provide routes from the leafs to the root and from the root to the leafs, respectively. A full explanation of both methods was provided in Section 3.2. ACME implements the version of R-RPL with confirmation to ensure that all the critical procedures for the optimization are well carried on.

is intended to reflect the expected benefit of selecting each neighbor as a next hop towards each of the subsinks. As will be explained later in Section 4.4.4, the pheromone tables for the neighbors are only used when there is active a complementary objective for which an alternative route, different to the shortest path towards a given subsink, might have sense. This is the case of our third objective, since it could be desirable to slightly deviate a packet from the shortest route in order to avoid collisions. When only efficient routing and subsink balance are active in the optimization, only the subsink pheromone vectors are necessary.

Both the pheromone vector and table at every node are filled considering some greedy information on the potential benefit of each node as G-subsink or next hop. For this, each node uses at least the information contained in its own cost vector and that of its neighbors. We could roughly say that the closest subsinks are given larger pheromone levels as G-subsinks and the neighbors which are closest to a given subsink are given larger pheromone levels as next hops towards it. Since subsink balance is also a main target for us, we also incorporate greedy information about this objective on the initialization of the subsink pheromone vectors. More precisely, we let run an auxiliary, simplistic protocol able to produce ideal solutions in terms of subsink balance in a very quick and cost-efficient manner. It is important to remark that this auxiliary protocol is not able to deal with the trade-off between efficient routing and subsink balance or to incorporate other objectives to the optimization as ACME does. Its use is thus limited to support in the initialization of pheromone vectors within ACME. The greedy protocol used in this thesis for subsink balance is explained later in Section 4.4

1.5. ***Launch:*** finally, each node generates a first ant and the optimization starts.

Note that more elaborate methods could be used to set a more central subsink as checkpoint. However, here we use the first node as checkpoint for simplicity on the explanation of the method, and also because the selection of a central node as checkpoint is expected to have a negligible impact on the performance of the method.

◇ **Step 2.** ***Generation of ant packets:*** as explained in Section 4.3.1, ant packets are generated at the nodes to produce paths towards the MULE. Each time a node generates an ant, it selects a G-subsink and a next hop for it and forwards it through the subnetwork, where it is dynamically routed by other nodes until reaching the selected G-subsink. Every time a batch of ***ants*** from all the nodes reach their corresponding G-subsinks, the system obtains a candidate ***solution***. In turn, every time that the number of solutions necessary for a global pheromone update is reached, we say that the system has completed a ***colony***. This hierarchical relationship is illustrated in Figure 4.2.

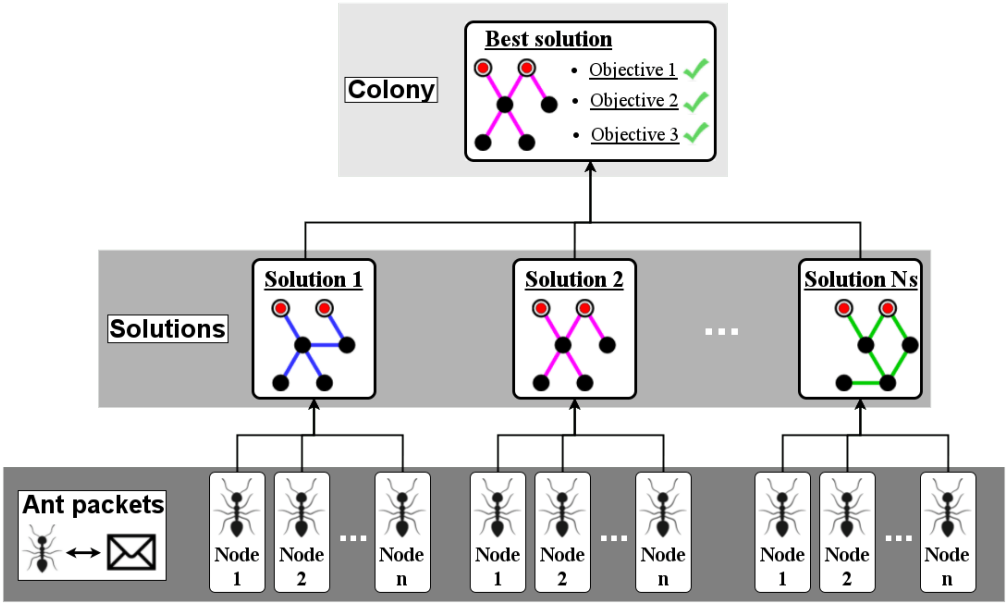


Figure 4.2: Hierarchical relationship among *ant packet*, *solution* and *colony*. The parameters n and N_s denote the number of regular (non-subsink) nodes and solutions in one colony, respectively. The subsinks are not included in the set of possible ant sources since they do not generate this type of packet. Data collected by one subsink simply stays stored at it until the MULE visits the subnetwork.

Ants from different solutions must be generated spaced enough in time to allow the ones launched first to explore a routing alternative without colliding with others launched later. The time between ants generated by a node is provided by the checkpoint in Step 1.3. Such a time is common among nodes and remains fixed during the optimization. A tiny random number is added to the timer that controls the generation of the first ant at each node, helping to prevent ant collisions while departing from their source nodes. More details on the timing of the protocol are provided in Appendix A.

◇ **Step 3.** ***Subsink selection:*** each time a sensor node generates an ant packet, the first thing it does is decide the G-subsink for it. The node makes such a decision based on the *subsink pheromone level* and the *subsink decision rules*. Details on these two elements are provided below.

- ***Subsink pheromone level:*** each node stores a custom pheromone vector which provides information about the potential benefit of selecting each subsink as a G-subsink. At each node, we formally denote by γ_k the pheromone level of subsink $k \in \{1, \ldots, K\}$, with K the total number of subsinks in the subnetwork. The simplest way to initialize the pheromone vector is to set equal pheromone levels for all the subsinks. This approach, however, often implies too long learning periods due to the need to invest some colonies in the initial screening of the solution space. As explained in Step 1.4,

ACME adopts a more elaborate initialization approach which is to feed the subsink pheromone vectors with greedy information indicating good G-subsink selections in terms of our two main objectives of optimization. The idea is to favor the G-subsink selection that has greater potential of providing good results in terms of one of the two objectives, or even better in terms of both. This way, the ants start exploring the solution space with some bias towards a region of potentially good solutions. More details are provided in Section 4.4.

- ***Subsink selection rules:*** once the pheromone vectors have been initialized, each node proceeds with the selection of a G-subsink. This decision is made by means of a pseudo-stochastic system of rules defined as follows:

$$\text{rule} = \begin{cases} \textbf{\textit{Rule 1}} & \text{if } q \leq q_0, \\ \textbf{\textit{Rule 2}} & \text{otherwise,} \end{cases} \tag{4.1}$$

with $q_0 \in [0, 1]$ a threshold to be defined by the user and q a random value from $U(0, 1)$ to be generated during the optimization each time the system of rules is implemented. The two decision rules are defined below.

Rule 1. Select the subsink with the greatest pheromone level. Mathematically, this rule can be expressed as:

$$k^* = \underset{k \in \{1,\dots,K\}}{\arg\max}\, \gamma_k, \tag{4.2}$$

where k^* represents the subsink selected as a G-subsink.

Rule 2. Select the subsink randomly based on a probability pie resulting from the standardization of the pheromone vector to sum 1. Formally, the probability of choosing subsink k can be computed as:

$$P_k = \frac{\gamma_k}{\sum_{k=1}^{K} \gamma_k}, \tag{4.3}$$

The value of q_0 in Eq. (4.1) leads to the trade-off between exploration of new solutions and exploitation of acquired knowledge. The lower it is, the more frequently each node will use the second decision rule oriented to diversification of the solutions. The higher it is, the more frequently each node will repeat part of the best known solution.

We recall that the pheromone vector stored by each node is different. Even though all the nodes use the same system of decision rules.

◇ **Step 4.** ***Neighbor selection:*** once a sensor node has chosen the G-subsink for the ant it generated, the node has to choose the next hop for it. This process is made exactly the same as if a sensor node receives an ant packet generated at a different node, and needs to pick a neighbor as its next hop. In that case, the node must only select a neighbor node to transfer the ant packet and keep its G-subsink unchanged. Analogously to the selection of the G-subsink, the next hop is selected based on the *neighbor pheromone level* and *neighbor selection rules*. Both elements are detailed below.

- ***Neighbor pheromone level:*** in this case, each node stores a custom tridimensional pheromone matrix providing information on the potential benefit of selecting any neighbor $j \in \{1, \ldots, J\}$ as next hop for an ant packet generated at node $i \in \{1, \ldots, n\}$ moving towards subsink $k \in \{1, \ldots, K\}$, with J, n and K denoting the total number of neighbors of the node making the decision, regular nodes and subsinks in the subnetwork, respectively. We formally denote the elements of such a matrix by τ_{ijk}. Note that the definition of source-specific pheromone loads helps the protocol to mitigate congestion-related issues by enabling the nodes to distribute, over alternative routes, the traffic they receive from different sources and going to a common subsink. This is particularly relevant for the optimization of our third objective of performance, related to the mitigation of collisions.

  Since the source and destination of the ant are defined at the moment of its generation, the subscripts i and k are kept fixed during the selection of the neighbor. Thus, the node making the decision only considers a j-element vector of pheromones while choosing the next hop for the ant, corresponding to the i-th row of the bidimensional pheromone submatrix stored for subsink k. The neighbor pheromone level at each node is initialized only based on the cost vector filled in Step 1.2. As explained in Step 1.4, neighbors with lower cost towards a given subsink receive larger initial pheromone loads. More details are provided in Section 4.4.

- ***Neighbor selection rules:*** for the selection of the neighbor, a node makes use of a system of rules identical to the one used for the selection of the subsink (see Eq. (4.1)). For one node selecting the next hop of an ant packet generated at node i and traveling towards subsink k, the two neighbor selection rules are defined as follows:

  Rule 1. Select the neighbor with the greatest pheromone level:

$$j^* = \underset{j \in \{1,\ldots,J\}}{\arg\max}\, \tau_{ijk}, \tag{4.4}$$

  where j^* represents the selected neighbor.

Rule 2. Select the neighbor randomly based on a probability pie resulting from the standardization of the pheromone vector linked to the duple source-subsink (i, k). The probability of selecting neighbor j is formally defined as:

$$P_j = \frac{\tau_{ijk}}{\sum_{j=1}^{J} \tau_{ijk}}, \tag{4.5}$$

As explained in Step 1.4, the pheromone tables for the neighbors are only used when there is active a complementary objective for which an alternative route, different to the shortest path towards a given subsink, might make sense. Consequently, neighbor selection based on pheromone levels and the system of rules presented above is reserved only for those cases. In other instances, where only the objectives of efficient routing and subsink balance are optimized, each node always selects as next hop for any ant it forwards, its neighbor with the lowest cost to reach the relevant G-subsink. The result in those cases is a routing based on the shortest-path trees formed in Step 1.2.

◇ **Step 5.** ***Local pheromone update:*** each time a G-subsink or next hop is selected, a reduction in the corresponding pheromone value takes place. This procedure promotes the diversification of solutions within a given colony, seeking for the exploration of unknown regions of the solution space. The equations used in both cases are presented below.

- ***Update for subsink:*** after a subsink selection, the corresponding pheromone value, γ_k, is modified as follows:

$$\gamma_k \leftarrow (1 - \rho_l) \cdot \gamma_k + \rho_l \cdot \tilde{\gamma}_k, \tag{4.6}$$

  where $\tilde{\gamma}_k$ is the initial pheromone level of subsink k and $\rho_l \in [0, 1]$ is a parameter that determines how fast a subsink loses the traction gained for being part of the best solution found in a previous colony. If ρ_l is set to 1, the subsink pheromone vector at each node easily gets back to its original state after a colony is complete. If it is set to 0, then it becomes very hard for the system to explore solutions different to the best found in the first colony. This parameter is often referred to, in the literature, as pheromone evaporation rate.

- ***Update for neighbor:*** after a neighbor selection, the corresponding pheromone value, τ_{ijk}, is modified as well. The update is made as follows:

$$\tau_{ijk} \leftarrow (1 - \phi_l) \cdot \tau_{ijk} + \phi_l \cdot \tilde{\tau}_{ijk}, \tag{4.7}$$

where $\tilde{\tau}_{ijk}$ is the initial pheromone level of neighbor j for the duple of source and G-subsink (i,k). The parameter $\phi_l \in [0,1]$ can be interpreted similarly to ρ_l in Eq. (4.6).

◇ **Step 6.** ***Evaluation of the quality of the solution:*** every solution made by the system is evaluated in terms of a weighted sum of three objective functions which are conceptually defined as follows:

- $\mathbf{O_1}$**:** ***minimization of total path length.*** Sum of the number of hops in the route of all the ant packets;
- $\mathbf{O_2}$**:** ***minimization of subsink overload.*** Sum of deviations in number of ants received by each subsink with respect to the ideal number of ants it must receive;
- $\mathbf{O_3}$**:** ***Minimization of total number of retransmissions.*** Sum of retransmissions experienced by all the ant packets along their path.

We remark that O_1 and O_2 constitute our main targets of performance since those alone respond to the primary needs of a MULE-assisted subnetwork in our ASAS application. O_3, on the other hand, is regarded as a complementary objective that could help to improve packet loss rate and energy consumption in instances heavily affected by collisions. Further details on the computation of the three objectives are provided in Section 4.3.3.

Mathematically, the quality of any solution generated by the system is denoted by Ψ, and is defined as follows:

$$\Psi := w_1 \cdot \mathcal{N}(O_1; \lambda_1) + w_2 \cdot \mathcal{N}(O_2; \lambda_2) + w_3 \cdot \mathcal{N}(O_3; \lambda_3), \tag{4.8}$$

where w_1, w_2 and w_3 are weighting coefficients defined by the system administrator, which indicate the relative relevance of each objective, and $\mathcal{N}$ is a normalization function parametrized by the scaling parameter λ_i, for $i = 1, \ldots, 3$. The normalization function is used to put all the objectives in a common scale and control the way of penalizing deficits in each objective. We define $\mathcal{N}$ as a monotonic decreasing function where greater Ψ is perceived for lower values of the objectives. Since our three objectives are of minimization, we seek solutions with larger Ψ. From here and on, we refer to the quality of any routing solution, computed by Eq. (4.8), as its ***fitness***. More details on the characteristics of $\mathcal{N}$ and its calibration are provided later in Sections 4.4.2 and 4.4.3, respectively.

Let us now explain how ACME goes from the local pheromone update in Step 5 to the evaluation of the quality of a complete solution. Along its path towards its selected G-subsink, every ant records the information necessary to compute objectives O_1 and O_3.

More specifically, each ant maintains a variable **Nb_hops**, indicating the number of hops it has covered, and a variable **Nb_rets**, indicating the number of retransmissions it has experienced. Since information on retransmissions is not available at the Network layer, in ACME, the number of retransmissions at each hop is computed based on the departure time from the sender node, stored at the **S_time** field of the ant, the time of arrival at the recipient node, and the ideal packet delivery time between two nodes (i.e., without experiencing any congestion related issue). The computation of O_2 does not require the ant to store any information along its path since this objective is computed from the number of ants received by each subsink. Every time an ant reaches its selected G-subsink, it delivers the information it recorded on O_1 and O_3. The format of an ant packet is presented in Figure 4.3. The **Packet_type** field must indicate that the packet is an ant. The **Src_ID** and **Gsub_ID** fields must provide the source node and selected G-subsink of the ant, respectively. Those are used for routing and pheromone maintenance in Steps 4 and 5. Finally, the **Sol_ID** field serves as an identifier of the solution to which the ant belongs. This field is used later in Step 7.

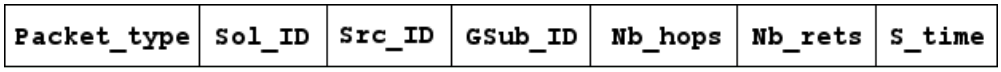


Figure 4.3: Format of an ant packet.

All the subsinks wait for a period long enough to let every ant complete a feasible path. Once every subsink has received all the ants directed to it, it forwards to the checkpoint a *consolidation packet* (Cpack) with the fields **TN_ants**, **TN_hops** and **TN_rets**, indicating the total number of ants received, and the total number of hops covered and retransmissions experienced by them, respectively. This transmission is made using RPL routes. The checkpoint then uses the information received from the subsinks to compute the three objectives by adding up the corresponding values. The format of a Cpack is presented in Figure 4.4.

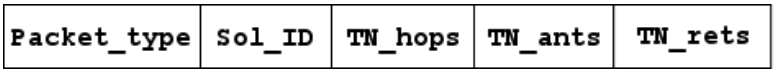


Figure 4.4: Format of a consolidation packet.

Note that some solutions might not become complete because some ants did not reach the selected G-subsink. This could happen, for instance, if the ant made a loop at some point. Incomplete solutions are not feasible since they do not provide a practicable path for the source nodes of the lost ants. Thus, solutions involving at least one lost ant are discarded by the checkpoint. To identify unfeasible solutions of this kind, the checkpoint just adds up the **TN_ants** of all the Cpacks it receives, and compares the result with the total number of regular nodes, determined in Step 1. Each time the checkpoint identifies

a feasible solution, it compares its fitness with that of the best known solution. After the comparison, the checkpoint only keeps the ID and fitness of the best solution. Each N_s solutions, a colony is constituted and the checkpoint triggers the global pheromone update. This process is described in the next step.

◇ **Step 7.** ***Global pheromone update:*** each time a colony gets complete, an increment in the pheromone level of the G-subsink and next hops selected for each ant in the best known solution takes place. This procedure is intended to direct the exploration to a region of the solution space containing high quality solutions. The checkpoint indicates the nodes which pheromones increase and to which extent by means of a *feedback packet* (Fpack) indicating the ID of the best solution and its fitness. Those two pieces of information are stored in the `Best_Sol_ID`, and `Best_fitness` fields of the Fpack. At the moment of global pheromone update, each node has stored the decisions it made in every solution during the current colony, and also in the best known solution up to the previous global update (see Figure 4.5). The nodes thus use the information in the `Best_Sol_ID` of the Cpack to know which G-subsinks and next hops should have pheromone reinforcement.

Solutions of current colony

SOL_ID	i	j	k
1	8	5	3
1	9	6	2
1	12	6	1
1	10	5	2
2	9	5	2
2	8	6	1
3	9	6	1
3	8	6	3
3	10	5	1

Best known so far

i	j	k
12	5	1
9	7	2
8	6	1

Figure 4.5: Example of routing data stored at the nodes in ACME. The numbers in the columns `i`, `j` and `k` represent IDs of sources, neighbors and G-subsinks, respectively. The components of the best known solution are made different from those of any solution in the current colony to remark that the best known solution could belong to the current or any previous colony.

The transfer of the Fpack to each node is made in a cascade scheme supported on R-RPL with confirmation. The process starts with the checkpoint sending the Fpack to its children and those performing global pheromone update. Then, each of those nodes transfer the Fpack to their children who perform global pheromone update as well. This is repeated until each node has received the Fpack and the global pheromone update in the subnetwork gets complete. Each time a node receives a Fpack, it must update the

pheromone level of the subsink it selected as G-subsink for its ant in the solution indicated by the `Best_Sol_ID` field. The node must also update the pheromone level of any neighbor it used as the next hop for any ant in that solution. The equations used in both cases are presented below.

- ***Update for subsink:*** the pheromone level, γ_k, of a selected G-subsink is updated as follows:

$$\gamma_k \leftarrow (1 - \rho_g) \cdot \gamma_k + \rho_g \cdot \Psi, \tag{4.9}$$

  where Ψ is the fitness of the new best known solution and $\rho_g \in [0, 1]$ is a parameter to be set by the user, which can be interpreted as the learning reinforcement rate.

- ***Update for neighbor:*** the pheromone level, τ_{ijk}, of any neighbor selected as the next hop for a packet with source and G-subsink duple (i, k), is updated as follows:

$$\tau_{ijk} \leftarrow (1 - \phi_g) \cdot \tau_{ijk} + \phi_g \cdot \Psi, \tag{4.10}$$

  where Ψ is the fitness of the new best known solution and $\phi_g \in [0, 1]$ a parameter to be set by the user, analogous to ρ_g in Eq. (4.9).

After every global pheromone update, each node clears all the data it has on the ants of that colony and retains only the information of the best known solution so far (left and right panels of Figure 4.5, respectively). Finally, the checkpoint lets the nodes know when the optimization is done by means of the `Is_final` field of the Fpack, which contains a binary flag. When the nodes receive this notification, they take the decisions of G-subsinks and next hops making part of the best known solution as static routing strategy for the transfer of data packets in the operational phase. The format of an Fpack is presented in Figure 4.6.

Packet_type	Best_Sol_ID	Best_fitness	Is_final

Figure 4.6: Format of a feedback packet.

End of steps

Steps 2 to 7 should be repeated until N_c colonies are complete.

4.3.3 Objectives of optimization

As explained in the previous section, ACME evaluates each candidate routing solution in terms of a weighted sum of three objective functions. To facilitate the overall comprehension of ACME's workflow, only a conceptual definition of those functions was used in the step-by-step description of the protocol. In the present section, we proceed with their mathematical definition.

- **O_1: *minimization of total path length.*** This objective seeks to minimize the total energy consumed in packet transmission along the paths from the nodes to the corresponding G-subsinks. The total path length is computed as the sum of the number of hops made by all the ants in one solution. Thus, O_1 can be mathematically expressed as follows:

$$O_1 := \min \sum_{i=1}^{n} L_i, \tag{4.11}$$

  where L_i denotes the number of hops made by the ant packet generated at sensor node i.

- **O_2: *minimization of subsink overload.*** This objective seeks to maximize the chances of all the subsinks being able to effectively deliver all their stored data to the MULE during an opportunistic visit. As observed in the experiments of Section 3.3, these chances are reduced when any subsink receives an excessive packet load. Thus, with this objective we try to keep the load of all the subsinks as even as possible. The total subsink overload is computed as the sum of differences between the actual load of each subsink, Q_k, and its ideal subsink load Q_k^*. Thus, O_2 can be mathematically expressed as follows:

$$O_2 := \min \frac{1}{2} \sum_{k=1}^{K} |Q_k - Q_k^*|. \tag{4.12}$$

  Note that without the division by 2, the result of Eq. (4.12) can be interpreted as the total number of imbalances in the subsink loads with respect to the ideal load. By means of the division, we just change the interpretation to the number of nodes wrongly assigned to subsinks. Thus, if there is one node wrongly assigned, we count one unit of overload instead of two units of imbalance.

  The computation of this objective is straightforward when the number of regular nodes is a multiple of the number of subsinks, and thus, the ideal subsink load is common to all the subsinks and is an integer number. In the contrary case, the computation is just a bit more elaborate, but still very simple. We first form a vector of ideal integer subsink loads distributed as evenly as possible among the subsinks, and we sort it in

ascending order. Then, we sort the vector of actual loads in ascending order too. Finally, we apply Eq. (4.12) while maintaining the ascending order in both vectors. To illustrate, consider an instance with 4 subsinks and 22 regular nodes. Consider also a candidate routing solution where the subsinks receive 7, 4, 5 and 6 packets, respectively. To compute the total subsink overload of this solution, we first form the vector of ideal subsink loads in ascending order $\boldsymbol{Q}^* = (5, 5, 6, 6)$, and then we sort the vector of actual subsink loads in increasing order to obtain $\boldsymbol{Q} = (4, 5, 6, 7)$. Finally, we compute the sum of element-wise absolute differences between the two vectors and divide by 2 to obtain: $(1/2)(|4-5|+|5-5|+|6-6|+|7-6|) = 1$. This result would be interpretable as one node wrongly assigned.

- **O_3: *Minimization of total number of retransmissions.*** This objective seeks to minimize the total energy consumed in packet *retransmission* along the paths from the nodes to the corresponding G-subsinks. This is achieved by forming the paths in such a way that collisions are avoided. The total number of retransmissions in one solution is computed as the sum of the retransmissions experienced by each ant, at each hop. Thus, O_3 can be mathematically expressed as:

$$O_3 = \min \sum_{i=1}^{n} \sum_{j \in \Omega_i} R_{ij}, \tag{4.13}$$

with Ω_i the set of nodes included in the path of the ant originated at node i, and R_{ij} the number of retransmissions experienced by it when attempting to get into node j. Since information on retransmissions is not available at the Network layer, in ACME, the number of retransmissions at each hop is estimated through the division of the actual packet delivery time, t_{ij}, by the ideal packet delivery time between two nodes with no retransmissions, t^*. The actual packet delivery time is determined based on the departure time from the sender node and the time of arrival at the recipient node. The ideal delivery time, on the other hand, is computed in OMNeT++ as the sum of the: (i) backoff time, T_b, (ii) time for clear channel assessment, T_c, (iii) time to switch from reception to transmission state, T_s, and (iv) transmission time, T_t, computed as the division of packet size to bitrate.

We acknowledge the fact that delays could be the consequence not only of collisions, but also of the queue at the receiver node. Thus, it is possible to perceive a slightly inflated value in the number of retransmissions with the estimation described in the previous paragraph. However, at the Network layer we are not able to differentiate delays caused by collisions from delays caused by queues. In addition, although the queue time implies a marginal energy consumption, a queue could be an indication of overload in that section of the subnetwork and thus, of predisposition to collisions.

Hence, although the queue time might have a slight influence on its value, we still find it relevant to use the packet delivery time as an indicator in the estimation of the number of retransmissions, and we stick to the approach described above.

The optimization of O_3 gives support to the MAC protocol by selecting a routing strategy with low or no conflict for channel access. This is achieved by making the ants simulate the behavior of the data packets and observe their interactions in order to mitigate the collisions. In this sense, our approach could be classified as a *fixed assignment* technique, where channel access is treated during a phase prior to the operation [173]. In order for the optimization to work, Eq. (4.13) must work as a deterministic function. In other words, if a given routing solution is generated by two different batches of ants, the interactions experienced by the ants on each group must be identical, and so the perceived values of O_3. If this condition were not met, the system would not be able to learn and optimize. Consequently, each time that O_3 is active in the optimization, ACME implements a deterministic MAC protocol. more specifically, CSMA/CA with constant backoff and contention window fixed equal to 1. This setting must be retained by the nodes during the operational phase in order to maintain the number of retransmissions achieved by the ants. Conversely, when O_3 is not included in the optimization, ACME implements CSMA/CA with constant backoff, but random contention window.

We recall that this objective is regarded as a complement of O_1 and O_2 in this thesis. Thus, we primarily devise to test it as a mechanism to obtain similar solutions to the ones obtained when the optimization strictly depends on O_1 and O_2, but with lower energy spent in retransmissions.

4.3.4 Handling node failures

Our system is ought to operate in an ad-hoc manner. Thus, it is crucial to have mechanisms prepared in case a node fails, enabling the subnetwork to keep operating. In this section we explain how node failures are handled in ACME.

Generally speaking, node failures could be of two types: foreseen and sudden. Foreseen failures occur due to battery depletion, while sudden failures might be caused by any damage in the device. In ACME, sensor nodes monitor their battery level and notify their neighbors before having a foreseen failure. Sudden failures, on the other hand, are identified by the neighbors of the affected node after a number of fruitless attempts of communication with it. More precisely, the sudden failure of a node is presumed by some of its neighbors if during three consecutive transfer rounds, either of ant or data packets, the node is unable to receive any message due to exhaustion of the maximum number of retransmissions. Either foreseen or sudden, once the failure is notified to all the neighbors of the affected node, one out of two sets of actions is implemented depending on the phase at which the subnetwork is at the

time of the failure; those are described below.

1. ***Node fails during optimization:*** the neighbors that detect the failure notify the checkpoint using the next preferred route indicated by their pheromone table. The checkpoint indicates the whole network to stop the current optimization, and triggers the reconstitution of the RPL trees and the neighbor discovery. This way, the protocol ensures that the costs to reach the subsinks and the list of neighbors refreshed to account for the node's failure. Finally, the checkpoint reboots the optimization to ensure that the learning process is not biased by the decisions made when the fallen node was active.
2. ***Node fails during operation:*** the neighbors that detect the failure notify the checkpoint using the next preferred route indicated by their pheromone table. The checkpoint indicates the whole network the ID of the fallen node and it is removed from every routing table. Finally, every node that had the fallen node as G-subsink replace it by the subsink with the next largest value in its pheromone table, and analogously for those who had the node as the next hop towards any destination. Once these nodes have updated their selections the operation moves on.

In any case, the checkpoint notifies the MULE as soon as possible about the nodes' failure and this last informs the sink so that the fallen node can be restored.

4.4 Further technical details

This section provides some complementary details on the initialization of pheromone loads, the normalization and calibration of the fitness function, the management of times in the workflow and some simplifications that can be made when only O_1 and O_2 are optimized.

4.4.1 Pheromone initialization

As explained in Step 1.4 of the step-by-step description of the protocol (Section 4.3.2) pheromone initialization is a procedure that provides the nodes with some base information on how good solutions are constituted. Thanks to this information, the system is able to concentrate from the beginning of the optimization on a region of the solution space containing good solutions. Otherwise, the system would have to spend various solutions, or even colonies, in acquiring that knowledge about the solution space through testing. Pheromone initialization in ACME is performed with support on two auxiliary greedy protocols, each able to generate a greedy solution optimal in terms of one of our two main objective functions, total path length (O_1) and subsink overload (O_2).

The subsink pheromone vector at each node is filled by taking into account both objectives. On the side of O_1, the subsinks closer to the node receive greater pheromone load than the ones farther to it. On the side of O_2, the subsink selected by the auxiliary protocol that seeks

ideal subsink balance is given a larger pheromone level than the others. Also the two subsinks closest to it receive a slightly higher subsink load than the others, letting the node know that those are also potential good G-subsink selections. The neighbor pheromone tables at each node only consider information from the protocol that seeks efficient routing (O_1). Thus, the pheromone values linked to each neighbor and subsink duple are assigned proportional to the number of hops separating them. This lets the node know which neighbor selections are the most efficient for reaching the subsink to which any given packet might be traveling.

Auxiliary protocols

The two auxiliary protocols used for pheromone initialization are briefly described below.

• ***N-Sub - focused on*** $\mathbf{O_1}$**:** greedy information on O_1 is obtained from the N-Sub protocol introduced in Chapter 3. Although this detail was omitted in the step-by-step description of the protocol to facilitate the comprehension of the workflow in ACME, N-Sub is actually embedded in the procedures of *filling of cost vectors* (Step 1.2) and *neighbor discovery* (Step 1.3). This protocol lets ACME rank the different subsinks in their role as potential G-subsinks for each node, based on the number of hops required to reach them.

• ***B-Sub - focused on*** $\mathbf{O_2}$ **:** greedy information on O_2 is obtained from a new protocol called MULE-assisted data collection with assured subsink balance (***B-Sub***), which we designed for this specific task. This protocol provides ideal subsink balance while maintaining reasonable performance in terms of O_1. The procedure starts with each node sending to each subsink two pieces of information: (i) the number of hops required to reach it; and (ii) a binary flag indicating if the subsink is the nearest destination for the node. The first subsink that ever had contact with the MULE awaits for a timer large enough to let all the messages arrive to the subsinks. Once the timer expires, this leading subsink sorts the IDs of the nodes who selected it as closest subsink in ascending order of the number of hops, and adds to its list of selected nodes, at much the number of IDs indicated by the ideal subsink load[2]. The IDs of the remaining nodes are sent to the next subsink, with the aim to check if those can be better accommodated by this subsink without surpassing the ideal load. If there are no remaining nodes, an empty message is sent instead. Once the second subsink has received the IDs of the extra nodes (if any), it becomes the leading subsink. Thus, it groups the IDs of the set of nodes that have it as the nearest subsink with those received from the first subsink. Then it does the same procedure as the first subsink to select a set of nodes and sends the IDs of the remaining nodes to the next subsink. This process is repeated until reaching the last subsink, as illustrated in Figure 4.7.

[2]We recall, from Section 4.3.2, that the ideal subsink load is computed as the division of the number of regular nodes to the number of subsinks.

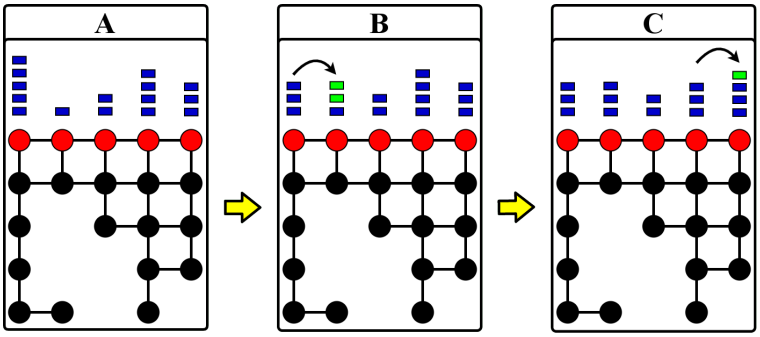


Figure 4.7: B-Sub auxiliary protocol for subsink balance - forward sweep. Black points represent regular nodes and red points represent subsinks. Blue blocks indicate the number of regular nodes temporarily assigned to each subsink (i.e., its temporary load). Green blocks indicate the nodes that one subsink shares in order to achieve the ideal subsink load.

Once a first sweep, where all the subsinks got the chance to pick some nodes, has been complete, the process starts again in the opposite direction. A backward sweep, going from the last subsink to the first one, is performed to distribute any excess of load concentrated at the last subsink among the ones that still have less than the ideal load. The result of this double sweep procedure is an assignment of sensor nodes to subsinks as even as possible. This process is illustrated in Figure 4.8.

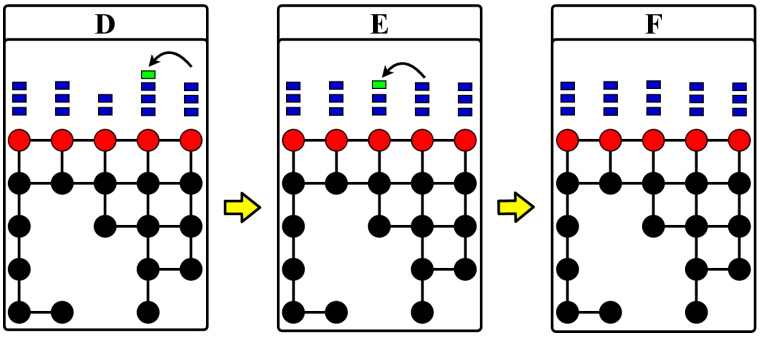


Figure 4.8: B-Sub auxiliary protocol for subsink balance - backward sweep. The colors and figures are interpreted the same as in Figure 4.7.

In the case where the ideal load is a decimal number (i.e., the number of nodes is not a multiple of the number of subsinks), a third sweep is performed to distribute the remainder of load among the first set of subsinks. During this last sweep, each subsink is only selects one remaining remaining node and passes the others to the next subsink. The round starts once again from the subsink that first got contact with the MULE, and moves through the other subsinks until the remainder of load has been fully assigned.

Once the load is distributed, each subsink sends to the checkpoint the IDs of the nodes it has selected, without including the ones that have it as the nearest subsink. Then, the checkpoint sends a message to each of those nodes using the routes provided by R-RPL, letting them

know their selected G-subsink. The nodes that do not receive any message interpret that they must keep the nearest subsink to them as G-subsink.

4.4.2 Normalization of objective functions

4.4.2.1 The need for normalization

Our multi-objective problem involves the optimization of three quantities with different units and ranges. This makes comparisons between objectives unfair since, the same value could indicate great performance in terms of one objective and awful performance in terms of the other. For example, consider a case where only objectives O_1 and O_2 are optimized. Consider a case where the best value of O_1 we can potentially find is 18, and the worst one is 100, while the same quantities for O_2 are 0 and 20, respectively. Assume that both objectives are given the same weights in the fitness function, indicating that the system administrator is equally interested about performance in both dimensions. If for instance we obtained a solution where both objectives were 19, such a solution would be great in terms of O_1, but at the same time awful in terms of O_2. A common way to overcome this difficulty is to transform the objective functions so that they become homogeneous in units and range. In this regard, the most popular approach in the literature is to apply a linear transformation involving the shifting and posterior scaling of the objective values [174, 175]. Following this approach, we define our normalization function as follows:

$$\mathcal{N}(\Delta_i; \lambda_i) := \mathcal{L}(\lambda_i \cdot \Delta_i), \quad \text{for } i = 1, 2, 3, \tag{4.14}$$

with $\Delta_i = (O_i(\boldsymbol{s}) - O_i^{\min}))$ denoting the difference between the objective value $O_i(\boldsymbol{s})$, obtained for some solution $\boldsymbol{s}$, and the ideal objective value for this objective, $O_i^{\min}$, the parameter λ_i denoting a scaling factor which helps to express the differences for all the objectives on a common unit, and finally $\mathcal{L}$ denoting a monotonic decreasing loss function which puts the scaled differences of all the objectives in the range $[0, 1]$ and specifies the way of penalizing those scaled differences. For the sake of readability, in the remainder of the manuscript we refer to the difference Δ_i and the scaled difference $\lambda_i \cdot \Delta_i$ as the ***deficit*** and ***scaled deficit*** of objective O_i, respectively.

4.4.2.2 Reasoning behind the deficits Δ_i

The operation $\lambda_i \cdot (O_i(\boldsymbol{s}) - O_i^{\min}))$ in Eq. (4.14) has a similar reasoning than the one used for the conversion of temperatures in °F to temperatures in °C. In the Celsius scale, the freezing point of water is measured at 0°C and the boiling point at 100°C, while in the Fahrenheit scale the freezing point of water is measured at 32°F and the boiling point at 212°F. Thus, the conversion from one unit to the other requires both, shifting and scaling:

°C = 5/9 (°F−32). In our problem, the total number of hops (O_1) of one solution is never zero. Conversely, it is possible to obtain solutions with zero subsink overload (O_2) and zero retransmissions (O_3). Thus, it is necessary to perform a subtraction on the value of $O_1(\boldsymbol{s})$ in order to put all the objectives in a common frame starting at zero, before multiplying them by the corresponding scaling factors. Otherwise, the result after scaling would be inflated for O_1 and the comparison with the other scaled objectives would be unfair. We use the sum of the number of hops separating all node nodes from their nearest subsinks as the value of $O_1^{\min}$. In order to make the checkpoint get this information, each node sends it its cost to reach the nearest subsink and then, the checkpoint adds the values reported by all the nodes. From the previous discussion, a valid choice for $O_2^{\min}$ and $O_3^{\min}$ is zero in both cases, which corresponds to the objective values obtained for solutions with no subsink overload and no retransmissions, respectively.

4.4.2.3 Selecting a suitable loss function $\mathcal{L}$

The fact of $\mathcal{L}$ being decreasing is just a matter of convention that facilitates the writing and interpretation of the formulas used for the computation of the fitness (Eq. (4.8)) and for global pheromone update (Eqs. (4.9) and (4.10)). Following this decision, we relate a higher fitness with a better solution. Anyway, a decreasing function would have also been a valid choice. In that case, we would just have had to interpret a lower fitness as a better one, and replace Ψ by $(1 - \Psi)$ in the equations for global pheromone update.

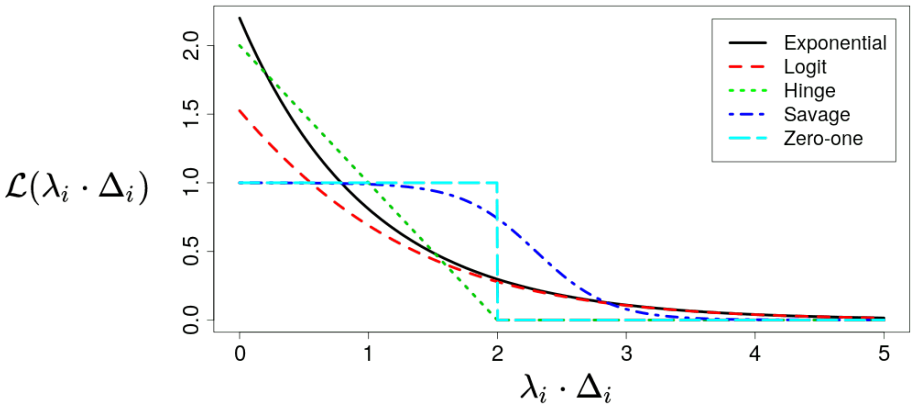


Figure 4.9: Five classic monotonic decreasing loss functions.

Some examples of classic monotonic decreasing loss functions are the Exponential, Logit, Hinge, Savage and Zero-one (see Figure 4.9). Among these, the Savage loss function presents some interesting properties that make it a leading candidate for application in our protocol. On the one hand, it allows to penalize gently small deficits and then smoothly switch to a more aggressive way of penalty as the deficit increases. This way of penalty fits well the needs of our problem, where a small number of unnecessary hops, misplaced packets among the subsinks, or a few retransmissions, might all cause minor deficits, which should not be

strongly penalized in order to let the protocol get a fair reward when it finds a solution close to the optimal one. On the other hand, this function is bounded from below by 0 and from above by 1, offering an implicit normalization mechanism and benefits in terms of interpretability (e.g., in the computation of the fitness) and stability of the formulation (e.g., for keeping the pheromone values controlled and prevent the method from getting stuck at local optima). In addition, unlike the Hinge and Zero-one functions, the Savage does not require the preselection of a specific deficit value from which to become zero. The Savage function has incorporated a progressive convergence to zero as the deficit grows.

To sum up, the Savage loss function comprises various of the good features found spread among the other mentioned alternatives. Considering this, and based on positive results during some preliminary optimization trials performed for the stabilization of the code, we select the Savage as the loss function for use in this thesis. Nevertheless, we remark that ACME remains able to implement alternative monotonic functions that the user might prefer.

Formally, the Savage loss function is defined as follows:

$$\mathcal{L}(\delta_i) := 1 - \frac{1}{1 + e^{-\beta(\delta_i - \delta_{\text{piv}})}}, \tag{4.15}$$

with the scaled deficit, $\lambda_i \cdot \Delta_i$, denoted by δ_i for readability, and β and δ_{piv} shape parameters controlling the loss rate and the location of the segment of fastest loss, respectively.

4.4.2.4 Resulting normalized fitness function

As a result of the normalization procedure explained along this section, we obtain a fitness function adapted to properly handle the different units and ranges involved in our multi-objective optimization problem:

$$\Psi = w_1 \left(1 - \frac{1}{1 + e^{-\beta(\delta_1 - \delta_{\text{piv}})}}\right) + w_2 \left(\frac{1}{1 + e^{-\beta(\delta_2 - \delta_{\text{piv}})}}\right) + w_3 \left(\frac{1}{1 + e^{-\beta(\delta_3 - \delta_{\text{piv}})}}\right). \tag{4.16}$$

In the next section, we explain how we set the values of the scaling factor λ_i and the Savage parameters β and δ_{piv}.

4.4.3 Calibration of the fitness function

In order to optimize our fitness function, ACME requires the definition of the λ_i, β and δ_{piv} parameters. Although the ideal value of each parameter might vary from one routing instance to the other, optimizing them for each instance seems implausible. Doing so would require running the ACO procedure multiple times for different combinations of the three

parameters in order to identify convenient values. Thus, we find this approach prohibitive due to the large amount of resources it would imply. In this thesis we follow a simpler process where only the parameter λ_i is customized to the characteristics of the routing instance at hand, and the parameters β and δ_{piv} are given unique values which are kept fixed over the instances. Moreover, the approach followed for λ_i does not imply any optimization and is straightforward to follow, which facilitates its use in practice.

4.4.3.1 Setting the scaling factor λ_i

As just mentioned above, in this section we propose a way to adapt the value of λ_i to the routing instance at hand. Our approach is an integration of two methods often used in the literature for setting up these kind of instance-dependent optimization parameters: (i) automatically deducing reasonable values from the extreme, single-objective optimal solutions to the optimization problem [176, 177]; and (ii) inferring reasonable values based on what the system administrator expects as the output of the optimization [178]. Since transmissions and retransmission, the corresponding units of O_1 and O_3, are equivalent in terms of cost to the system, we use the same scaling factor for deficits on both dimensions. Thus, we preset $\lambda_3 = \lambda_1$, and we concentrate on adjusting the values only of λ_1 and λ_2. Below we explain the steps of our method.

◇ **Step 1.** ***Automated deduction based on extreme solutions:*** in this method, some candidate λ_i values are deduced from the solutions obtained when O_1 and O_2 are optimized individually. In this technique, inherent from multi-objective optimization, the scaling factors are set so that the deficits that will appear during the ACO optimization are divided by the largest deficit expected to be observed for the corresponding objective. Mathematically, this reduces to the following simple expression:

$$\bar{\lambda}_i = \frac{1}{\bar{\Delta}_i^{\max}}, \tag{4.17}$$

with $\bar{\Delta}_i^{\max}$, for $i = 1, 2$, the deficit observed in objective O_i when the other objective is individually optimized. Note that we use a bar in this notation to differentiate the candidate scaling factor and max deficit from analogous quantities that will arise in Step 2.

Now we describe how this procedure is implemented in a subnetwork. First of all, the individual optimization of each objective is performed by means of the auxiliary N-Sub and B-Sub protocols that support pheromone initialization in ACME (Section 4.4.1). Once N-Sub is done, each node sends a tiny packet to its selected G-subsink so that all the subsinks are able to count the number of nodes assigned to them. Then, every subsink sends this count to the checkpoint who computes the total subsink overload, and subsequently the max deficit $\bar{\Delta}_2^{\max}$. The procedure for $\bar{\Delta}_1^{\max}$ is slightly simpler since, at the end of B-Sub, the subsinks already have the count of the number of hops separating them from the nodes

for which they will serve as G-subsinks. Thus, each subsink just sends the sum of number of hops of its assigned nodes to the checkpoint who computes the total number of hops, and subsequently the max deficit $\bar{\Delta}_1^{\max}$. Once the max deficits have been obtained by the checkpoint, it computes the first candidate scaling factors based on Eq. (4.17).

◇ **Step 2.** ***Inference based on the system admin (optional):*** in this second method, alternative candidate λ_i values are inferred from information provided by the system administrator. The process starts with the administrator informing the checkpoint the deficit values he would expect, or find reasonable for the instance at hand, if O_1 and O_2 were given equal optimization weights (i.e., $w_1 = w_2 = 0.5$). Let us denote those expected deficit values by $\hat{\Delta}_1^{\text{eql}}$ and $\hat{\Delta}_2^{\text{eql}}$, respectively. Then, an estimation of the max deficits is made by multiplying these expected deficit values by 2, yielding the quantities $\hat{\Delta}_1^{\max}$ and $\hat{\Delta}_2^{\max}$. The multiplication by 2 obeys a simple linear approximation of the deficit as a function of the optimization weight (see Figure 4.10). The estimation of the max deficits is followed by the computation of the corresponding scaling factors $\hat{\lambda}_i$, by means of an equation analogous to Eq. (4.17).

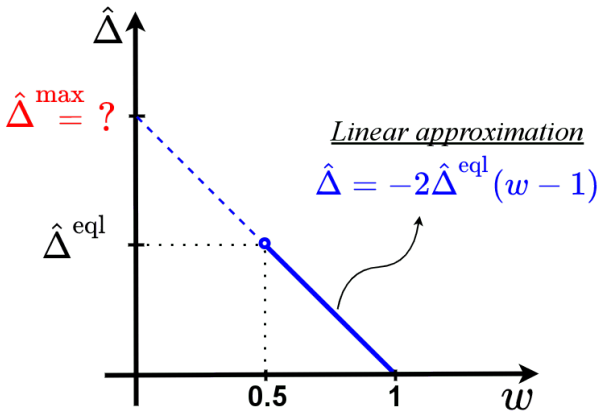


Figure 4.10: Linear approximation of the deficit, $\hat{\Delta}$, as a function of the optimization weight, w. The estimation of $\hat{\Delta}^{\max}$ is obtained by evaluating a weight of 0 in the linear expression, which yields $\hat{\Delta}^{\max} = 2\hat{\Delta}^{\text{eql}}$.

The implementation of this method in the subnetwork is quite simple as it only requires the system administrator to send its $\hat{\Delta}_i^{\text{eql}}$ expectations to the checkpoint. This could be achieved with support on the MULE, during its first visit to the subnetwork.

◇ **Step 3.** ***Integration (optional):*** once the two candidate pairs of scaling factors $\bar{\lambda}_i$ and $\hat{\lambda}_i$ have been obtained, the following rule is applied to determine which to use as definitive parameters during the optimization: use $\bar{\lambda}_i$ if $\bar{\lambda}_i \geq \hat{\lambda}_i$, otherwise use $\hat{\lambda}_i$. Note that decision is made individually for each of the two objectives. Thus, it is possible to end up using, for instance, $\bar{\lambda}_1$ and $\hat{\lambda}_2$ as definitive scaling factors. The reasoning behind this rule is that, if $\bar{\lambda}_i$ were lower than $\hat{\lambda}_i$, and it were used as definitive scaling factor, then the optimization

would value the units of this objective below the expectations of the system administrator. Our decision rule gives thus preference to higher valuations of the units of one objective, inferred from the information provided by the administrator.

Note that Step 2 requires some experience from the system administrator and its ability to indicate expected deficit values. We acknowledge the possibility of the system administrator not being able to provide those estimations for some routing instances, specially since now we are considering a broad class of subnetwork topologies. As a way to overcome this potential difficulty, we provide some reference values that we came up with after several hours of tests during the implementation and stabilization of the code. We observed consistent satisfactory results using values of $\hat{\Delta}_1^{\text{eql}}$ and $\hat{\Delta}_2^{\text{eql}}$ less than or equal to the 50% of the number of regular nodes and the 33% of the number of subsinks in the subnetwork, respectively. Even though, Steps 2 and 3 were marked as optional since the system would be able to automatically select the value of λ_i with support only on Step 1, if the system administrator ultimately prefers not to indicate any expectation.

4.4.3.2 Setting the parameters β and δ_{piv} of the Savage loss function

Now we briefly show how we set the parameters of the Savage loss function, β and δ_{piv}. The effect of each parameter is illustrated in Figure 4.11. The value of δ_{piv} determines at which value of scaled deficit the curve starts to make a strong distinction between good and bad solutions. The value of δ_{piv}, on the other hand, determines how sharp is the transition from soft to strong penalization as the scaled deficit increases. If any of the two quantities were set too low, small deficits would be strongly penalized and near-optimal solutions would be undervalued, thus inducing difficulties in the algorithm to converge. Conversely, if any of the two quantities were set too high, large deficits would not be properly penalized and ordinary solutions would receive too much reward, also inducing convergence issues and additionally hampering the ability of the protocol to find high quality routing strategies.

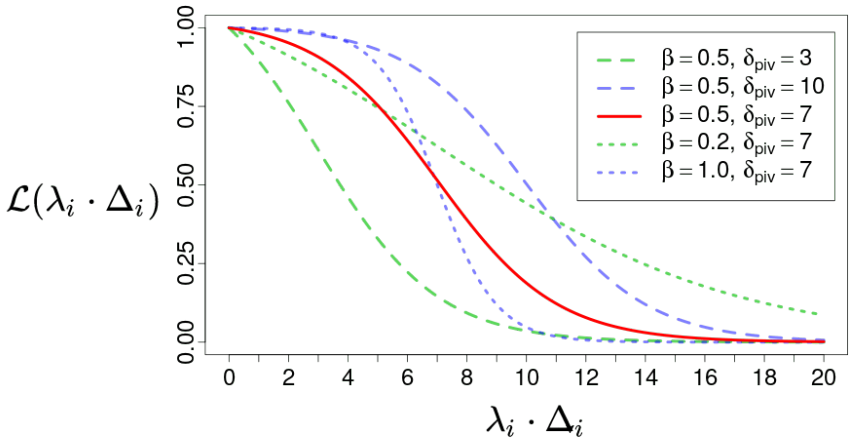


Figure 4.11: Effect of the parameters β and δ_{piv} on the shape of the Savage loss function.

Based on Figure 4.11, the combination ($\beta = .5$, $\delta_{\text{piv}} = 7$) seems to provide a good compromise between penalization and tolerance. With this combination, small scaled deficits are softly penalized, then the function progressively gets sharper as the scaled deficit starts to increase, and smoothly shrinks down to zero as the scaled deficit starts to get overly large. The other combinations seem prone to issues due to a lack or excess of penalty. Thus, we adhere to this combination for the experiments on ACME's performance, to come in Chapter 5.

4.4.4 Simplifications when only O_1 and O_2 are optimized

When only the total number of hops (O_1) and the subsink overload (O_2) are optimized, there is no reason for using a path different to the shortest one to communicate a given pair node-subsink. Thus, various simplifications can be made to the protocol in those cases:

1. The next hop for any packet is selected just by picking the neighbor with the lower number of hops to the relevant G-subsink, and not by the pseudo-random system of rules presented in Step 4 of the step-by-step description of the protocol.
2. The ant packets are reduced to the format presented in Figure 4.12. The fields **`Src_ID`** are removed since it would no longer be needed during neighbor selection, and the fields **`Nb_rets`** and **`S_time`** are removed since those are exclusive of the optimization of O_3.

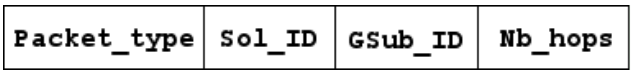


Figure 4.12: Simplified format of an ant packet

3. The neighbor pheromone matrix is simplified. Since the index of the source is no longer relevant in the selection of the neighbor, the matrix is reduced from a tridimensional structure with elements τ_{ijk}, to a bidimensional one with elements τ_{jk}. As explained above, the pheromone levels are no longer used for neighbor selection during the optimization, however, the pheromone tables are built just to support the recovery of the system in case of node failures (see Section 4.3.4).

Note that although these simplifications can be made, the general version of ACME presented in Section Section 4.3.2 is greatly valuable. That version serves as a reference on how objectives that depend on local information about the neighbors (e.g., their residual energy) can be incorporated in the optimization.

4.5 Sketch to some key ACME extensions

In this section we concentrate on two potential extensions of ACME which are of primary importance based on the interest that the research community has demonstrated on them:

1. Generalization to nodes with heterogeneous traffic profiles;
2. Incorporation of data aggregation to boost energy efficiency.

The content of this chapter must be perceived as a sketch for future research and not as finished work. However, we made an effort to provide the ideas as developed as possible to enable the interested researcher to quickly move to the implementation of the extension with little or no intervention in the method. Hence, although considerably simpler in description and analysis than other technical sections of this thesis, we find the present section of significant value for researchers working in related topics.

4.5.1 Nodes with heterogeneous traffic profiles

The motivation for this extension in the context of ASAS is illustrated in Figure 4.13. The nodes placed at different points of the sensing field could be assigned different variables to monitor, resulting in heterogeneous packet sizes. The methods and experiments presented so far simplify this aspect by assuming that the nodes collecting less variables fill the packet with dummy values upon reaching the packet size of the nodes that collect the most variables. This approach is perhaps the most direct way to circumvent the challenging task of balancing subsink load in settings involving heterogeneous packet sizes. However, it is inefficient in terms of energy spent since the packets with less number of usable variables are transferred at higher cost than they should due to the inflation of the packet size.

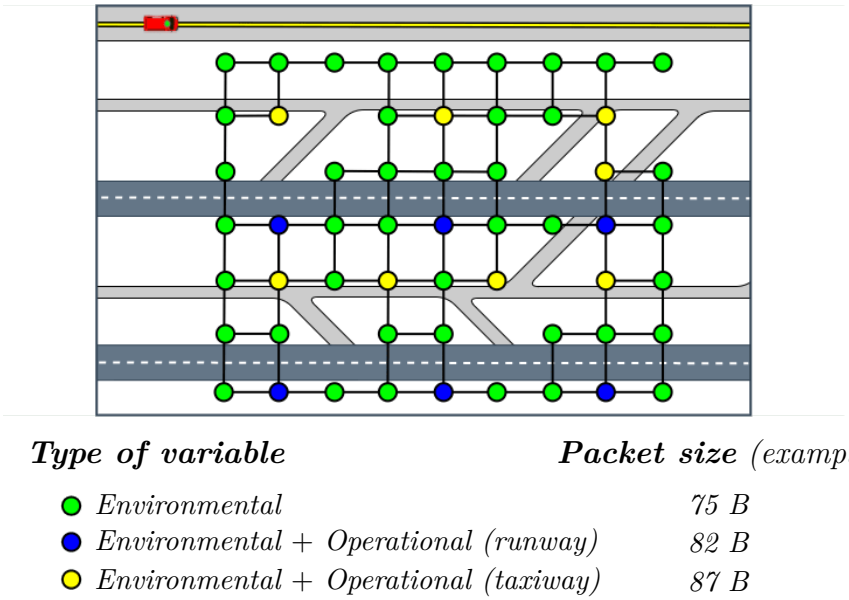


Figure 4.13: Illustration of heterogeneous traffic profiles in ASAS.

The extension to heterogeneous packet sizes implies the adaptation of the B-Sub protocol, used as a auxiliary protocol for the initialization of pheromone tables within ACME, and also the adaptation of ACME itself. The version of these two protocols simply use the number of packets assigned to each subsink as its load, neglecting the fact that the packets could be of different size. The proposed extensions are presented below.

4.5.1.1 Adapting B-Sub

The modifications required in this protocol are listed below:

1. The information sent by the sensor nodes to the subsinks should include not only the number of hops and the binary flag, but also the size of packet they manage;
2. The subsinks should not sort the nodes in ascending order of the number of hops, but rather in ascending order of the division of number of hops to packet size;
3. The ideal subsink load should not be computed as the division of number of nodes to number of subsinks, but instead as the division of the sum of packet sizes to the number of subsinks;
4. A third round should not be longer performed when the ideal load is decimal, but instead, a new phase should be added to the procedure involving multiple rounds that go until all the nodes have been assigned one subsink. During those rounds, the subsinks will keep grabbing nodes and passing the remaining ones, but this time each subsink would select only one node at a time, and would be allowed to exceed the ideal load. More precisely, each node would select, among the nodes it receives, the one with the lowest ratio of number of hops to packet size. This new procedure is needed since distribution of nodes over subsinks during the first two rounds might end up with some nodes unassigned, whose load might not be able to be accommodated by any subsink without exceeding the ideal subsink load.

The proposed adaptation is inspired in the classical Longest Processing Time first (LPT) rule, an heuristic method which seeks load balancing in the frame of jobs scheduling in identical parallel machines [179]. This rule assigns the jobs to the machines one by one starting from the one with the longest processing time and ending with the one with the lowest processing time. At each step it assigns the next job to the least loaded machine. The reasoning behind starting with the jobs with longest processing time is that the ones with shortest processing time are easier to accommodate at the end and help to fill the holes (like the smaller shopping boxes in the car trunk). The method had to be adapted here in order to take into account the distance separating the nodes from the subsinks, a feature which is not present in the job scheduling setting. Our method is expected to deliver close-to-optimal solutions in terms of subsink load, and reasonable results in terms of total number of hops.

4.5.1.2 Adapting ACME

The modifications required in ACME are the following:

1. The modified version of B-Sub explained above should be used instead of the original one during the initialization of the pheromone tables;
2. Objective O_2 should be computed in terms of the sum of packet sizes of the nodes assigned to the different subsinks, instead of the count of the number of nodes assigned to each subsink.

4.5.2 Data aggregation

The idea of data aggregation is to combine the data gathered by multiple sensors in order to reduce the amount of traffic and thus the energy spent of the network. Different aggregation schemes can be adopted. Notably, the literature emphasizes on the following two:

- *Data re-packing:* data payloads of several packets are combined into one packet with single header [180];
- *Data compression:* the data is projected onto a space of lower dimension while preserving its main statistical or geometrical properties [181]

Below we present a scheme that can be followed to attach any of these two techniques to our methods. Our proposal considers an implementation post-optimization since the integration with the ACO procedure would require the treatment of several challenging aspects like the synchronization of the nodes to form the aggregated ant packets and the adaptation of the objective functions, which constitute dense material for future research and fall beyond the scope of this thesis.

Coordination

Once the optimization in ACME has finished, all the nodes send a message to the G-subsink they selected during the optimization. Each node stores the ID of the source node of each packet it receives and then groups those IDs by G-subsink. Then, data collection starts.

Implementation

The nodes that did not receive any packet during the coordination are those placed at the extremes of the routing trees, i.e., the leafs. Those start the process by sending their collected data upon their tree. Each time a node receives a packet, it checks its G-subsink and proceeds as follows;

- If the node has already received all the packets destined to that G-subsink, the node aggregates all their information and sends the aggregated packet to the next hop;

- Otherwise, it stores the packet and keeps waiting for the remaining packets destined to the same subsink to come.

  Remark: in the particular case where the node receives the last packet that goes to the same G-subsink as itself, the node must also aggregate its own data to the packet before forwarding it.

In the next chapter we evaluate the performance of ACME through computer experimentation. We remark that the two extensions proposed here are not implemented yet in our code and will not be evaluated during the experiments.

Chapter 5

ACME's performance assessment

The chapter in brief

In this chapter we present the computational experiments carried out to validate our ACO-based routing protocol, ACME. The experiments are divided into three parts, denoted by **Part$_1$**, **Part$_2$** and **Part$_3$**. **Part$_1$** and **Part$_2$**, presented in Sections 5.2 and 5.3, respectively, concentrate on the optimization phase, where each node is assigned a G-subsink and the routing tables are built. This phase does not involve neither data collection nor data transfer to the MULE. More precisely, we evaluate ACME's ability to provide high quality routing solutions, consistent with the specified optimization weights, for a variety of irregular sub-network configurations. **Part$_1$** exclusively considers our two main objective functions, total path length (O_1) and subsink overload (O_2). **Part$_2$**, on the other hand, explores the benefits of incorporating our third objective function, total number of retransmissions (O_3), to the optimization. In **Part$_3$**, presented in Sections 5.4, we move our focus to the operational phase, where data collection and contact with the MULE take place. This section compares the efficiency in the interactions with the MULE achieved by ACME with that offered by the N-Sub protocol, presented in Chapter 3.

Contents

5.1 Experiment setup

This section describes all the parameters which are common to the three parts of the experiment, **Part$_1$**, **Part$_2$** and **Part$_3$**. This includes the subnetwork configuration (base topology, obstacles and size of field), the protocol stack and device configuration, the parameters for the ACO algorithm, and the simulation environment. Further parameters necessary for the experiment involving the MULE in **Part$_3$** are provided later in Section 5.4.2.

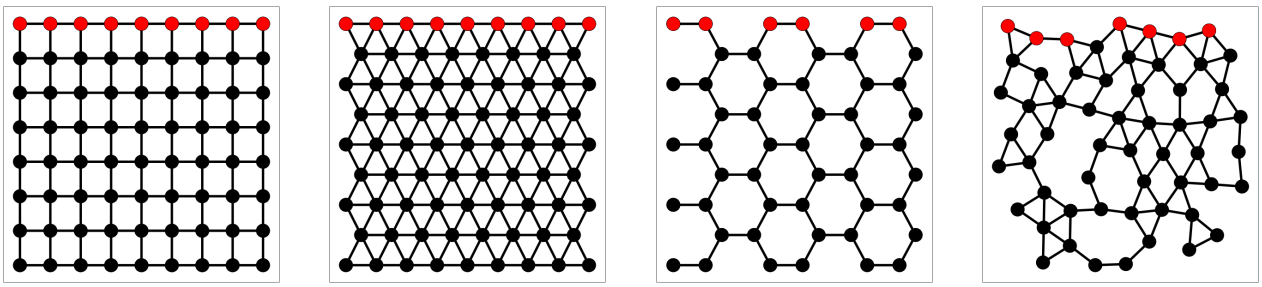
5.1.1 Subnetwork configuration

In Chapter 3 we conducted an experiment to compare the performance of N-Sub over C-Sub as routing protocols in the ASAS context. The results showed a clear superiority of N-Sub over C-Sub. Indeed, N-Sub results ideal when the subnetwork has a regular grid-like topology, since in those cases it is able to provide optimal subsink overload (O_2) as a side effect of minimizing the total path length (O_1). However, for irregular subnetwork structures (e.g., where some nodes are missing due to obstacles) or regular but non-square subnetwork structures (e.g., a triangular grid), the distribution of load over the subsinks delivered by this protocol might be unsatisfactory. Motivated by the variety of subnetwork topologies that might appear in the ASAS application, we designed ACME as a tool to deal with the trade-off between O_1 and O_2 when the subnetwork structure is anything different from a regular square grid. Thereby, in this set of experiments we make strong emphasis on testing ACME's performance for a variety of irregular subnetwork configurations. In this regard, we consider a total of 32 instances resulting from the combination of various levels of the following features: (i) base topology, (ii) type of obstacle or degree of connectivity, and (iii) size of the sensing field. Those are described below.

Base topology

This feature refers to the underlying pattern followed by the distribution of sensor nodes over the sensing field. We call it *base topology* since the pattern is altered by the obstacles that we will discuss later. We consider four different base topologies, three of them based on regular tessellations and a fourth one based on a random distribution of nodes. Tessellations are constituted by repeating patterns of polygons covering an area with no gaps or overlaps. In particular, *regular* tessellations are those where only one type of regular polygon is used to fill the area (e.g., only equilateral triangles). To define the base topology for a communication network, one can deploy the nodes at the vertices of the polygons while ensuring that their sides remain shorter than the communication range of the nodes to ensure the connectivity (see e.g., [182], [183], [184] and [96]). For the experiments, we use the *square*, *triangular* and *hexagonal* tessellations, illustrated in Figures 5.1a, 5.1b and 5.1c, respectively. For the random distribution, we consider a range-aware filling approach where the nodes are deployed

at random locations within the sensing field while ensuring to fulfill the conditions that: (i) there are no isolated nodes, and (ii) there are no pair of nodes separated by a distance less than the communication range. The pseudocode used for the generation of random subnetworks is provided in Appendix B. An example of our random base topology is shown in Figure 5.1d.



(a) Square. **(b)** Triangular. **(c)** Hexagonal. **(d)** Random.

Figure 5.1: Four base topologies used for ACME's validation. The red points at the top of each subnetwork represent the subsinks. Black points represent regular nodes.

We remark that the subnetworks in our ASAS system are independent since there is no data transfer between them. Thus, our experiments consider only one subnetwork at a time, and focus on the solution of the data routing problem for it. Each of the frames displayed in Figure 5.1 already represents a single subnetwork to be used in one instance of the experiments. In practice, if multiple subnetworks made part of the system, each of them would use ACME independently to constitute its data routing strategy.

Type of obstacle

Aside from the random base topology, the subnetwork structures presented in Figure 5.1 are still regular. This type of ideal pattern is sometimes not possible to achieve in reality due to the presence of obstacles that prevent the deployment of some nodes at their supposed location in the grid. This condition was illustrated in Section 3.1 for the ASAS application, where the runways, taxiways and service buildings are some examples of structures that may introduce irregularity in the subnetwork structure. Thus, in order to make the experiment more realistic, we consider different types of alterations of the subnetwork structure. For the square and triangular base topologies, we consider small and big obstacles:

- ***Small obstacles:*** those remove individual nodes at different parts of the subnetwork. Each small obstacle is represented by a missing node in the square or triangular base topologies as shown in Figure 5.2. The nodes to remove are chosen randomly while ensuring that no node results isolated. We let obstacles of this type vary in number between 4 and 6. For the size of sensing field and number of nodes used in our experiments, those amounts of small obstacles generate subnetworks not so connected that the effect of the irregularity

is negligible, nor so disconnected that the solution becomes obvious due to the scarcity of routing alternatives.

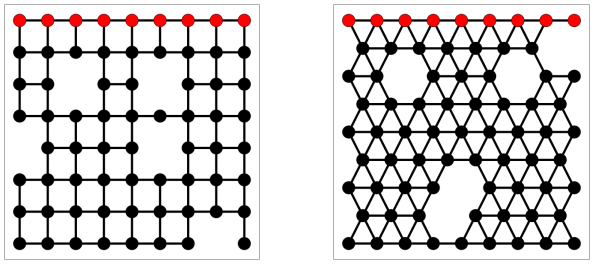


Figure 5.2: Example of Square and Triangular topologies with five small obstacles.

- ***Big obstacles:*** those remove entire blocks of multiple adjacent nodes located at the same sector of the subnetwork. We only consider subnetworks affected by one big obstacle, and we let its location vary among: center high, right center and bottom left (see Figure 5.3). A big obstacle makes some subsinks become relatively far choices for multiple nodes, stressing the conflict between O_1 and O_2. Similarly as we did with the number of small obstacles, we fix the size of the big ones in order to obtain a suitable degree of connectivity in the subnetworks for the validation of our method.

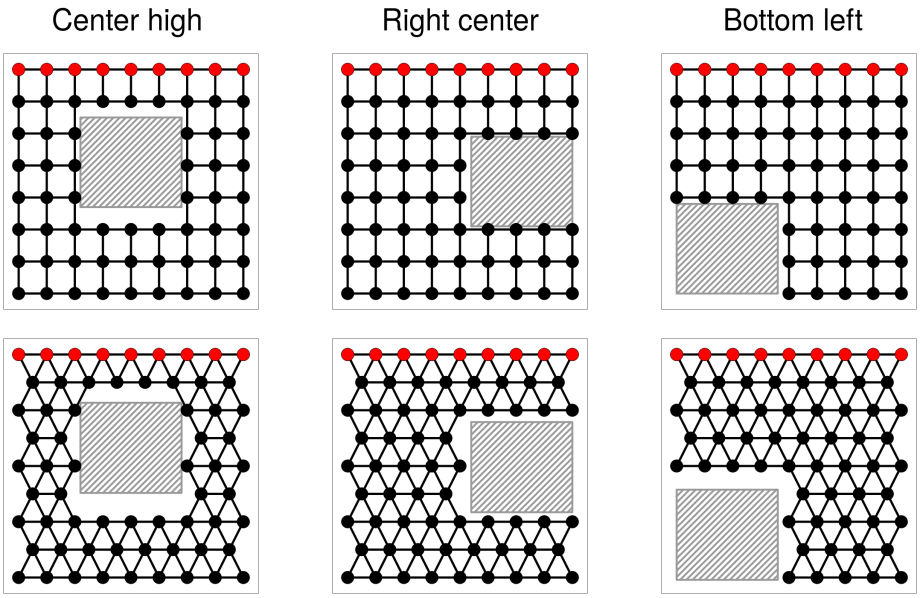
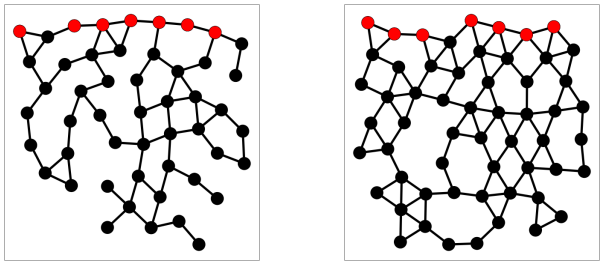


Figure 5.3: Example of Square and Triangular topologies with a big obstacle placed at the three locations used in the experiment.

With the hexagonal and random base topologies, we follow a different approach. As can be noted in Figure 5.1, the hexagonal topology is much less connected (number of neighbors per node) and densely filled (number of nodes per unit of area) than the other base topologies. As a consequence, both the small and big types of obstacles often result in a strong reduction

in the number of routing alternatives to a point where the problem becomes trivial to resolve. Thus, although the problem with obstacles in the hexagonal topology could be resolved by ACME, we leave this possibility out of the experiment and we strictly consider the hexagonal topology in obstacle-free conditions. The random topology, on the other hand, naturally leaves some zones of the sensing field unfilled without the need of adding obstacles. Instead of using big or small obstacles to increment the irregularity on this type of topology, we find it better to use a so-called connectivity degree:

- ***Connectivity degree:*** it refers to a coefficient in $[0, 1]$, used by the algorithm presented in Appendix B to control how likely are two forming branches of nodes to get connected. A coefficient of 0 would produce a tree-like structure, while a coefficient of 1 would generate a strongly connected and densely filled subnetwork. Our experiments consider subnetworks with connectivity degree of 0.4 and 0.8. Examples of the resulting subnetworks are provided in Figure 5.4.



(a) Con. deg.: 0.4 **(b)** Con. deg.: 0.8

Figure 5.4: Example of random topology with two different values of connectivity degree.

Sensing field size

On top of the base topology and the type of obstacle, we considered sensing fields of two sizes: *small* (800 m x 700 m) and *big* (1000 m x 900 m). Those values were set having in mind a potential subnetwork to be deployed over a portion of the movement area[1] of a midsize airport. For the sake of exposition, in Figure 5.5 we provide an example of a subnetwork over a big sensing field. The picture shows that the subnetwork is large enough to cover two runways in the direction perpendicular to them, and a significant distance in parallel direction considering that most commercial runways have a length between 1.5 and 5.5 km [185]. Note that some nodes in the sample subnetwork of Figure 5.5 are missing since their expected coordinate in the grid falls over a taxiway. Also note that the subsinks are located by the side of the service road (in yellow) so that they have contact with service vehicles (the MULEs) as frequently as possible.

[1]Recall, from Chapter 1, that the movement area is the section of the airport used for takeoff, landing and taxiing.

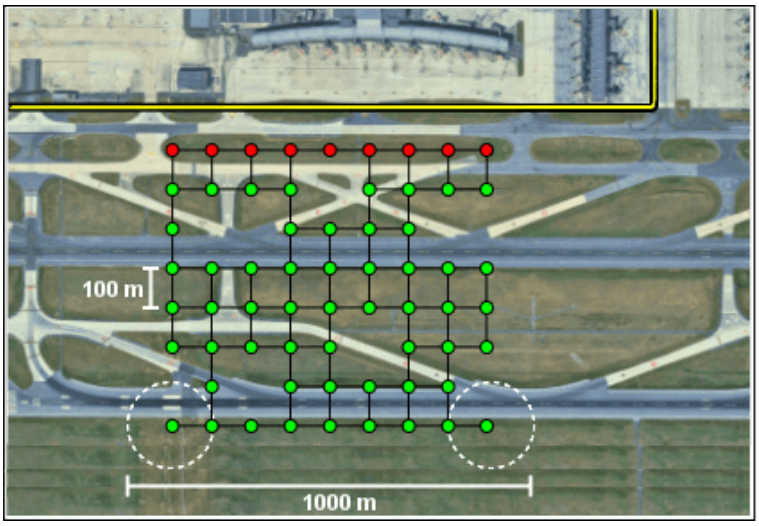


Figure 5.5: Big size sensing field based on dimensions of a section of a real airport. Red nodes at the top represent the subsinks, the white circumference indicates the extension of the communication range of a node and the yellow line at the top denotes the service road used by the ground service vehicles playing the role of MULEs.

Depending on the base topology, the same number of nodes might be used to fill sensing fields of very different size. Thus, in order to make standard the comparison in performance for different base topologies, we define the subnetwork size in terms of the size of the sensing field. Then, as a general rule, we fill the sensing field of any size with nodes until we obtain a subnetwork as connected as possible, consistent with the selected base topology.

▷ ***Remark 1:*** the instances used in our experiments are not intended to represent the layout of a subnetwork for some particular real airport. Moreover, we do not have as objective to conduct a sensitivity analysis to determine which factor (base topology, type of obstacle, connectivity and size of the sensing field) has a greater impact on the performance of the protocol. Instead, we test different levels of those factors in order to obtain a variety of subnetwork configurations allowing us to evaluate the ability of our protocol to adapt to the characteristics of the instance to solve. Our objective is to show that ACME would be a suitable solution for a general class of airport layouts.

▷ ***Remark 2:*** we do not include any instance where the square base topology is tested with no obstacle. The reason for this is that the optimal solution for such an instance in terms of objective O_1 is the same as the one in terms of objective O_2; thus, the variation of the optimization weights is pointless. This scenario corresponds to the type of topology studied in Chapter 3, whose routing can indeed be optimally solved (in terms of both objectives) by any of our two auxiliary protocols, N-Sub and B-Sub.

▷ ***Remark 3:*** we perform a single run for each instance involving a random base topology (i.e., we do not make replicates). Replicates are not imperative in our experiment since we are not interested in characterizing the performance of the protocol for any particular

type of subnetwork. Instead, we focus our efforts in checking its ability to resolve instances presenting a variety of heterogeneous characteristics while being consistent with the weights specified by the user; thus, the role of the instances involving a random base topology is merely to test if ACME is able to manage properly a less regular type of subnetwork than the ones resulting from the other base topologies.

5.1.2 Protocol stack and device configuration

The architecture implemented in the nodes is similar to the one used in Chapter 3. It is displayed in Figure 5.6. The Physical and MAC layers are based on the IEEE 802.15.4 standard. We use the maximum bitrate and communication range enabled by this standard, which are of 250 kbps and 100 m, respectively. We leave the maximum queue length at 100 packets which is the default value for this parameter in OMNeT++. This selection of parameters is consistent with the choices made in various previous studies [153, 154]. Acknowledgement messages are enabled and the maximum number of transfer attempts is fixed at 7. After that, a loss due to collision is registered and the message is dropped. The CSMA/CA procedure with constant backoff is implemented in the MAC layer to help prevent collisions. We recall that the contention window is fixed equal to 1 when O_3 is active in the optimization (i.e., in $\mathbf{Part_2}$ of the experiments). ACME is used as the control plane protocol next to IPv6 as the data plane protocol. As a complementary action to mitigate collisions, the first ant packet at each node is generated with a tiny random delay uniformly distributed in $[0, 1]$ s (see Appendix A). We use UDP in the Transport layer since the loss of an environmental or operational data packet is not critical in the ASAS application, and it omits the connectivity control procedures in TCP. The gathering process implemented in the Application layer and the architecture of the MULE are specified in Section 5.4.2, since they do not have any impact on $\mathbf{Part_1}$ and $\mathbf{Part_2}$ of the experiments.

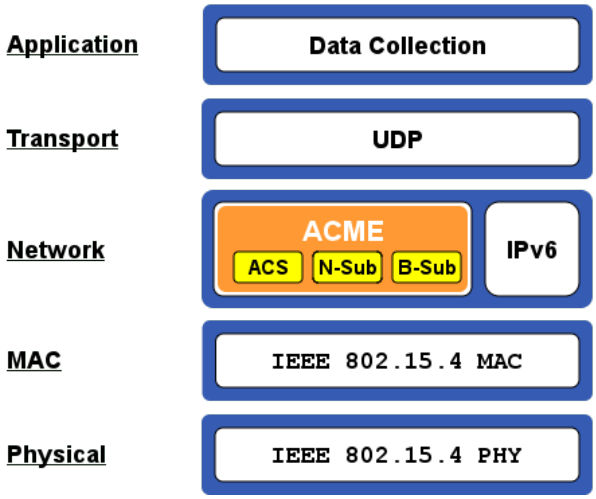


Figure 5.6: Protocol stack for ACME. ACS stands for Ant Colony System, and corresponds to the module responsible for the optimization.

We take as reference a TelosB mote, often used in WSNs to simultaneously monitor multiple variables such as temperature, humidity and light intensity (see e.g., [155, 156]). A TelosB mote is equipped with a CC2420 radio compatible with IEEE 802.15.4 standard, operating in the 2.4 GHz band. The radio transceiver of the node consumes 18.8 mA (56.4 mW at 3 V) in reception and listening mode, 17.4 mA (52.2 mW at 3 V) in transmission mode, and 1 μA in sleep mode (0,003 mW at 3V) [158, 159]. Each node uses two AA rechargeable batteries, whose energy load is typically set to 18720 Joules [151, 160]. The memory capacity of a TelosB goes up to 1 MB [186].

5.1.3 Heuristic parameters

Our routing protocol relies on a set of parameters typical of any ACO-based algorithm. Those parameters control the pheromone evaporation rate, the learning reinforcement rate, the degree of exploration, and the number of solutions and colonies. In this thesis, we fix these values based on analogous quantities used by Dorigo et al. in the introductory paper of the Ant Colony System [170]. More precisely, we select the triple $(0.1, 0.1, 0.9)$ for the parameters (ρ_l, ρ_g, q_0), related to the the subsinks, and the parameters (ϕ_l, ϕ_g, q_0), related to the neighbors. During a set of preliminary informal tests, we found those values offer a good trade-off between solution discovery (exploration) and convergence (exploitation). It is worth noting that the selected values fall in the range of ideal values found by multiple authors in various disciplines for analogous parameters (see e.g., [187, 188, 189]). Our preliminary trials also allowed us to identify a convenient setup at 10 colonies and 10 solutions per colony. This number of colonies enables our protocol to achieve high quality routing solutions at a reasonable energy consumed.

5.1.4 Simulation environment

We implemented all the codes for the generation of the subnetwork layouts in the statistical software R [190]. On the other hand, we used the discrete event simulator OMNeT++ [161] for the implementation of ACME and the execution of our simulations.

5.2 Optimizing total path length (O_1) and subsink overload (O_2)

This section presents **Part$_1$** of our experiment, dedicated to the assessment of ACME's performance when O_1 and O_2 are optimized. We first address a simple example aimed to show the reader: (i) how the value of the scaling factors for the objective functions are set, (ii) how the optimization evolves and the deficits are reduced as the iterations are complete, and (iii) how a solution provided by ACME looks like compared to solutions fully oriented to either O_1 or O_2. Then, we proceed with the analysis of ACME's robustness to changes in the structure of the subnetwork.

5.2.1 Simple example optimizing O_1 and O_2

In this example, we consider a subnetwork with square topology and six small randomly located obstacles, covering a big sensing field. Such a subnetwork is illustrated in Figure 5.7.

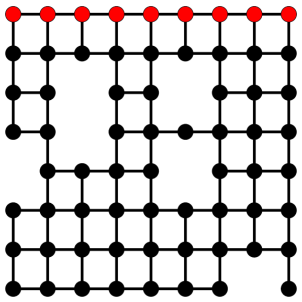


Figure 5.7: Subnetwork to show ACME's performance when optimizing O_1 and O_2.

Below we present the calibration of the scaling factor λ_i for this example. Then we proceed with the optimization of the routing strategy.

5.2.1.1 Setting the scaling factors λ_1 and λ_2

For the calibration of the scaling factors in this chapter, we adhere to the three steps described in Section 4.4.3.1:

◇ **Step 1.** ***Automated deduction based on extreme solutions:*** we let the system perform N-Sub and B-Sub for this instance in order to obtain worst case values for the deficits of objectives O_2 and O_1, respectively. The procedures performed by these protocols yielded $\bar{\Delta}_2^{\max} = 13$ and $\bar{\Delta}_1^{\max} = 25$, respectively. Thus, from equation Eq. (4.17) we get

$$\bar{\lambda}_1 = \frac{1}{\bar{\Delta}_1^{\max}} = \frac{1}{25} \qquad \text{and} \qquad \bar{\lambda}_2 = \frac{1}{\bar{\Delta}_2^{\max}} = \frac{1}{13}.$$

◇ **Step 2.** ***Inference based on the system admin:*** in this step we use the reference values we offered in Section 4.4.3.1, which are $\hat{\Delta}_1^{\text{eql}} = 50\%$ of the number of regular nodes, and $\hat{\Delta}_2^{\text{eql}} = 33\%$ of the number of subsinks in the subnetwork. Our example involves 57 regular nodes and 9 subsinks, thus yielding $\hat{\Delta}_1^{\text{eql}} = 28.5$ and $\hat{\Delta}_2^{\text{eql}} = 3$. We multiply these values by 2 to obtain $\hat{\Delta}_1^{\max} = 57$ and $\hat{\Delta}_2^{\max} = 6$. Finally, we apply an equation analogous to Eq. (4.17) to obtain

$$\hat{\lambda}_1 = \frac{1}{\hat{\Delta}_1^{\max}} = \frac{1}{57} \qquad \text{and} \qquad \hat{\lambda}_2 = \frac{1}{\hat{\Delta}_2^{\max}} = \frac{1}{6}.$$

◇ **Step 3.** ***Integration:*** finally, we integrate the two results by applying the simple rule: use $\bar{\lambda}_i$ if $\bar{\lambda}_i \geq \hat{\lambda}_i$, otherwise use $\hat{\lambda}_i$. Thus, we set $\lambda_1 = 1/25$ and $\lambda_2 = 1/6$.

Just for convenience in the presentation of the results, we use

$$\lambda_1 = \frac{6}{25} \qquad \text{and} \qquad \lambda_2 = 1,$$

which are proportional to the values obtained above and thus, have the same effect in the optimization. The idea of using these alternative values is that the deficits of objective O_2 are directly interpretable as the number of nodes wrongly assigned to subsinks (see Section 4.3.3).

5.2.1.2 Evolution of the optimization

Now we present some results of the optimization performed by ACME in our example, which helps to verify that its learning mechanism is working properly. First of all, in Figure 5.8 we display the evolution of the fitness along the colonies. More precisely, we show the fitness of each individual solution tested, the median fitness per colony and the best fitness found so far up to each colony. As the colonies pass, the maximum and median fitness progressively improve. In the first colonies, the range of fitness values is larger, but it gradually shrinks as the exploration converges to the final solution. Although the global max fitness and the median fitness per colony seem relatively flat along some segments, the zoom presented in the panel at the bottom shows that both values were consistently improved along the optimization, colony after colony. ACME was able to reach a fitness of 0.938. In the real subnetwork, this optimization would have taken a little less than 10 min. Figure 5.8 verifies ACME's ability to learn and converge to a high quality solution.

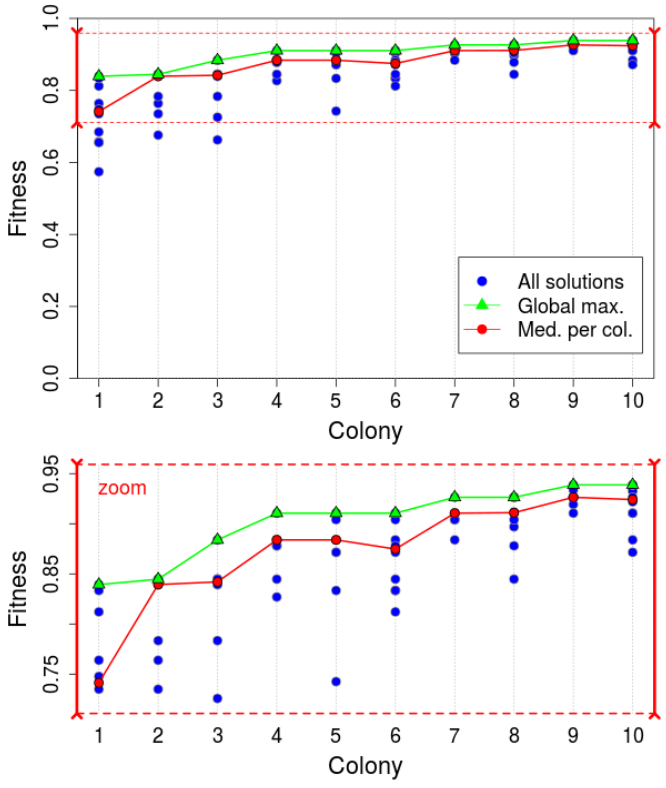


Figure 5.8: Evolution of fitness when optimizing O_1 and O_2.

As a complement to Figure 5.8, in Figure 5.9 we report the scaled deficit for each objective in each solution along the colonies. This time we also report the median scaled deficit per colony for each objective. Since the optimization was made with an even distribution of weights (i.e., $w_1 = w_2 = 0.5$), ACME simultaneously pursued the minimization of each objective and the construction of a solution with similar scaled deficits for both objectives. It is also notable that the dispersion in the scaled deficits of both objectives shrinks as the colonies pass. This shows the convergence of the algorithm as an effect of the exploitation mechanism.

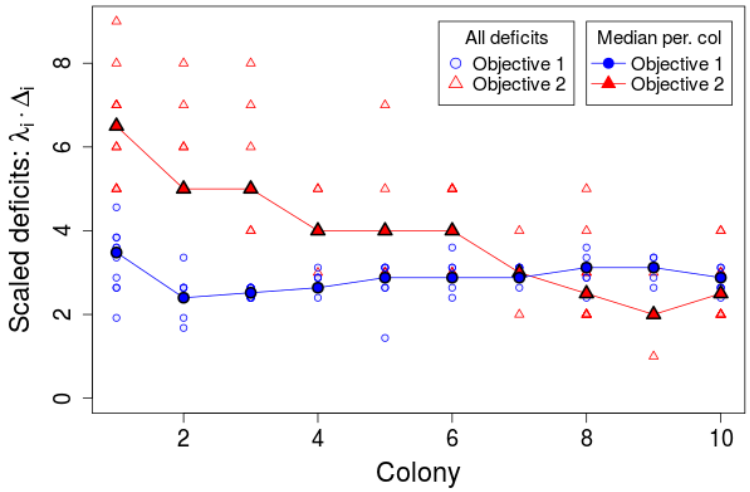


Figure 5.9: Evolution of scaled deficits when optimizing O_1 and O_2.

5.2.1.3 Visualizing a solution provided by ACME

The solution found by ACME for this example is displayed in the middle column of Figure 5.9. Each row shows the group of nodes (in green) that selected each G-subsink (in red) and the multi-hop paths generated for them (in blue). In the left column we display the optimal solution in terms of objective O_1, produced by N-Sub. Similarly, in the right column we show the optimal solution in terms of objective O_2, produced by B-Sub. At the top of each frame, we report the total number of hops included in the paths of the nodes assigned to the corresponding subsink, along with the number of nodes assigned to it (i.e., the subsink load). This figure shows how ACME, using equal optimization weights for both objectives, is able to find an intermediate solution providing an equilibrium between both objectives.

In this example, the deficits in number of hops and subsink overload, in the solution provided by ACME, are 11 and 2, respectively. Since the scaling factors used in the optimization were $\lambda_1 = 6/25$ and $\lambda_2 = 1$, the resulting scaled deficits are 2.64 and 2. Note that ACME found a solution where the two scaled deficits are very similar, which is consistent with the optimization weights we specified.

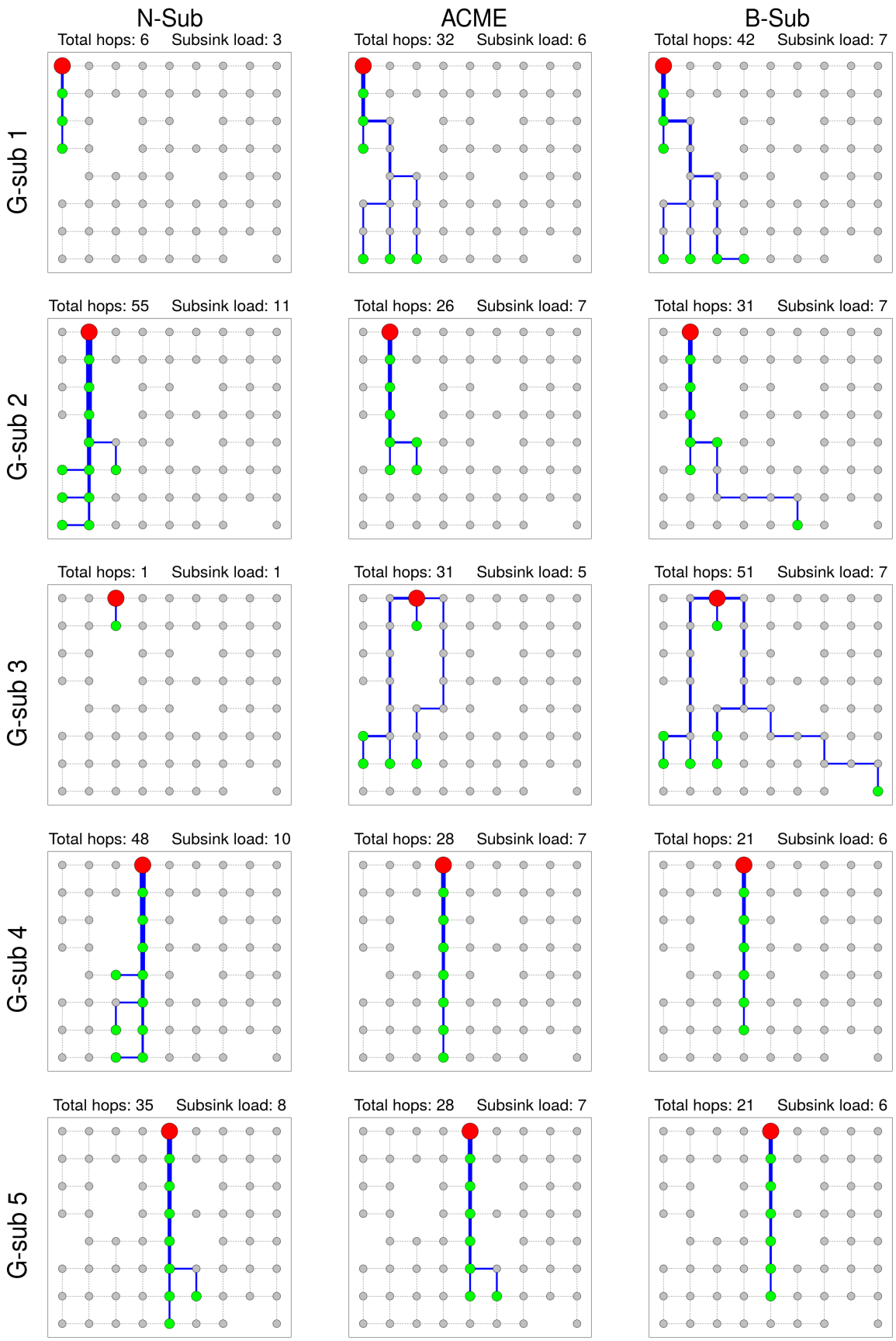


Figure 5.10 - *Continued on next page*

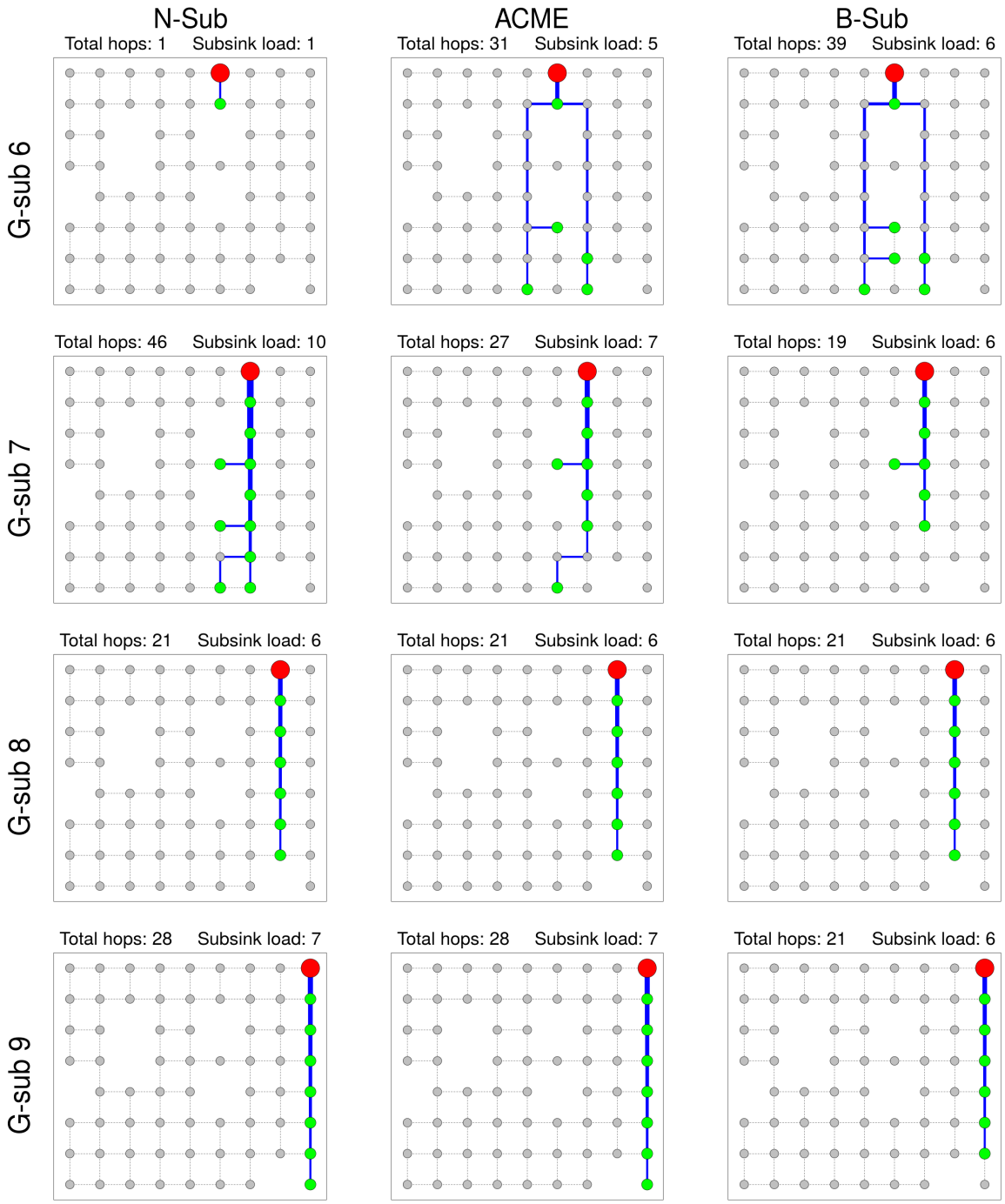


Figure 5.9: ACME's solution for simple example with O_1 and O_2. Both objectives are assigned the same weights (i.e., $w_1 = w_2 = 0.5$). The optimal solutions in terms of total number of hops (provided by N-Sub) and total subsink overload (provided by B-Sub) are displayed as well. **Results for N-Sub:** $(O_1, O_2) = (241, 13)$, $(\Delta_1, \Delta_2) = (0, 13)$. **Results for ACME:** $(O_1, O_2) = (252, 2)$, $(\Delta_1, \Delta_2) = (11, 2)$, $(\lambda_1\Delta_1, \lambda_2\Delta_2) = (2.64, 2)$. **Results for B-Sub:** $(O_1, O_2) = (266, 0)$, $(\Delta_1, \Delta_2) = (25, 0)$.

5.2.2 ACME's robustness and consistency optimizing O_1 and O_2

Now we move forward with the evaluation of ACME's ability to produce satisfactory solutions for a wide variety of network structures. For that purpose, we set up diverse network structures based on combinations of base topology, type of obstacle, connectivity and size of the sensing field, as described in Section 5.1.1. For each resulting subnetwork, we asked ACME to produce a solution five times, each using a different combination of complementary optimization weights $(w_1, w_2) \in \{(1,0), (0.75,0.25), (0.50,0.50), (0.25,0.75), (0,1)\}$. The results of those runs are reported in Table 5.1. Its left panel indicates the combination of levels of the factors constituting each instance (IT); the convention used in the abbreviations is explained in its caption. For example, Instance 2 involves a big sensing field (FS:BG), a big obstacle (OT:BG) located at the right center spot (P:RC), and a subnetwork with square base topology (BT:SQ).

The second and third panel of Table 5.1 from left to right indicate the deficits Δ_1 and Δ_2, resulting from the solution delivered by ACME for each instance and combination of weights. For each instance (a row), the results of one optimization with a given combination of weights (w_1, w_2) are displayed in two cells, one in the column of the second panel with w_1 in the header and the other in the column of the third panel with the complementary weight w_2 in the header. For example, the deficits $\Delta_1 = 11$ and $\Delta_2 = 2$ obtained for the simple example resolved in Section 5.2.1 with $(w_1, w_2) = (0.5,0.5)$ are displayed in the 9-th row of the table, 3rd column of the second and third panel, respectively. In contrast, the deficits $\Delta_1 = 7$ and $\Delta_2 = 6$ for the same instance with $(w_1, w_2) = (0.75,0.25)$ are displayed in the 9-th row of the table as well, but this time in the 2nd column of the second and third panel, respectively. A shade scale is used in the background of the deficit panels to illustrate how, for each instance, the quality of the solution changes for each objective as a result of the variation of the optimization weights.

The fourth panel from left to right reports the scaling factor λ_1 used for each instance. This value was computed at each time based on the procedure explained in Section 4.4.3.1 and later illustrated in the example of Section 5.2.1.1. The value of λ_2 is not shown in the table since we rescaled the factors every time to make it take the value of 1.

The fifth panel of the table reports ACME's performance from a perspective that goes beyond the quality of the solution. The first column of this panel indicates the time required by ACME to reach the stopping conditions in each simulation (i.e., 10 colonies, each with 10 solutions). The second column indicates the maximum, among the values registered for the different combinations of weights, of the average energy consumed by one node during ACME's execution. Finally, the last column of the table expresses this same quantity as a percentage of the initial energy load of the node. The max and min value of each column in this panel are marked with a pointing up and a pointing down triangle, respectively.

IT	Network Configuration				Δ_1					Δ_2					λ_1	Sol time (min)	$\bar{E}$ (Joules)	$\frac{\bar{E}}{E_{AA}}$%
					w_1 = 1 - w_2 (%)					w_2 = 1 - w_1 (%)								
	FS	OT	BT	P/Q/C	100	75	50	25	0	0	25	50	75	100				
1	BG	BG	SQ	P: CH	0	9	15	23	26	15	7	3	1	0	0.23	9.6	0.43	0.002%
2				P: RC	0	17	31	42	54	16	7	3	1	0	0.11	9.7	0.49	0.003%
3				P: BL	0	6	8	17	39	5	3	2	1	0	0.15	9.2	0.39	0.002%
4			TR	P: CH	0	0	3	4	6	6	4	2	2	0	1.00	9.6	0.58	0.003%
5				P: RC	0	10	21	29	32	17	7	3	1	0	0.19	9.8	0.67 ▲	0.004% ▲
6				P: BL	0	0	0	1	4	6	1	0	0	0	1.50	9.3	0.53	0.003%
7		SM	SQ	Q: 4	0	6	7	8	16	8	2	2	1	0	0.38	10.1	0.41	0.002%
8				Q: 5	0	6	10	13	23	13	7	3	0	0	0.26	10,0	0.42	0.002%
9				Q: 6	0	7	11	17	25	13	6	2	0	0	0.24	9.9	0.42	0.002%
10			TR	Q: 4	0	1	4	9	11	12	6	3	1	0	0.55	10.6	0.59	0.003%
11				Q: 5	0	1	6	7	14	12	6	3	2	0	0.43	10.5	0.58	0.003%
12				Q: 6	0	7	14	17	20	17	7	3	1	0	0.30	10.4	0.62	0.003%
13		NO	RD	C: 0.8	0	5	9	13	17	11	4	2	1	0	0.35	8.9	0.45	0.002%
14				C: 0.4	0	16	20	29	34	17	7	3	0	0	0.18	8.2	0.37	0.002%
15			TR	NA	0	0	0	0	0	4	0	0	0	0	1.00	11,0 ▲	0.62	0.003%
16			HX		0	4	5	12	20	7	3	2	0	0	0.30	8.2	0.38	0.002%
17	SM	BG	SQ	P: CH	0	3	4	7	12	6	3	2	0	0	0.50	6.9	0.3	0.002%
18				P: RC	0	5	14	18	27	9	5	3	1	0	0.22	6.7	0.35	0.002%
19				P: BL	0	1	4	13	23	4	3	2	1	0	0.26	6.4	0.3	0.002%
20			TR	P: CH	0	0	0	0	1	5	0	0	0	0	6.00	6.4	0.34	0.002%
21				P: RC	0	1	4	6	10	5	3	2	1	0	0.60	6.2	0.37	0.002%
22				P: BL	0	0	0	1	2	2	0	0	0	0	3.00	6.5	0.36	0.002%
23		SM	SQ	Q: 4	0	1	2	5	8	3	2	2	1	0	0.75	6.9	0.29	0.002%
24				Q: 5	0	1	2	5	10	3	2	1	0	0	0.60	6.8	0.29	0.002%
25				Q: 6	0	3	5	5	11	6	3	2	1	0	0.55	6.7	0.28	0.001%
26			TR	Q: 4	0	1	3	6	7	7	3	2	1	0	0.86	6.6	0.37	0.002%
27				Q: 5	0	0	2	3	5	7	3	2	1	0	1.20	6.5	0.34	0.002%
28				Q: 6	0	1	3	5	7	5	4	2	1	0	0.86	6.4	0.38	0.002%
29		NO	RD	C: 0.8	0	0	1	2	5	4	1	1	1	0	1.20	6.2	0.33	0.002%
30				C: 0.4	0	0	3	8	10	4	3	1	0	0	0.60	5.7 ▼	0.29	0.002%
31			TR	NA	0	0	0	0	0	2	0	0	0	0	1.00	7,0	0.4	0.002%
32			HX		0	2	3	8	11	4	2	1	0	0	0.55	5.8	0.26 ▼	0.001% ▼

Table 5.1: ACME's performance optimizing O_1 and O_2 for diverse subnetwork configurations. From left to right, the 1st panel specifies the characteristics of each instance (IT) based on the following convention: • sensing field size (**FS**): big (BG), small (SM); • obstacle type (**OT**): big (BG), small (SM), no obstacle (NO); • base topology (**BT**): square (SQ), triangular (TR), random (RD), hexagonal (HX); • position of the obstacle (**P**): center high (P: CH), right center (P: RC), bottom left (P: BL); • number of obstacles (**Q**): 4 (Q: 4), 5 (Q: 5), 6 (Q: 6); • connectivity degree (**C**): 0.8 (C: 0.8), 0.4 (C: 0.4). When the subnetwork has no obstacle, the feature P/Q/C does not apply (NA). The 2nd and 3rd panels present the deficits Δ_1 and Δ_2, respectively, for various combinations of complementary weights (w_1, w_2). In the 5th panel, the quantities *Sol time*, $\bar{E}$ and E_{AA} stand for ACME's solution time, the average energy consumed by a node during the optimization, and the initial energy of a node equipped with two AA batteries (18720 J), respectively.

Several things stand out from the results presented in Table 5.1:

1. As shown by the shade scale in the table, ACME's performance in terms of each objective improves as its corresponding weight increases, and deteriorates as it decreases (see Figure 5.10). We never had a case where the performance in one objective deteriorated after increasing its weight. This can be interpreted as evidence of the consistency between the solutions provided by ACME and the priorities specified by the user.

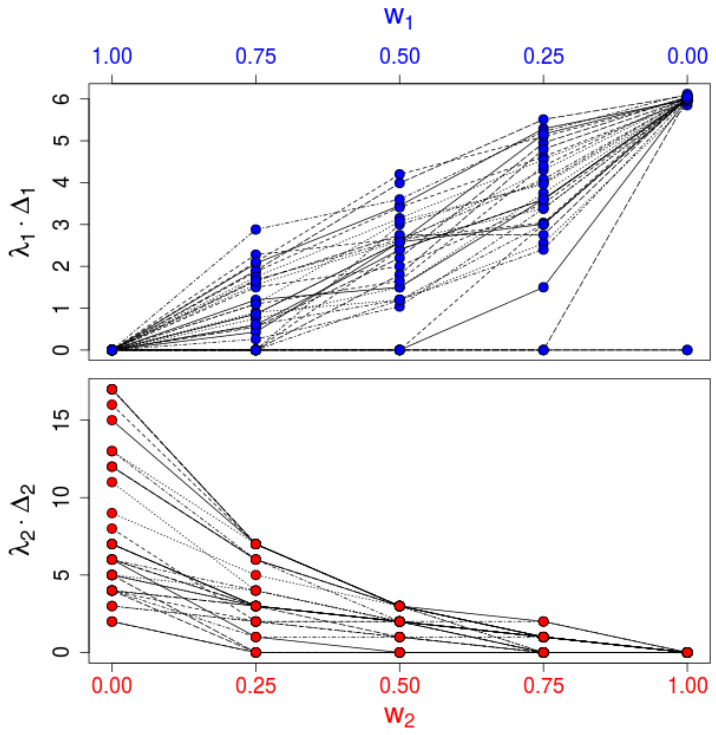


Figure 5.10: Scaled deficit for O_1 and O_2 as a function of the optimization weights (w_1, w_2). The horizontal axes of the two frames are oriented in opposite directions so that pairs of complementary weights are placed at the same horizontal point. This figure provides a summary, in scaled deficits, of the 2nd and 3rd panel of Table 5.1.

2. The consistency between objectives and weights discussed in the previous point was observed persistently over all the instances. This is a first proof of ACME's robustness to changes in the configuration of the subnetwork.

3. Each time we assigned equal weights to both objectives (i.e., $w_1 = w_2 = 0.5$), we obtained fairly close values of scaled deficits $\lambda_1 \cdot \Delta_1$ and $\lambda_2 \cdot \Delta_2$. This can be corroborated by multiplying the columns of $w_1 = 0.5$ and λ_1 from Table 5.1, and then comparing the value obtained in each row with the corresponding value in the column of $w_2 = 0.5$. This shows the ability of the protocol to provide an equilibrium between the two objectives when required by the user through the optimization weights.

4. Table 5.1 shows some instances where the deficits Δ_1 and Δ_2 obtained for a combination $(w_1, w_2) = (0.75, 0.25)$ dominate[2] the ones obtained for a combination $(w_1, w_2) = (1, 0)$. Analogously, in some cases, the deficits Δ_1 and Δ_2 obtained for a combination $(w_1, w_2) = (0.25, 0.75)$ dominate those obtained for a combination $(w_1, w_2) = (0, 1)$. This happens for example in Instance 8, which registers deficits of $(\Delta_1, \Delta_2) = (23, 0)$ for $(w_1, w_2) = (0, 1)$, and deficits of $(\Delta_1, \Delta_2) = (13, 0)$ for $(w_1, w_2) = (0.25, 0.75)$. In that case, the combination $(w_1, w_2) = (0.25, 0.75)$ prevented the algorithm from neglecting objective O_1 and led it to find a solution where Δ_1 was further improved without deteriorating Δ_2. Considering this, a sound strategy when the interest is totally focused on one objective might be to give at least a tiny weight to the other, making it active but clearly secondary. This way, the protocol will give at least some importance to it. In the best case, the protocol will find a dominant solution where the secondary objective is improved at no cost for the main objective; in the worst case, the loss in terms of the main objective will be minor.

5. Throughout the whole experiment, the fitness stayed fairly high. For example, more than 90% of the solutions found by ACME were above 0.9, and the 100% of them were above 0.8 (see Figure 5.11). This result can be interpreted as a second proof of ACME's robustness, validating its ability to provide high quality solutions irrespective not only of the subnetwork configuration but also of the combination of optimization weights.

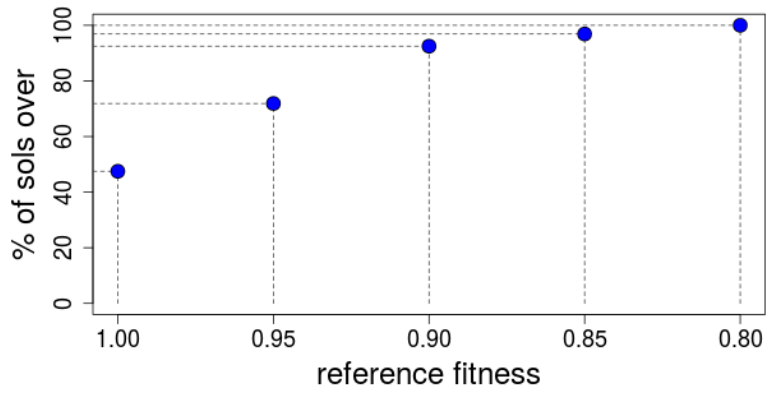


Figure 5.11: Percentage of solutions with fitness above a reference value. The vertical axis indicates the percentage of solutions in Table 5.1 whose fitness was above or equal to each of the reference values specified in the horizontal axis. The percentages were computed over a basis of 160 solutions, corresponding to 32 instances, each with 5 different combinations of weights.

[2]Dominance: solution A is said to dominate solution B if the deficits Δ_1 and Δ_2 obtained for solution A are all lower than or equal to those obtained for solution B, and at least one of the deficits for solution A is strictly lower than the corresponding deficit for solution B. For instance, a solution with $\Delta_1 = 10$ and $\Delta_2 = 8$ dominates a solution with $\Delta_1 = 10$ and $\Delta_2 = 9$.

6. In the best case, ACME took 5.7 minutes to resolve the problem; in the worst, it took 11. Those resolution times are negligible compared to the periods that the network layout is expected to stay unchanged in the ASAS application (e.g., several months). Indeed, even if for some application the subnetwork executed the protocol at the beginning of each labor day, and then it functioned during 8 hours, the worst resolution time of 11 min would correspond to 2.3% of the working time, which is still a rather low percentage. Given that the size of our subnetworks was set realistically for the ASAS application (see Section 5.1), this result supports the suitability of our protocol for use in real life scenarios.

7. In the worst case, ACME reported an average energy consumption per node of 0.67 J, corresponding to the 0.004% of its initial energy load (assuming two AA batteries per node). These numbers are already considerably modest, however, they could still be improved by means of alternative stopping conditions. As shown in Figure 5.12, an important proportion of the best solutions were found before reaching the pre-specified number of colonies (10). Specially in the instances with small sensing field, the best solution was found many times among the first four colonies. In the light of this result, we would expect a lower energy spent in many instances if a stopping condition based on convergence were implemented. This strategy is out of the scope of this thesis, but is proposed as subject of future research.

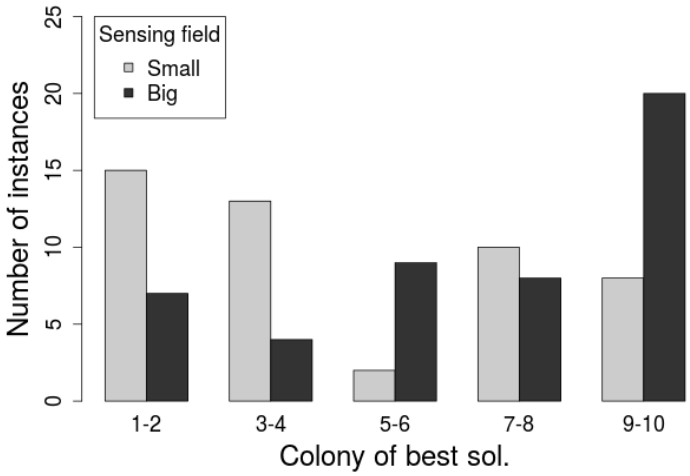


Figure 5.12: Number of times that the best solution was found in the i-th colony, for $i = 1, \ldots, 10$. The horizontal axis is grouped by two for the sake of exposition.

5.3 Adding the total number of retransmissions (O_3) to the optimization

In this section we explore the incorporation of the total number of retransmissions as a complementary objective of optimization. The idea is to check if the energy consumption

of the system can be improved by reducing this quantity. This experiment considers the nine instances listed in Table 5.1 for which the number of retransmissions was the greatest. For each of them we repeat the optimization using different triples (w_1, w_2, w_3). Note that in this case, the ant packets must have the size that the data packets would have. Thus is necessary for the system to be able to replicate during the optimization the exact interactions that the data packets would have during the operation for each candidate routing solution. Before examining the results of the new experiment, let us briefly show an example of an optimization when O_3 is made active in the optimization.

5.3.1 Simple example optimizing O_1, O_2 and O_3

This time we use a triangular base topology with four small randomly located obstacles, covering a big sensing field. Such a subnetwork is shown in Figure 5.13.

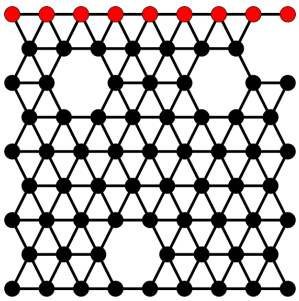


Figure 5.13: Subnetwork to show ACME's performance when optimizing O_1, O_2 and O_3.

5.3.1.1 Setting the scaling factors λ_1, λ_2 and λ_3

Both objective O_1 and objective O_3 are oriented to minimize the energy consumption of the subnetwork. Objective O_1 addresses the issue from the perspective of the number of hops (or packet transmissions), while objective O_3 does it from the perspective of the number of retransmissions. Since both transmissions and retransmissions are equally costly in terms of energy, we use the same scaling factor for the deficits of both objectives. More precisely, we first set λ_1 and λ_2 as described in Section 5.2.1.1 and then we set $\lambda_3 = \lambda_1$.

5.3.1.2 Evolution of the optimization

Similarly as we did for the optimization with only O_1 and O_2, we start by analyzing the evolution of the fitness of the solutions found by ACME along the colonies. This information is displayed in Figure 5.14. The improvement of the fitness as the colonies pass verifies that the learning mechanism of the protocol is working properly also with O_3 active. The final fitness value achieved was 0.925, which was obtained in about 10 minutes.

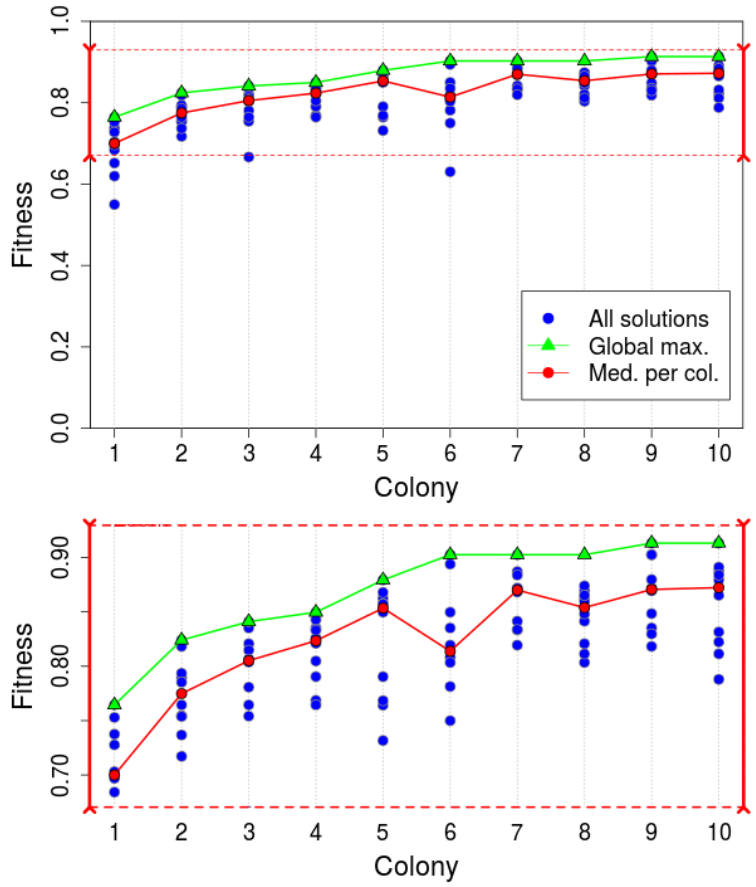


Figure 5.14: Evolution of fitness when optimizing O_1, O_2 and O_3.

As complementary information, the scaled deficit for each objective in each solution along the colonies is reported in Figure 5.15. This figure shows how the three objectives compete against the others until reaching a midpoint where all of them have similar scaled deficits. During this optimization, the protocol accepted some extra units of Δ_2 in order to reduce both, Δ_1 and Δ_3.

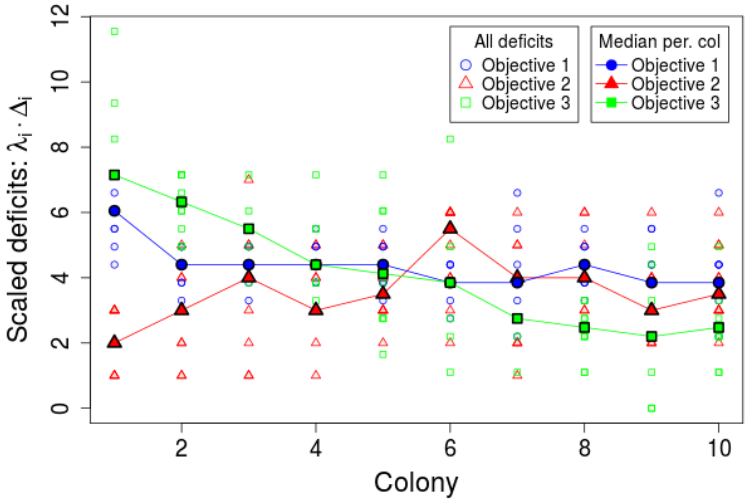


Figure 5.15: Evolution of scaled deficits when optimizing O_1, O_2 and O_3.

5.3.1.3 Visualizing a solution provided by ACME

The solution found by ACME for this example is illustrated in Figure 5.16. The left frame of the figure shows the retransmissions experienced when only O_1 and O_2 were optimized. The right frame shows the retransmissions achieved by adding O_3 to the optimization. The number of retransmissions was significantly improved in this case. Moreover, the decrease in Δ_3 was significantly larger than the increment in Δ_1, which indicates that the *total number of transmissions* (regular plus retransmissions) was effectively reduced.

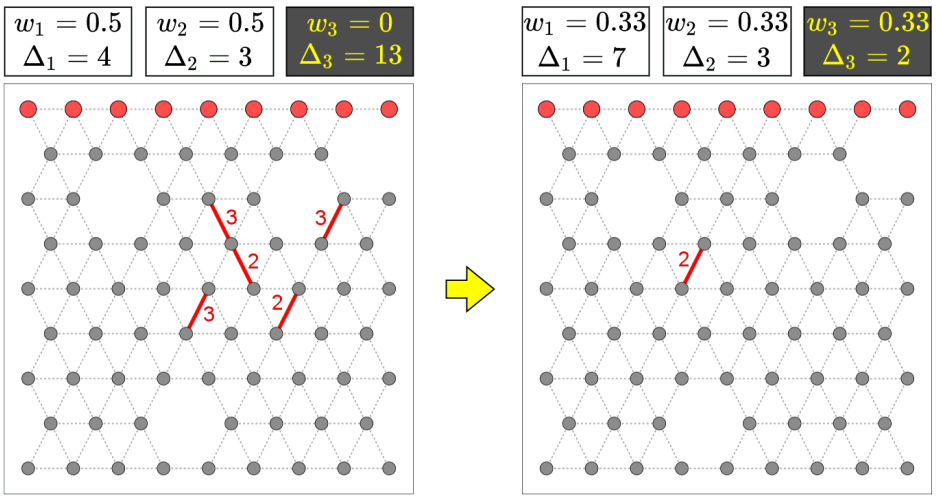


Figure 5.16: Comparison between ACME's solution for simple example with and without activating O_3 in the optimization. The solution with O_3 active is the one shown in the right panel. The deficits of the tree objectives are also reported in the figure to illustrate the potential degradation of O_1 and O_2 that might take place in order to improve O_3. The red links in the figure represent the ones where the retransmissions occurred. The number by the side of the affected links indicates the number of retransmissions experienced.

5.3.2 Updated results after adding O_3

In this section, we test ACME's ability to reduce the number of retransmissions in the instances the most affected by this issue among the ones evaluated in Section 5.2.2. More precisely, we consider Instances 4, 5, 6, 10, 11, 12, 13, 14 and 15, from Table 5.1. For each of those instances, we made ACME resolve the problem using different combinations of complementary optimization weights, this time letting w_3 take values different from zero. In the experiment presented in this section, we fix the values of w_1, w_2 and w_3 by maintaining the proportion between w_1 and w_2 used in Table 5.1, and setting $w_1 = w_3$. This last decision obeys to the reasoning that objectives O_1 and O_3, both seek to reduce the total number of transmissions and are equivalent in terms of impact over energy consumption. For instance, instead of the duple $(w_1, w_2) = (0.75, 0.25)$, now we use the triple $(w_1, w_2, w_3) = (0.43, 0.14, 0.43)$[3].

[3]We could say that the new weights w_1, w_2 and w_3 are set based on the following straightforward system of equations: Eq.1: $w_1 + w_2 + w_3 = 1$; Eq.2: $(w_1 = w_3)$; Eq.3: $(w_1/w_2) = (w_1^o/w_2^o)$, with w_1^o and w_2^o denoting the optimization weights used in 5.2.2 for O_1 and O_2.

The results of the new optimizations are reported in Table 5.2. Part of this table follows a structure similar to that of Table 5.1. The left panel indicates the number of the instance (IT), the cells at the top indicate the weights used for each optimization and the columns Δ_1, Δ_2 and Δ_3 report, on each row, the deficits obtained in one instance. The columns I_3 and ES were added to allow comparison against the results obtained without using O_3 in the optimization. I_3 indicates the number of retransmissions that ACME was able to reduce in each instance with respect to the optimization only considering O_1 and O_2. ES, on the other hand, specifies the amount of energy that would be saved by the network per round of data sensing and transfer, thanks to the addition of O_3 in the optimization. Negative values imply that there was no energy saving but rather an increment in energy consumption. The time for resolution and energy consumed by the subnetwork in the optimization remained almost the same as reported in Table 5.1.

IT	$w_1 = 50\%$ $w_2 = 0\%$ $w_3 = 50\%$					$w_1 = 43\%$ $w_2 = 14\%$ $w_3 = 43\%$					$w_1 = 33\%$ $w_2 = 33\%$ $w_3 = 33\%$					$w_1 = 20\%$ $w_2 = 60\%$ $w_3 = 20\%$					$w_1 = 0\%$ $w_2 = 100\%$ $w_3 = 0\%$				
	Δ_1	Δ_2	Δ_3	I_3	ES	Δ_1	Δ_2	Δ_3	I_3	ES	Δ_1	Δ_2	Δ_3	I_3	ES	Δ_1	Δ_2	Δ_3	I_3	ES	Δ_1	Δ_2	Δ_3	I_3	ES
4	3	8	4	6	0.001	2	6	3	5	-0.001	4	3	3	7	0.001	4	2	7	1	-0.002	6	0	8	0	0
5	0	13	0	17	0.011	6	9	9	17	0.003	23	3	13	15	0.002	29	1	15	12	0.009	32	0	27	0	0
6	0	6	0	8	0.007	0	1	0	6	0.001	0	0	0	6	0.006	1	0	0	3	-0.001	4	0	3	0	0
10	0	14	0	5	0.008	1	7	2	11	0.006	7	3	2	11	-0.003	8	2	5	8	0.004	11	0	13	0	0
11	0	10	0	14	0.001	1	6	4	0	0.001	7	3	4	3	-0.002	7	2	7	0	0.000	14	0	7	0	0
12	2	18	0	10	0.001	6	8	6	13	-0.002	14	3	7	6	-0.005	16	2	6	5	0.002	20	0	11	0	0
13	2	10	2	8	0.005	4	5	3	3	0.002	10	2	6	2	0.001	13	1	3	6	0.001	17	0	17	0	0
14	0	16	0	7	0.003	18	6	2	5	-0.003	19	4	2	5	0.003	30	0	0	7	0.001	34	0	7	0	0
15	0	9	8	28	0.016	0	1	5	13	0.002	0	0	17	4	0.004	0	0	12	4	0.001	0	0	18	0	0

Table 5.2: ACME's performance optimizing O_1, O_2 and O_3 for the subnetwork configurations of Table 5.1 with greatest number of retransmissions. Each panel displays the deficit Δ_i for each of the objectives, the number of retransmissions removed (I_3), and the energy savings in Jules (ES) resulting from the new value of w_3.

We have the following comments on the results:

1. In all the instances, and for all the combinations of weights, ACME was able to reduce the number of retransmissions with respect to the optimization without objective O_3. This proves the ability of the protocol to optimize this third objective, if required.

2. Objective O_3 shows a marked conflict with objective O_2. Except for a few cases, an improvement in Δ_3 implied a deterioration in Δ_2 and vice versa.

3. Although the optimization always allowed to reduce the number of retransmissions with respect to the results of Table 5.1, the energy savings were negligible and in some cases there was even an energy overspent (the negative values of ES). This last aspect was

out of our expectations before observing the results. Indeed, we expected to be able to reduce the energy consumption at any instance where we achieved a $\Delta_1 + \Delta_3$ lower than in the optimization without O_3. A deeper analysis of the solutions revealed that the cause of the negative ES values is mostly the overhearing[4]. When only O_1 and O_2 are optimized, the ants circumvent an obstacle using routes which pass right by its side. With only those two objectives active, there is no reason for deviating the route further away. In contrast, when O_3 is made active, the ants might move off the border of the obstacle to try to prevent collisions. The routes used in the optimizations involving O_3 might thus unintentionally cause greater overhearing, ending up in larger overall energy consumption. This result points out that the incorporation of the overhearing in the optimization might be a valuable contribution. We leave this possibility as an open research topic for future research.

4. Another reason for our small energy savings is that our number of nodes is not large enough to have very strong congestion issues to be corrected. Indeed, we have chosen a relatively large communication range (100 m), and thus a subnetwork less densely filled than in many studies in the literature relying on the IEEE 802.15.4 standard which go down to a communication range of 20 m (e.g., in [192]). For our application, the communication range and number of nodes selected seem reasonable, however, with the resulting number of nodes and the amount of traffic moved there is not a strong reason to put a nonzero weight to objective O_3. This is especially true given that O_3 is in conflict with O_2 which is crucial for the efficient communication with the MULE. Thus, for the experiment in the next section which involves interactions between the subnetwork and the MULE, we set w_3 back to zero, and we return to the optimization relying strictly on our two main objectives, O_1 and O_2. Based on the observed improvements of Δ_3, we still believe that O_3 has some potential of improving the energy consumed in situations involving greater traffic. Complementary experiments could be devoted in the future to the analysis of ACME performance in that kind of scenario.

5.4 ACME vs N-Sub: once the solution is implemented

Sections 5.2 and 5.3 focused on the evaluation of ACME's robustness to variations in the configuration of the subnetwork. Now we incorporate the MULE in the analysis. Throughout the thesis, our main goal has been to find a routing strategy able to efficiently transfer collected data from the subnetwork to the MULE. In this section, we tests ACME's performance in this dimension by comparing it with N-Sub in terms of the ability to exploit moments of contact with the MULE. Our tests are based on the irregular subnetworks used

[4]Overhearing: when a node transmits a packet, all the nodes in its communication range receive this packet, even if they are not the intended destination [191]. Thus, energy is wasted when a node receives packets that are destined to other nodes.

in the experiments presented in Sections 5.2 and 5.3. We expect ACME to provide subsink balance in some instances where N-Sub will not be able to. As a consequence, we foresee some untenable solutions of N-Sub, where some subsinks would not be able to transfer all the stored information to the MULE. The experiment is explained in detail below.

5.4.1 The experiment in brief

This last experiment is conducted in two stages. The ***first stage*** focuses on the time that the subnetwork is able to tolerate between two consecutive visits of the MULE. Since the system proposed in this thesis relies on the opportunistic contact with the MULE, an essential requirement is that the subnetwork tolerates considerably long periods without being visited by it. In general, the longer the period between two consecutive visits of the MULE (hereon called ***MULE's timeout*** or simply ***timeout***), the more data each subsink accumulates and the more prone it becomes to retain part of the data after the visit of the MULE. Thus, for diverse subnetwork configurations, we check the maximum timeout that the subnetwork can handle without reaching an untenable state of uncontrolled data accumulation at the subsinks. More precisely, we estimate the maximum tolerable timeout when operating based on ACME, and we compare it with the analogous time when implementing N-Sub. The analysis is performed considering the 32 irregular subnetwork configurations used for the experiments in Sections 5.2 and 5.3. We use N-Sub as a benchmark for ACME in this experiment, given the popularity of its underlying philosophy of routing to the closest candidate destination in multi-sink WSN settings [105, 193] (see e.g., [194], [195], [196], for examples of studies where this approach has been followed).

The second stage of the experiment evaluates ACME's performance in terms of energy consumption. In this regard, we check the energy consumed by the subnetwork during the optimization, data collection and data transfer periods. As before, we compare ACME's results with those of N-Sub. Our main purpose for this stage is to verify the feasibility of ACME for application in real contexts based on its energy demand.

5.4.2 Complements to the experiment setup

The experiment setup presented at the beginning of this chapter, in Section 5.1, was enough for the analysis presented in Sections 5.2 and 5.3. Those were focused on the optimization phase, where each node is assigned a G-subsink and the routing tables are built. Our new experiment expands the frame to the operational phase, where data collection and contact with the MULE take place. Thus, some additional parameters are needed to specify how these two processes are carried out. Those complementary parameters are described below.

MULE's functioning

Architecture

The architecture implemented in the MULE is displayed in Figure 5.17. It adopts the same configuration than the sensor nodes at the MAC and PHY layers (e.g., bitrate of 250 kbps and communication range of 100 m), specified in Section 5.1. At the Network layer it only implements IPv6 as data plane protocol since the control plane protocol (N-Sub or ACME) is not required. The MULE implements a particular Application layer protocol controlling: (i) the periodic broadcast of beacons to alert the subsinks about its presence, and (ii) the storage of the data received from the subsinks. In addition, the transfer of acknowledgement packets to the subsinks following the reception of data packets is activated on the MAC layer to ensure their successful reception. As soon as a subsink receives a beacon it starts sending its stored data until it identifies that the contact with the MULE is lost (i.e., until it stops receiving beacons).

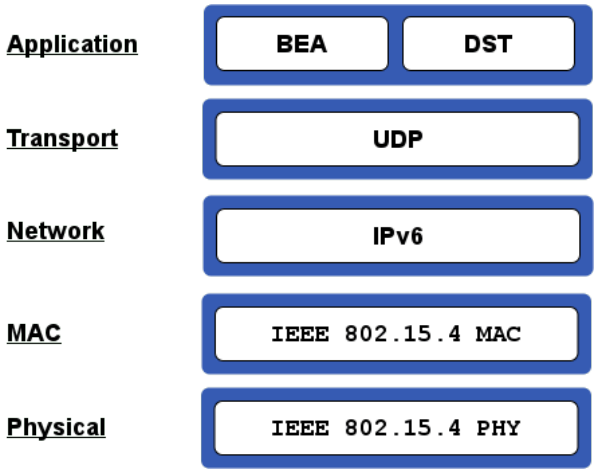


Figure 5.17: Protocol stack for the MULE. BEA and DST stand for the modules responsible for the sending of beacons and data storage, respectively.

Further parameters

In this experiment, the role of MULE is played by the ground service vehicles belonging to the airport where the subnetwork is operating. For the sake of interpretability of the results, we consider only one MULE visiting the subnetwork at a time. Nonetheless, we remark that such an assumption is by no means necessary for the implementation of ACME and can be removed without implications on the functioning of the subnetwork. In this test, we seek more rigor in the characterization of the MULE. We start by setting three different values of MULE speed which properly describe the behavior of the ground service vehicles. The MULE speeds we use are 30, 40 and 50 $\mathrm{km\,h^{-1}}$, corresponding to the speed limits for service vehicles when moving along different segments of the service roads such as the *rear of aircraft roads* and the *perimeter roads* [197, 198, 199, 200]. We assume that the speed

of the MULE is constant during the time in contact with the subnetwork. In addition, we consider that the subsinks are deployed along a straight line, parallel to one section of a service road, as depicted in Figure 5.5. The only exception to this rule are the subnetworks based on the random topology, where the subsinks are still placed by the side of the service road, but due to the nature of this type of structure they are not perfectly aligned. We fix the perpendicular distance from any subsink to the MULE's path at 80 m. Based on several measurements performed by us with the distance calculator tool available in Google Maps, this value describes well the distance that might separate the subsinks from the service roads in a potential deployment of the system. Our measurements included airports of different sizes such as Toulouse Blagnac, Montpelier Airport, Charles de Gaulle and Miami Intl. For the random base topology, 80 m is the *maximum* distance separating a subsink from the MULE. The time in contact with the MULE for each subsink can be easily computed based on the speed of the MULE, the communication range of the subsinks and the perpendicular distance separating them from the trajectory of the MULE. Based on the MULE speed values specified above, for any subnetwork with a non-random base topology, the MULE stays in contact with each subsink along 120 m, which leads to times in contact per subsink of 14.40, 10.80 and 8.64 s, respectively. Two subsinks are allowed to maintain contact with the MULE simultaneously.

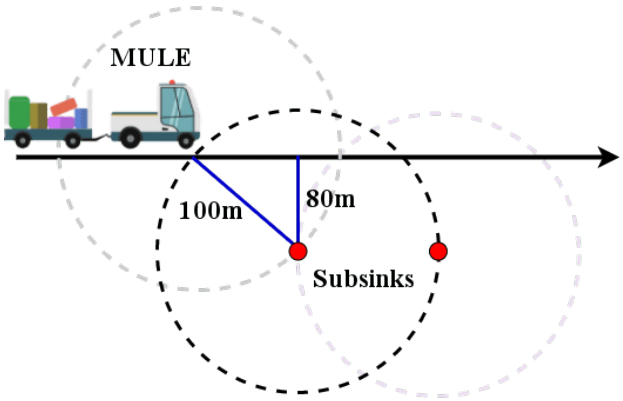


Figure 5.18: Location of the subsinks relative to the trajectory of the MULE. The circumferences represent the communication range of the devices. The numbers indicate the length of the communication range and the shortest perpendicular distance from a subsink to the trajectory of the MULE.

Along its path, the MULE sends beacons at a constant rate of 1 message each 0.2 seconds. We fix the time between sendings of data packets from the subsink to the MULE at 0.02 s. We assume that the storage capacity of the MULE is large enough for it to be able to recover all the data collected by the subnetwork during several days. Although in the ASAS application we do not expect the subnetwork to ever spend that much time without having contact with a MULE, this reasonable assumption is made to discard memory issues during the transfer of data to the MULE. In addition, this assumption, along with the fact that the data to collect

is tolerant to delays, removes any need for data transfer between the subnetworks. Finally, we assume that the device installed in the MULE is powered by the energy system of the vehicle, which avoids problems of energy.

Data collection

In this experiment, we consider a periodic data collection scheme where each node makes one measurement and sends data each 60 s. This periodicity is based on the nominal sensing rate used in automatic weather stations at the airports [35]. This value is in turn consistent with the range of sensing rates typically used for environmental monitoring in various applications (one measurement every 10 seconds to some minutes [112]). As we have said along the document, we assume that the messages related to critical events such as the presence of a foreign object in a runway are sent directly to the sink using long-range communication technology. Thus, we do not include this type of data in the analysis. We recall that these types of events occur with relatively low frequency and thus, the implications on the energy consumption at the nodes are negligible.

As in the experiments of Chapter 3, we consider data packets of 85 B containing the environmental and operational data collected during each interval of 60 s. The 85 B (payload of 16 B) constitutes a packet large enough to store five 2-digit variables along with the information added by the different layers of the protocol stack.

Performance indicators

• ***Maximum tolerable MULE's timeout*** (T_{out})**:** time between two consecutive visits of the MULE above which the most loaded subsink starts to accumulate packets that it will not be able to transfer to the MULE during the time in contact. This time can be computed based on the maximum number of packets that can be sent to the MULE during the time in contact with a subsink (Q), and the number of packets that the most loaded subsink will receive per unit of time (η):

$$T_{\text{out}} = \frac{Q}{\eta} \tag{5.1}$$

The value of Q in Eq. (5.1) can be obtained either theoretically (Q_{theo}) or based on simulation (Q_{simu}). Theoretically, it is computed as the time spent by the MULE in contact with one subsink, divided by the time between sendings of data packets from the subsink to the MULE. For the MULE speeds of 30, 40 and 50 km h^{-1} considered in our experiments, Q_{theo} gives 720, 540 and 432 packets. Both, in reality and simulation, those values are likely to be slightly different due to at least three reasons. Firstly, the first beacon that reaches a subsink might not be received exactly at the time when the MULE and the subsink enter

in the communication range of each other, but just a little later. Secondly, a subsink might receive information from other nodes in the subnetwork while a beacon is trying to reach it (hidden node problem), causing the loss of the beacon and therefore, reducing the amount of time in contact with the MULE exploited in data transfer. Thirdly, a subsink does not notice the loss of contact with the MULE at the exact moment it happens; thus, in some cases, the subsink keeps attempting to transfer data to the MULE during a short amount of time after the window of effective time in contact has ended. Seeking to incorporate the aforementioned situations into account, our results are based on a set of simulations intended to produce the Q_{simu} values. The details are provided in the corresponding section.

The value of η in Eq. (5.1) can easily be computed based on the time between data collection rounds and the number of nodes assigned to the most loaded subsink during the optimization. In our case, the time between collection rounds is 1 minute (see Section 5.4.2); thus, the value of η is just the number of nodes assigned to the most loaded subsink by either ACME or N-Sub, as appropriate.

• ***Energy Consumption (EC):*** amount of energy consumed by the subnetwork or by some specific nodes during the periods of: (i) optimization, (ii) data collection and transfer to the subsinks, and (iii) transfer to the MULE. We will specify in each case which nodes and period we are referring to during the analysis.

5.4.3 Maximum tolerable MULE's timeout

The most loaded node of one subnetwork represents a weak point of it. This subsink is the most susceptible one to accumulate data packets if the MULEs do not visit the subnetwork very often. If one subsink keeps consistently accumulating data packets, eventually it will have no option but dropping them. This would not only imply holes in the dataset received by the sink, but also energy wasted by the sensor nodes in the collection and transfer of data that will never be used. Hence, a desirable operating state for the subnetwork is one where the maximum tolerable MULE's timeout T_{out} is as large as possible.

As mentioned in the experiment setup of this chapter, in this experiment we compute T_{out} based on Q_{simu}. To obtain the value of Q_{simu} for each MULE speed, we performed one simulation where all the nodes were largely filled 10 times, and each time the MULE was allowed to pass as recover as much data as it could. Then, we averaged the number of packets effectively transferred by a node for each MULE speed, obtaining Q_{simu} values of 718, 537 and 429 packets, for the speeds of 30, 40 and 50 km h^{-1}, respectively. Based on these quantities and the η values resulting from the optimizations performed by ACME for $(w_1, w_2) = (0.5, 0.5)$, and also by N-Sub, we computed the T_{out} for each of the 32 subnetwork considered in $\mathbf{Part_1}$ of the experiment. The results are displayed in Table 5.3. This time we sorted the instances based on decreasing ordering of the *margin* column, which indicates

the difference between the T_{out} obtained by ACME and N-Sub in each case. The load of the heaviest subsink for each instance and protocol appears in the left panel of the table. We recall that, in our case, this value is the same as η since we fixed a time between rounds of data collection of 1 minute and T_{out} is given in minutes.

IT	Load of heaviest subsink		Maximum tolerable time between visits of the MULE, T_{out} (min)								
			MULE's speed: 50 kmph			MULE's speed: 40 kmph			MULE's speed: 30 kmph		
	N-Sub	ACME	N-Sub	ACME	Margin	N-Sub	ACME	Margin	N-Sub	ACME	Margin
2	20	8	21,5	53,6	32,2	26,9	67,1	40,3	35,9	89,8	53,9
18	12	7	35,8	61,3	25,5	44,8	76,7	32,0	59,8	102,6	42,7
20	7	5	61,3	85,8	24,5	76,7	107,4	30,7	102,6	143,6	41,0
22	7	5	61,3	85,8	24,5	76,7	107,4	30,7	102,6	143,6	41,0
1	13	8	33,0	53,6	20,6	41,3	67,1	25,8	55,2	89,8	34,5
5	18	10	23,8	42,9	19,1	29,8	53,7	23,9	39,9	71,8	31,9
11	15	9	28,6	47,7	19,1	35,8	59,7	23,9	47,9	79,8	31,9
13	15	9	28,6	47,7	19,1	35,8	59,7	23,9	47,9	79,8	31,9
6	10	7	42,9	61,3	18,4	53,7	76,7	23,0	71,8	102,6	30,8
9	12	8	35,8	53,6	17,9	44,8	67,1	22,4	59,8	89,8	29,9
19	8	6	53,6	71,5	17,9	67,1	89,5	22,4	89,8	119,7	29,9
21	8	6	53,6	71,5	17,9	67,1	89,5	22,4	89,8	119,7	29,9
27	8	6	53,6	71,5	17,9	67,1	89,5	22,4	89,8	119,7	29,9
32	8	6	53,6	71,5	17,9	67,1	89,5	22,4	89,8	119,7	29,9
12	17	10	25,2	42,9	17,7	31,6	53,7	22,1	42,2	71,8	29,6
8	14	9	30,6	47,7	17,0	38,4	59,7	21,3	51,3	79,8	28,5
10	13	9	33,0	47,7	14,7	41,3	59,7	18,4	55,2	79,8	24,5
16	13	9	33,0	47,7	14,7	41,3	59,7	18,4	55,2	79,8	24,5
30	11	8	39,0	53,6	14,6	48,8	67,1	18,3	65,3	89,8	24,5
14	16	11	26,8	39,0	12,2	33,6	48,8	15,3	44,9	65,3	20,4
4	10	8	42,9	53,6	10,7	53,7	67,1	13,4	71,8	89,8	18,0
24	7	6	61,3	71,5	10,2	76,7	89,5	12,8	102,6	119,7	17,1
26	7	6	61,3	71,5	10,2	76,7	89,5	12,8	102,6	119,7	17,1
28	7	6	61,3	71,5	10,2	76,7	89,5	12,8	102,6	119,7	17,1
31	7	6	61,3	71,5	10,2	76,7	89,5	12,8	102,6	119,7	17,1
7	11	9	39,0	47,7	8,7	48,8	59,7	10,8	65,3	79,8	14,5
15	11	9	39,0	47,7	8,7	48,8	59,7	10,8	65,3	79,8	14,5
29	11	9	39,0	47,7	8,7	48,8	59,7	10,8	65,3	79,8	14,5
17	8	7	53,6	61,3	7,7	67,1	76,7	9,6	89,8	102,6	12,8
23	8	7	53,6	61,3	7,7	67,1	76,7	9,6	89,8	102,6	12,8
25	8	7	53,6	61,3	7,7	67,1	76,7	9,6	89,8	102,6	12,8
3	9	8	47,7	53,6	6,0	59,7	67,1	7,5	79,8	89,8	10,0

Table 5.3: Maximum tolerable MULE's timeout, T_{out}, reached by ACME and N-Sub for 32 irregular subnetwork configurations and 3 typical speeds of ground service vehicles.

The remarks below follow from the analysis of Table 5.3.

1. Generally speaking, the values of T_{out} presented in the table seem reasonable for application in the ASAS context. Nonetheless, we acknowledge that those could significantly change depending on several factors. Primarily, we could mention the airport layout (determining the topology of the subnetworks), the number of nodes per subnetwork, the data collection rate, the number of service vehicles performing as MULEs, the placement of the subnetworks in the airport (i.e., if they are placed in zones frequently or rarely visited by the MULEs), among other factors. Thus, we restrain ourselves from making further judgments about the values of the times in themselves, and we concentrate on the comparison of the results achieved by the two protocols.

2. In all the instances, ACME reached a larger T_{out} than N-Sub. This is illustrated in Figure 5.19, which presents the values registered by both protocols for a speed of $50\,\text{km}\,\text{h}^{-1}$. The dominance of ACME comes from its ability to provide subsink balance even for irregular subnetwork configurations. Since the structure of the subnetwork will rarely be the same for different airports, and even for different spots of the same airport, ACME constitutes a more versatile approach, able to retain greater flexibility regarding the behavior of the MULEs regardless of the subnetwork structure.

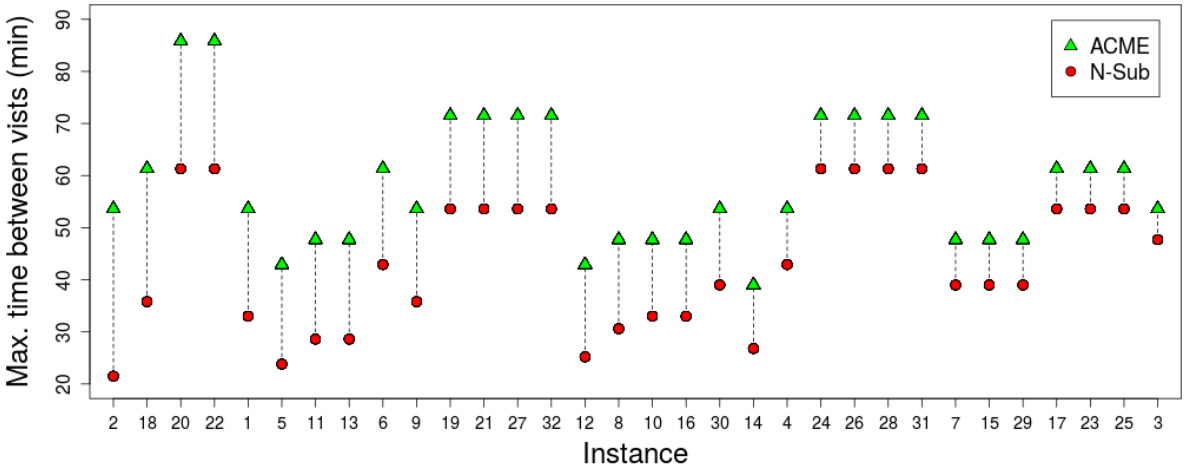


Figure 5.19: Maximum timeout for each protocol, at each instance.

3. For some subnetwork configurations, the T_{out} obtained by both protocols was very similar. This happened in instances where the routing solution strictly focused on path length (O_1) resulted in a reasonable subsink balance (O_2). However, after deployment, a subnetwork structure is very likely to change in the mid term due to battery depletion of the nodes, and in the long term due to the extension or relocation of the subnetwork. Those changes might turn a relatively easy subsink balance instance into a hard one. Thus, the safer approach may be to rely on ACME from the beginning, knowing that it will be able to adapt to all situations when required.

5.4.4 Energy management

Now we compare ACME and N-Sub in terms of energy consumption (EC). We consider two different facets of this performance indicator: the EC at the subsinks (EC_s) and the overall EC of the subnetwork (EC_o). Note that our goal is by no means to outperform N-Sub in terms of total energy consumed. We anticipate notably good results for N-Sub in this dimension thanks to the use of shorter paths than ACME, which in turn might cause less overhearing. Our aim is to check how the balance of subsink load offered by ACME contributes to a more homogeneous usage of the energy of the subsinks than in N-Sub. In addition, we seek to verify the suitability of ACME for application in real situations, based on its energy demand.

Energy consumption at the subsinks

Figures 5.20 and 5.21 display, for Instances 2 and 6, and for each protocol: (i) the number of nodes assigned to each subsink; (ii) the average energy consumed by each subsink during one round of data collection; and (iii) the average energy consumed by each subsink during one period of data transfer to the MULE. Averages are based on a simulation where the cycles of data collection and transfer to the MULE were repeated 10 times. For those simulations we fixed a MULE speed of 50 km h^{-1} and a timeout equal to the maximum tolerable timeout when using N-Sub, taken from Table 5.3 (21.5 min for Instance 2 and 42.9 min for Instance 6). We use the maximum tolerable timeout of N-Sub since this is the largest timeout value at which the comparison of the two protocols has sense. For values between the maximum tolerable timeout of the two protocols, the functioning of the subnetwork based on the solution of N-Sub is inefficient and inadvisable given that the subnetwork would be spending energy in collection and transfer of data that would never reach the sink.

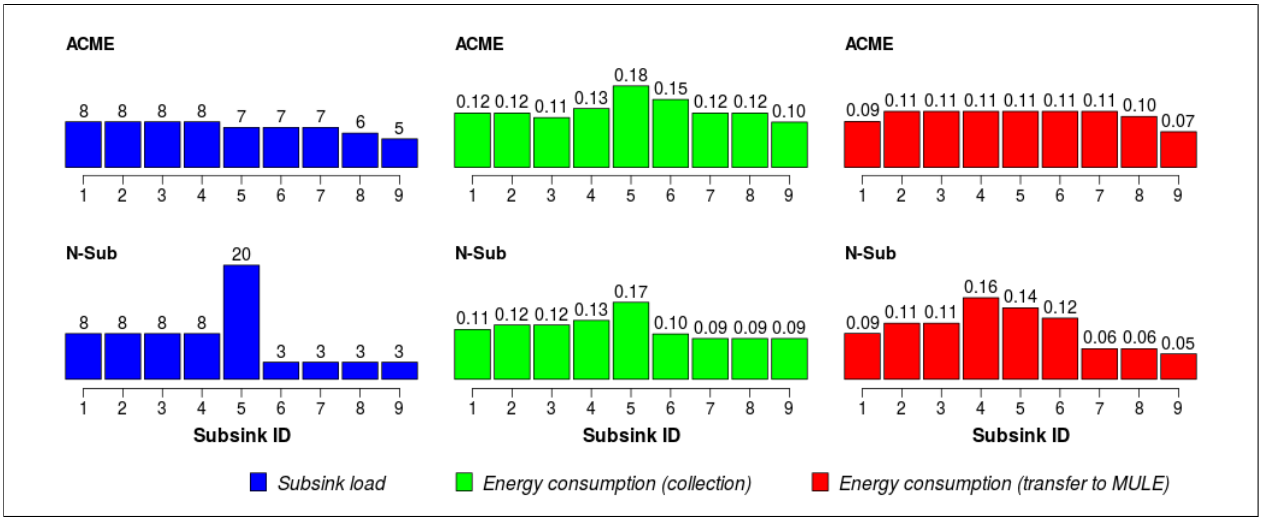


Figure 5.20: Average load and energy consumption per subsink for Instance 2.

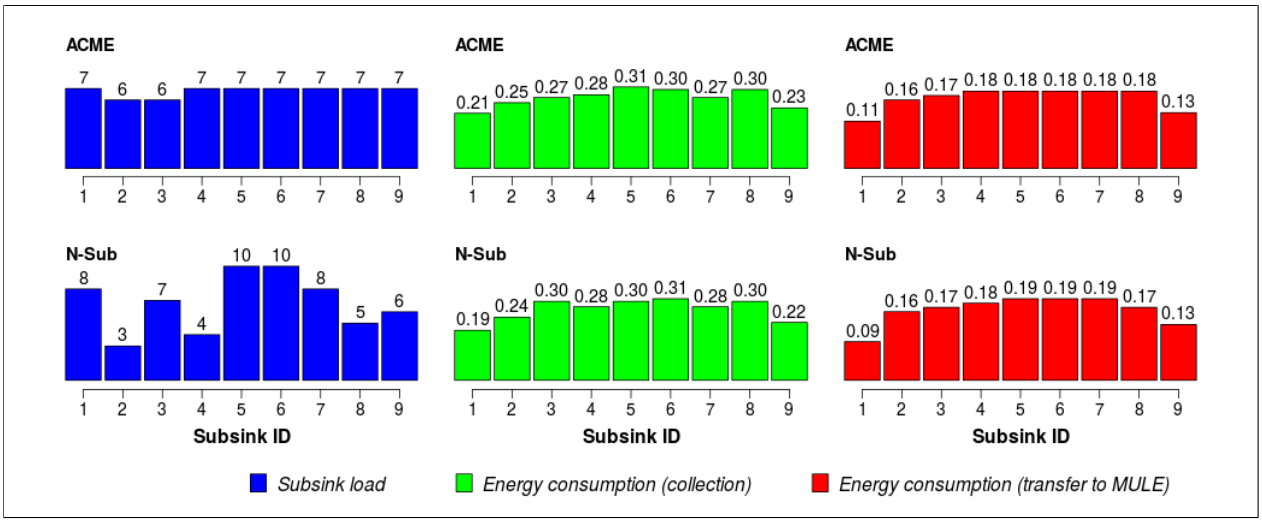


Figure 5.21: Average load and energy consumption per subsink for Instance 6.

Among all the 32 instances reported in Table 5.3, we show the results for Instances 2 and 6 which represent well the behavior observed in the whole group. Our remarks following the observation of the results are summarized below:

1. ACME was always able to provide a fairly good subsink balance, which translated almost directly into energy balance during the period of data transfer to the MULE. The consumption during data collection was not that balanced, even though, it was not so bad either. After inspection of the solutions, we concluded that such an heterogeneity mainly has two sources. The first one is that some subsinks are used as intermediate hops to reach other subsinks, thus experiencing higher energy consumption. The second one is overhearing during the period of data transfer to the MULE. During this period, the nodes at the two extremes of the subnetwork experience overhearing from only one node, while all the others experience overhearing from two nodes. This is partly the cause of the slightly lower energy consumption during data transfer to the MULE, visible in subsinks 1 and 9. This second result where the results of energy consumption were significantly affected by overhearing stresses the relevance of studying the way to incorporate this phenomenon in the optimization in future research.

2. The subsink balance offered by N-Sub was relatively far from ideal. In some scenarios like Instance 6, we observe that the energy consumption of both methods is fairly similar despite the subsink imbalance. This, once again is an effect of overhearing, which makes the less loaded subsinks spend more energy than one would expect based on its assigned load. The effect of overhearing is not so evident in other scenarios like Instance 2, where most of the load is placed over a single subsink. In that case, most of the energy consumption in reception of data packets and transfer to the MULE happens at that overloaded subsink, and most of the overhearing happens at its neighbors. Thus, the energy consumption of that group of nodes stays above that of the other subsinks.

3. We must admit that the results are not exactly what we were expecting. With the balance of the subsink load, we aimed to balance the energy consumption at the subsinks as well, which we only achieved partially (during the transfer of data to the MULE). Future research might be directed to the study of ways to incorporate in the optimization (at least) the two factors listed above that influenced the heterogeneous energy consumption with ACME.

Overall energy consumption of the subnetwork

Now we make some energy consumption estimations oriented to show the suitability of ACME for application in real contexts. The computations are based on the parameters and results for Instance 2. Based on our simulations, ACME consumed an average of 30.67 J during the optimization of routes, and 12.13 J during a round of data collection and transfer to the MULE. The corresponding values for N-Sub were 4.43 J and 11.56 J, respectively. For the sake of exposition, let us assume that the routes are re-optimized each day before the data collection starts[5]. Let us further assume that the energy consumption is a linear function of time. Then, considering the timeout of 21.5 min obtained by simulation, the total energy spent by the subnetwork operating with ACME at the end of a working day of 16 h would be

$$30.67\text{J} + \left\lfloor \frac{16\text{h} * 60\text{m/h}}{21.5\text{m}} \right\rfloor * 12.13\text{J} = 564.39\text{J},$$

or 8.82 J in average, per node, considering the total of 64 nodes in the subnetwork used in Instance 2. The analogous computation for N-Sub gives

$$4.43\text{J} + \left\lfloor \frac{16\text{h} * 60\text{m/h}}{21.5\text{m}} \right\rfloor * 11.56\text{J} = 513.07\text{J},$$

or 8.02 J in average, per node. These results imply that the pair of AA batteries installed in each node would last in average 2122 days (∼5.8 y) when using ACME and 2335 days (∼6.4 y) when using N-Sub. The value obtained for both protocols is reasonable and validates their suitability for application from the perspective of energy consumption. Although N-Sub offers a longer average lifetime per node, this comes at the cost of less flexibility in terms of the MULE's timeout which is a critical feature in opportunistic data collection settings.

[5]This assumption is made just to keep the computations fairly simple. In a real application the subnetwork might be able to maintain a solution for multiple days, weeks or even months.

Chapter 6

Conclusions and future research

6.1 Synthesis

This thesis was devoted to the study and development of routing protocols for MULE-assisted WSNs. Our work is motivated by the benefits that a monitoring system of this kind has to offer to the airports in terms of cost, resiliency, versatility and scalability. The relevance of our research was confirmed at the early stages through a set of interviews with management- and operation-level personnel from different airports. In Chapter 1 we introduced our concept of WSN-based system for integral monitoring at airports. A broad literature review comprising 41 routing protocols led us, in Chapter 2, to identify five important open research challenges, three of which were fully addressed during this thesis:

1. *Incorporation of* ***load balance between subsinks*** *in the problem with MULEs;*
2. *Integration of* ***efficient routing and subsink load balance*** *in a joint optimization;*
3. *Generalization of the technique for a broader class of* ***network topologies****.*

In Chapter 3 we performed the definition, implementation and evaluation of the ***C-Sub*** and ***N-Sub*** protocols, both inspired by routing strategies previously adopted in the literature for scenarios involving MULEs. N-Sub showed superior performance in terms of packet delivery ratio, power consumption and subsink load than C-Sub. Indeed, the routing solution provided by N-Sub is ideal in terms of efficient routing and subsink load when the subnetwork has grid-like structure. However, the need for a more sophisticated routing technique was clear due to the wide variety of network topologies that could arise in our airport monitoring application, for which N-Sub would not be well suited. In Chapter 4 we responded to this need with the development of ***ACME***, an Ant Colony based protocol capable of dealing with the trade-off between efficient routing and subsink load that appears in non grid-like subnetworks. In this chapter we modeled the problem as a multi-objective optimization task where the conflicting objectives of minimization of total path length and minimization of total subsink imbalance

were considered as main targets, while the minimization of total retransmissions was adopted as a secondary, optional target. In that chapter, we also present a sketch for the extensions of ACME that would take into account the other two open challenges we found in our literature review:

4. *Modeling of* ***heterogeneous traffic profiles*** *at the source nodes;*
5. *Explicitly incorporating* ***data aggregation*** *in the protocols and their validation.*

ACME's versatility was verified in Chapter 5 through extensive computer simulation involving a fairly wide class of network structures. The experiments also showed the superiority of ACME over N-Sub in terms of the efficiency of use of the time in contact with the MULE. ACME not only covers the three research challenges listed above, but features various other desirable characteristics such as its ability to optimize the overall network performance in a distributed manner and the possibility of incorporating the priority given by the user to each objective function during the optimization.

6.2 Contributions

Below, we present a list of the major contributions of this thesis.

- ***We carried out one of the first implementations of a MULE-assisted WSN in OMNeT++.*** OMNeT++ is a discrete event simulator specialized in the modeling of networks. For over 20 years it has been consistently used by the research community and proved its suitability for the modeling of plenty of wireless and wired network settings [201]. It stands out for being one of the most popular simulators in the research area of communications networks [202, 203]. Some of its main features are its modular architecture, which allows easy integration of protocols to the system, and its ease of use and debugging highly supported on a user-friendly graphical interface [204].

  Our literature review reveals that a significant portion of the revised studies either: (i) do not specify the simulation environment used; or (ii) conduct simulations in a programming environment not specialized for the modeling of networks. As a direct consequence, various important parameters and characteristics of the wireless communication process are often neglected. A notable example of this issue are the studies that assume that the system is collision-free (e.g., [90, 205]). This problem has been pointed out by other authors [206]

  Out of the five studies that we found addressing a data routing problem in MULE-assisted WSNs, only three used simulation environments specialized for telecommunications: More specifically, [111] simulated in TOSSIM, [109] in NS-2 and [110] in MATLAB. Note that we did not find any study in this topic which had made an implementation in OMNeT++. Thus, the implementations of RPL, R-RPL, C-Sub, N-Sub, B-Sub, ACME, and

the MULE's architecture made in this thesis constitute pioneer implementations of MULE-assisted WSNs in OMNeT++. We verified this claim by searching the triple "*MULE WSN OMNeT*" in Google Scholar under incognito mode, exploring up to the fifth page of results, and finding no articles with these characteristics.

- ***We studied the suitability, for the airport monitoring application, of two routing protocols previously adopted in the literature for MULE-assisted WSNs.*** In Chapter 3 we performed a comparison between two routing protocols for MULE-assisted WSNs previously used in the literature: (i) the subsink currently in contact with the MULE is used as a gateway; and (ii) each node uses the subsink closest to it as a gateway. As explained in that chapter, these protocols were not available as routing protocols in OMNeT++ and the description provided in the articles was lacking of detail. Thus, we had to formally define and implement the two protocols, attach them to suitable node's architectures and also conform the architecture of the MULE in OMNeT++ before becoming able to perform the comparison. Our implementation gave rise to the C-Sub and N-Sub protocols, corresponding to the two cited protocols.

  The main objective of this implementation was to evaluate how the two routing approaches taken from the literature performed when using system parameters specific to the airport monitoring application such as the data gathering rate, number of nodes, and time between visits of the MULE. Our experiments led us to conclude a clear superiority of N-Sub over C-Sub for this application; an outcome that was not simple to anticipate since the results strongly depend on the values used as parameters. Sufficiently lower values of the three parameters listed before, for instance, could lead the system to have similar results using both protocols. Thus, it was necessary to conduct the experiments using realistic parameters in order to reach valid conclusions. The experiments also led us to conclude the ability of N-Sub to deliver optimal results in terms of efficient routing and subsink balance for subnetworks with grid-like topology. Based on this conclusion, we went to consider the need for a more sophisticated technique capable of handling the more general class of network topologies that may appear in the airport monitoring application.

- ***We developed ACME, a routing protocol that integrates efficient routing and load balance among subsinks in a joint optimization.*** Driven by the conclusions of Chapter 3, in Chapter 4 we undertook the design of ACME, our ACO-based protocol capable of dealing with the trade-off between efficient routing and subsink load in non grid-like subnetworks. ACME features a number of characteristics that make it particularly well suited for our application case:

  - ◇ To achieve an efficient use of the time in contact with the MULE and help prevent subsinks from retaining data once the MULE has left the subnetwork, ACME adopts subsink load balance as an optimization criterion;

- ◇ To maintain a reasonable energy consumption, ACME also adopts the sum of lengths of the paths covered by all the packets as an optimization criterion;
- ◇ To avoid any dependency on central nodes with special memory, battery or processing capabilities, ACME works on a decentralized scheme where the most intensive part of the optimization procedure is decomposed into minor local actions performed by the nodes, and only some basic operations are left to the subsinks, enabling the evaluation of each potential solution;
- ◇ To make ACME easily extensible to other objectives of optimization, we adopt a valuation scheme where the quality of each explored solution is quantified at the checkpoint in terms of a generic, normalized objective function which scales the different optimization criteria to a common unit and integrates them into a single fitness measure. The versatility of this approach was highlighted in Chapter 4 through the addition of the minimization of the number of retransmissions as a secondary objective, without the need for any complex adaptation of the optimization algorithm;
- ◇ To take into account the preferences of the user over different optimization criteria, ACME implements a pooled sum of objective functions which receives as input a vector of weights. The default setup where all the weights are given the same value is always a suitable initial choice while the user gets familiar with the functioning of the system and develops the ability to set custom weights;
- ◇ To promote an agile reaction of the subnetwork to eventual node failures and keep open the possibility of including optimization objectives depending on local variables (e.g., the residual energy of neighbor nodes), ACME incorporates local pheromones that serve as an indicator of the quality of the neighbors of each node as next hops for packets forwarded by it.

- ***We performed an improved implementation of the ACO algorithm for WSN data routing compared to previous studies.*** Most of the studies found in the literature proposing routing protocols for WSNs based on ACO use the Ant System (AS) [121], the first version of the family of ACO algorithms. This fact is surprising since more than 20 years ago an improved version of AS was introduced by the same author and since then it has been a preferred choice in most fields where ACO has been applied. This improved version called Ant Colony System (ACS) [170] allows a higher degree of exploration of the solution space, which prevents the algorithm from getting stuck at local optima and avoids a premature convergence. ACME is developed upon ACS, making a step forward in the evolution of the state of the art in WSN routing towards more sophisticated and better performing techniques.

- ***We performed a literature review of routing protocols for WSNs with emphasis on MULE-assisted settings and provided a taxonomic classification of the 41 reviewed papers.*** In Chapter 2 we conducted a review of routing protocols for WSNs. Our survey comprised protocols in four different categories which are of great relevance within the state of the art, namely: (i) flat routing, (ii) cluster-based routing, (iii) routing in settings involving MULEs, and (iv) routing supported in advanced optimization. So far, these topics had not been collectively revised in one literature review, and the coverage of categories (iii) and (iv) in previous surveys is overall scarce. Our survey also comprises a taxonomic classification of all the reviewed protocols which might serve as a reference for the identification and further tackling of relevant research gaps.

- ***With the development of ACME, we resolve one of the key issues for the potential future installation of cost-accessible WSN monitoring systems in airports: efficient data routing.*** We acknowledge that the constitution of the full data monitoring system for airports proposed in Chapter 1 will require several further steps including a variety of actions such as revision and improvement of concept with personnel from the airports, improvement of the method, tests involving real devices, among others. However, the development of ACME makes an important contribution to the maturation of this idea by solving the fundamental problem of efficient data routing. Hopefully, this work will motivate future research putting in place the remaining necessary pieces for the consolidation of the devised airport monitoring system.

6.3 Concluding remarks

The remarks below follow from the analysis of the developments and results achieved along this thesis:

- There is currently a strong heterogeneity between the quality and scope of the monitoring techniques used by airports worldwide, mainly due to the high acquisition, installation and maintenance costs of most technology-based solutions. Supported on the available sensing and wireless communication technologies, and the data routing protocols developed in this thesis, a WSN-based monitoring solution presents an inclusive alternative that could open the doors to many small- and mid-size airports to integral monitoring.

- Subsink load balance plays a fundamental role in the sustainability of MULE-assisted WSNs supported on opportunistic contact. It enables the system to better use any time in contact with the MULE, and also to have higher tolerance to long periods of no contact.

- For subnetworks with grid-like topology, the strategy of using the closest subsink as gateway, implemented in the N-Sub protocol, provides both, optimal routes and optimal subsink balance.

- For subnetworks with irregular topology (i.e., non grid-like), the objectives of efficient routing and subsink balance tend to be in conflict. ACME demonstrated an ability to deal with the trade-off between the two objectives and even to adapt the solution to the preferences of the user for each objective, stated through the optimization weights.
- The design of a protocol supported on light centralization allowed us to find high quality solutions in terms of overall subnetwork performance by means of only local actions of the nodes and the execution of simple algebraic operations at the subsinks. Light centralization offers the benefits of global optimization without the need for central nodes with special processing, energy or memory capabilities as in the full centralization scheme. Indeed, the system is mostly decentralized, except for the basic computations that the subsinks make in order to evaluate the quality of the candidate solutions. Naturally, swarm optimization techniques like the ACO algorithm implemented here are better suited for a system working under this type of scheme.
- Taking advantage of the optimization framework implemented in ACME, in Chapter 4 we proposed an alternative way of mitigating packet collisions to what has traditionally been done in the literature. The experiments conducted in Chapter 5 showed the ability of ACME to effectively reduce the number of retransmissions in the system when this objective is added. Although energy savings were not impressive in our system, which seems to have a nominal low number of collisions even with the objective inactive, this novel approach of dealing with collisions worth further analysis in systems involving larger traffic and thus naturally exposed to greater issues of collisions.
- Among all the experiments presented in Chapter 5, ACME took 5.7 and 11 minutes to find the solutions in the instances that took the least and the most time, respectively. Considering that the layout of the subnetworks are expected to remain unchanged for several weeks or months, and thus there would be no need to relaunch the optimization so often, ACME's resolution times are perfectly suitable for application. Even in the extreme case where the optimization were launched each day, the processing time would be negligible compared to the operational time of the subnetworks.
- The energy consumed by the nodes during the optimization was also very modest. In the worst case, the average consumption per node was 0.67 J, corresponding to the 0.004% of the assumed initial energy load of 18720 J per node obtained from two AA batteries. These numbers show the feasibility of ACME as a routing protocol for application in real airport surface monitoring instances.
- Although the focus of this thesis was on airport surface monitoring, we have envisaged various other applications where ACME could be relevant, for instance: Precision Agriculture [207, 208] and Urban IoT [209, 210]. In those applications the network and the MULE operate in a similar way as in airport monitoring. It would be thus interesting

to evaluate the performance of our method in those contexts, taking into account the particular value of parameters like the number of nodes, speed of the MULE and typical time between visits.

6.4 Scientific production

Published papers

- **Routing in Wireless Sensor Networks for Surveillance of Airport Surface Area**, *International Workshop on Communication Technologies for Vehicles*, pp. 27–38. Springer, 2018.
  In bibliography: [45] Online: find it here
- **Opportunistic Data Collection and Routing in Segmented Wireless Sensor Networks**, *International Conference on Ad-Hoc Networks and Wireless*, pp. 183–195. Springer, 2019.
  In bibliography: [46] Online: find it here

6.5 Research perspectives

For near future development, we propose the following research avenues:

A. On the optimization criteria

• **Residual energy as routing criterion:** the balance on energy consumption at the nodes has been consistently pointed out in the literature as an effective strategy for extending network lifetime [211]. Although the optimization of subsink balance helps to generate overall load balance in the subnetwork, significant improvements could be achieved if this last were explicitly incorporated into the optimization. This could be done, for instance, by maximizing the sum of residual energy of the nodes visited by the packets. This way, the protocol would promote the use of the nodes with largest residual energy after each optimization. The implementation and evaluation of this or any other strategy oriented to explicitly integrate overall load balance in the protocol seems a promising avenue leading to the improvement of ACME's performance.

• **Total energy spent as routing criterion:** in this application it is reasonable to assume sensor nodes provided with a regular pair of AA batteries with nominal load enough to operate for a few years. Although energy supply is thus not a critical issue for us, the elongation of the lifetime of the nodes translates into maintenance cost savings for the airport and therefore, it

must be pursued. In addition, any improvement in terms of energy savings might expand the scope of application of the protocol to fields with stronger energy constraints. An interesting research line would thus point out to the incorporation of total energy consumption as routing criterion during the optimization.

• **Automated prioritization of routing criteria:** this thesis already considered three optimization criteria, and proposes the study of other two in future research. As the optimization problem increases in complexity, it gains relevance the development of some automated methodology oriented to help the user with the definition of the weights indicating the priority of the different objective functions. We propose the construction of such a methodology as the subject of future research. Note that this development may likely not be limited to use in complement to ACME, but could also serve to other optimization-based routing protocols involving multiple objectives of optimization.

B. On the assumptions

• **Nodes with heterogeneous traffic profiles:** as explained in Section 4.5, many practical instances might involve nodes with different traffic profiles. This feature should be integrated to the protocol in order to achieve a more realistic modeling and get rid of the noise that the assumption of homogeneous traffic profiles may cause in the computation of the objective functions. The approach proposed in Section 4.5 might be useful in this regard.

• **Multiple sets of subsinks:** our methods and experiments consider MULEs with fixed trajectory, which for simplicity we also extended to the assumption that the same set of nodes have reach contact with it at each visit. Although the assumption is not very limiting and might in most cases be realistic for airport monitoring, it might also occur that two or more trajectories exist, generating thus two or more sets of subsinks that vary from one visit to the other. An improvement to this method could be reached by incorporating in the optimization the frequency of contact that each subsink has with the MULE so that the system can progressively learn along the time how to optimally distribute the information over the subsinks.

• **MULEs with varying speed during time in contact:** our methods and experiments consider MULEs moving at constant speed during the time in contact with the subnetwork. Once again, for our application this assumption is not very limiting since the service roads at the airports have nominal speed values that the service vehicles must respect. It would be interesting, however, to consider situations where a variation in the speed of the MULE during the time in contact occurs in a systematic manner visit after visit. This situation may

occur, for instance, if part of the segment of contact is curved and the MULE always follows a similar pattern of deceleration and posterior acceleration when it passes over that zone. To take into account this feature, the system could learn from the pattern over the first visits of the MULE and incorporate this information into future optimizations in order to assign greater data loads to the subsinks with greater time in contact and vice versa.

C. On efficient traffic management

• **Incorporate data aggregation to the optimization:** data aggregation is one of the most popular strategies to reduce energy spent. In Section 4.5 we presented a brief sketch on how this strategy could be used as a performance booster implemented in a post-optimization procedure after ACME has delivered a solution. Although this alternative is already expected to significantly improve energy spent by reducing the amount of traffic in the subnetworks, even greater improvements could be reached if data aggregation were already implemented during the optimization. The approach proposed in [212] might be worth exploring to achieve this. They send two batches of ants per round instead of only one as us; the first batch builds the routes without considering data aggregation, and the second one follows the same routes and incorporates data aggregation. The integration of data aggregation into the optimization either by this or other approach is proposed for future research.

D. On the stopping conditions

• **Processing time, convergence and energy spent as stopping conditions:** in its current version, ACME evaluates solutions in an recurrent fashion until a prefixed number of iterations has been reached. Our results from Section 5.2.2 show that an important proportion of the best solutions were found before reaching the number of iterations that we used. Thus, a lower energy spent during the optimization would have been achieved if a stopping condition based on convergence was implemented. On the other hand, stopping conditions based on processing time and energy spent result very convenient from the perspective of the final user, since it is easier to just give a budget for the optimization in any of the two dimensions and let the procedure run until exhausting the budget. Based on this, we propose the integration of these three complementary forms of stopping condition as subject of future development and research.

Bibliography

[1] F. Fanian and M. K. Rafsanjani, "Cluster-based routing protocols in wireless sensor networks: A survey based on methodology," *Journal of Network and Computer Applications*, vol. 142, pp. 111–142, 2019.

[2] L. K. Ketshabetswe, A. M. Zungeru, M. Mangwala, J. M. Chuma, and B. Sigweni, "Communication protocols for wireless sensor networks: A survey and comparison," *Heliyon*, vol. 5, no. 5, p. e01591, 2019.

[3] M. Ghribi and A. Meddeb, "Survey and taxonomy of mac, routing and cross layer protocols using wake-up radio," *Journal of Network and Computer Applications*, vol. 149, p. 102465, 2020.

[4] D. Kandris, C. Nakas, D. Vomvas, and G. Koulouras, "Applications of wireless sensor networks: an up-to-date survey," *Applied System Innovation*, vol. 3, no. 1, p. 14, 2020.

[5] D. Kalubowila and H. Perera, "Wireless sensor networks for intelligent transportation systems,"

[6] A. Elrayes, M. H. Ali, A. Zakaria, and M. H. Ismail, "Smart airport foreign object debris detection rover using lidar technology," *Internet of Things*, vol. 5, pp. 1–11, 2019.

[7] C. D. Prather, *Current Airport Inspection Practices Regarding FOD (foreign Object Debris/damage)*, vol. 26. Transportation Research Board, 2011.

[8] "Bayanat Airports and QinetiQ Sign Partnership Agreement." http://fod-detection.com/2009/05/20/bayanat-airports-and-qinetiq-sign-partnership-agreement/, 2009. Accessed: 2009-05-20.

[9] X. Qunyu, N. Huansheng, and C. Weishi, "Video-based foreign object debris detection," in *2009 IEEE International Workshop on Imaging Systems and Techniques*, pp. 119–122, IEEE, 2009.

[10] Y. Zhongda, L. Mingguang, and C. Xiuquan, "Research and implementation of fod detector for airport runway," in *IOP Conference Series: Earth and Environmental Science*, vol. 304, p. 032050, IOP Publishing, 2019.

[11] P. D. Beasley, G. Binns, R. D. Hodges, and R. J. Badley, "Tarsier/spl r/, a millimetre wave radar for airport runway debris detection," in *First European Radar Conference, 2004. EURAD.*, pp. 261–264, IEEE, 2004.

[12] "Tarsier. Automatic Runway FOD Detection System." https://www.tarsierfod.com/, 2004. Accessed: 2004-01-01.

[13] E. E. Herricks, P. Lazar III, E. Woodworth, and J. Patterson Jr, "Performance assessment of a mobile, radar-based foreign object debris detection system," tech. rep., 2011.

[14] "FODFinder. The Total Solution for FOD Control." http://www.fodfinder.com/, 2011. Accessed: 2011-01-01.

[15] E. E. Herricks, P. Lazar III, E. Woodworth, and J. Patterson Jr, "Performance assessment of an electro-optical-based foreign object debris detection system," tech. rep., 2012.

[16] Z. Han, Y. Fang, H. Xu, and Y. Zheng, "A novel fod classification system based on visual features," in *International Conference on Image and Graphics*, pp. 288–296, Springer, 2015.

[17] "Xsight Systems FOD Solution." https://www.xsightsys.com/, 2013. Accessed: 2013-02-03.

[18] "Affordable Auto-METAR for small airports." https://www.baranidesign.com/news-innovations-blog/2018/5/26/affordable-auto-metar-for-small-airports-finally-a-reality-at-farnborough-airshow-2018, 2018. Accessed: 2018-05-29.

[19] "News and insights on Foreign Object Debris detection systems ." https://fod-detection.com/page/15/, 2010.

[20] C. Perez-Vidal, L. Gracia, C. Carmona, B. Alorda, and A. Salinas, "Wireless transmission of biosignals for hyperbaric chamber applications," *PloS one*, vol. 12, no. 3, p. e0172768, 2017.

[21] T. Ballaa and Z. Galb, "Overview of the internet of things short range communication technologies," in *Proceedings of the 9th International Conference on Applied Informatics (ICAI 2014), Eger, Hungary*, 2014.

[22] O. O. Kazeem, O. O. Akintade, and L. O. Kehinde, "Comparative study of communication interfaces for sensors and actuators in the cloud of internet of things," *Int. J. Internet Things*, vol. 6, no. 1, pp. 9–13, 2017.

[23] T. R. Sheltami, E. M. Shakshuki, and H. T. Mouftah, "A web-based application of telosb sensor network," *Mobile Information Systems*, vol. 7, no. 2, pp. 147–163, 2011.

[24] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "A comparative study of lpwan technologies for large-scale iot deployment," *ICT express*, vol. 5, no. 1, pp. 1–7, 2019.

[25] "ISM Frequency Bands." https://www.everythingrf.com/community/ism-frequency-bands, 2018. Accessed: 2018-04-24.

[26] B. Foubert and N. Mitton, "Long-range wireless radio technologies: A survey," *Future internet*, vol. 12, no. 1, p. 13, 2020.

[27] A. Anand, V. Pejovic, E. M. Belding, and D. L. Johnson, "Villagecell: Cost effective cellular connectivity in rural areas," in *Proceedings of the Fifth International Conference on Information and Communication Technologies and Development*, pp. 180–189, 2012.

[28] S. Hasan, K. Heimerl, K. Harrison, K. Ali, S. Roberts, A. Sahai, and E. Brewer, "Gsm whitespaces: An opportunity for rural cellular service," in *2014 IEEE International Symposium on Dynamic Spectrum Access Networks (DYSPAN)*, pp. 271–282, IEEE, 2014.

[29] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "Overview of cellular lpwan technologies for iot deployment: Sigfox, lorawan, and nb-iot," in *2018 ieee international conference on pervasive computing and communications workshops (percom workshops)*, pp. 197–202, IEEE, 2018.

[30] J. D. Borrero and A. Zabalo, "An autonomous wireless device for real-time monitoring of water needs," *Sensors*, vol. 20, no. 7, p. 2078, 2020.

[31] N. Azmi, L. Kamarudin, M. Mahmuddin, A. Zakaria, A. Shakaff, S. Khatun, K. Kamarudin, and M. Morshed, "Interference issues and mitigation method in wsn 2.4 ghz ism band: A survey," in *2014 2nd International Conference on Electronic Design (ICED)*, pp. 403–408, IEEE, 2014.

[32] B. Vejlgaard, M. Lauridsen, H. Nguyen, I. Z. Kovács, P. Mogensen, and M. Sorensen, "Interference impact on coverage and capacity for low power wide area iot networks," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2017.

[33] D. Zorbas and B. O'Flynn, "Autonomous collision-free scheduling for lora-based industrial internet of things," in *2019 IEEE 20th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pp. 1–5, IEEE, 2019.

[34] A. Lavric and V. Popa, "Performance evaluation of lorawan communication scalability in large-scale wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[35] "Difference Between ATIS, ASOS and AWOS." https://www.thinkaviation.net/difference-between-atis-asos-awos/, 2018. Accessed: 2018-01-24.

[36] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do lora low-power wide-area networks scale?," in *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 59–67, 2016.

[37] D. Bankov, E. Khorov, and A. Lyakhov, "Mathematical model of lorawan channel access with capture effect," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–5, IEEE, 2017.

[38] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of lorawan," *IEEE Communications magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[39] L. Alliance, "A technical overview of lora and lorawan," *White Paper, November*, vol. 20, 2015.

[40] "Air Traffic Control Frequencies." https://www.skyharbor.com/business/ForPilots/AirTrafficControlFrequencies, 2018. Accessed: 2018.

[41] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, "Wireless sensor networks: a survey on recent developments and potential synergies," *The Journal of supercomputing*, vol. 68, no. 1, pp. 1–48, 2014.

[42] Crossbow, *TelosB Mote Platform Crossbow - IEEE 802.15.4*, 2004. Datasheet.

[43] Crossbow, *MICAz - Wirelss Measurement System*, 2006. Datasheet.

[44] R. Petrolo, T. Delot, N. Mitton, A. Molinaro, and C. Campolo, "O-spin: an opportunistic data dissemination protocol for folk-enabled information system in least developed countries," in *International Conference on Ad-Hoc Networks and Wireless*, pp. 43–57, Springer, 2014.

[45] J. Garcia, A. Pirovano, and M. Royer, "Routing in wireless sensor networks for surveillance of airport surface area," in *International Workshop on Communication Technologies for Vehicles*, pp. 27–38, Springer, 2018.

[46] J. Garcia, A. Pirovano, and M. Royer, "Opportunistic data collection and routing in segmented wireless sensor networks," in *International Conference on Ad-Hoc Networks and Wireless*, pp. 183–195, Springer, 2019.

[47] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE wireless communications*, vol. 11, no. 6, pp. 6–28, 2004.

[48] A. Kanavalli, D. Sserubiri, P. D. Shenoy, K. Venugopal, and L. Patnaik, "A flat routing protocol for sensor networks," in *2009 Proceeding of International Conference on Methods and Models in Computer Science (ICM2CS)*, pp. 1–5, IEEE, 2009.

[49] H. Oudani, J. Laassiri, S.-d. Krit, and L. El Maimouni, "Comparative study and simulation of flat and hierarchical routing protocols for wireless sensor network," in *2016 International Conference on Engineering & MIS (ICEMIS)*, pp. 1–9, IEEE, 2016.

[50] Z. Manap, B. M. Ali, C. K. Ng, N. K. Noordin, and A. Sali, "A review on hierarchical routing protocols for wireless sensor networks," *Wireless personal communications*, vol. 72, no. 2, pp. 1077–1104, 2013.

[51] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[52] D. Teodorovic and M. Dell'Orco, "Bee colony optimization–a cooperative learning approach to complex transportation problems," *Advanced OR and AI methods in transportation*, vol. 51, p. 60, 2005.

[53] D. Whitley, "A genetic algorithm tutorial," *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.

[54] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 174–185, 1999.

[55] B. Xu, O. Wolfson, and S. Chamberlain, "Spatially distributed databases on sensors," in *Proceedings of the 8th ACM international symposium on Advances in geographic information systems*, pp. 153–160, 2000.

[56] J. Kulik, W. Heinzelman, and H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks," *Wireless networks*, vol. 8, no. 2-3, pp. 169–185, 2002.

[57] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *Proceedings of the 7th annual international conference on Mobile computing and networking*, pp. 70–84, 2001.

[58] T. Winter, P. Thubert, A. Brandt, J. W. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, and R. K. Alexander, "Rpl: Ipv6 routing protocol for low-power and lossy networks.," *rfc*, vol. 6550, pp. 1–157, 2012.

[59] I. S. AlShawi, L. Yan, W. Pan, and B. Luo, "A fuzzy-gossip routing protocol for an energy efficient wireless sensor networks," in *SENSORS, 2012 IEEE*, pp. 1–4, IEEE, 2012.

[60] D. Bhattacharyya, T.-h. Kim, and S. Pal, "A comparative study of wireless sensor networks and their routing protocols," *Sensors*, vol. 10, no. 12, pp. 10506–10523, 2010.

[61] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM transactions on networking*, vol. 11, no. 1, pp. 2–16, 2003.

[62] F. Zhao, L. J. Guibas, and L. Guibas, *Wireless sensor networks: an information processing approach*. Morgan Kaufmann, 2004.

[63] D. Braginsky and D. Estrin, "Rumor routing algorthim for sensor networks," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pp. 22–31, 2002.

[64] Y. Yu, R. Govindan, and D. Estrin, "Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks," 2001.

[65] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[66] O. Iova, F. Theoleyre, and T. Noel, "Stability and efficiency of rpl under realistic conditions in wireless sensor networks," in *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 2098–2102, IEEE, 2013.

[67] A. Dvir, L. Buttyan, *et al.*, "Vera-version number and rank authentication in rpl," in *2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems*, pp. 709–714, IEEE, 2011.

[68] V. Kiran, S. Rani, and P. Singh, "Towards a light weight routing security in iot using non-cooperative game models and dempster–shaffer theory," *Wireless Personal Communications*, vol. 110, no. 4, pp. 1729–1749, 2020.

[69] R. Rajagopalan and P. K. Varshney, "Data aggregation techniques in sensor networks: A survey," 2006.

[70] O. Younis, M. Krunz, and S. Ramasubramanian, "Node clustering in wireless sensor networks: recent developments and deployment challenges," *IEEE network*, vol. 20, no. 3, pp. 20–25, 2006.

[71] N. Nasser, C. Arboleda, and M. Liliana, "Comparison of clustering algorithms and protocols for wireless sensor networks," in *2006 Canadian Conference on Electrical and Computer Engineering*, pp. 1787–1792, IEEE, 2006.

[72] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Computer communications*, vol. 30, no. 14-15, pp. 2826–2841, 2007.

[73] P. Kumarawadu, D. J. Dechene, M. Luccini, and A. Sauer, "Algorithms for node clustering in wireless sensor networks: A survey," in *2008 4th International Conference on Information and Automation for Sustainability*, pp. 295–300, IEEE, 2008.

[74] S. Soro and W. B. Heinzelman, "Prolonging the lifetime of wireless sensor networks via unequal clustering," in *19th IEEE international parallel and distributed processing symposium*, pp. 8–pp, IEEE, 2005.

[75] V. Pal, G. Singh, and R. Yadav, "Balanced cluster size solution to extend lifetime of wireless sensor networks," *IEEE Internet of Things Journal*, vol. 2, no. 5, pp. 399–401, 2015.

[76] W. B. Heinzelman, *Application-specific protocol architectures for wireless networks*. PhD thesis, Massachusetts Institute of Technology, 2000.

[77] S. Bandyopadhyay and E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, vol. 3, pp. 1713–1723, IEEE, 2003.

[78] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd annual Hawaii international conference on system sciences*, pp. 10–pp, IEEE, 2000.

[79] K. Padmanabhan and P. Kamalakkannan, "Energy efficient adaptive protocol for clustered wireless sensor networks," *International Journal of Computer Science Issues (IJCSI)*, vol. 8, no. 5, p. 296, 2011.

[80] D. Prabha and V. K. Arora, "A survey on leach and its descendant protocols in wireless sensor network," in *international conference on communication, computing & systems*, pp. 162–167, 2014.

[81] R. Kaur, D. Sharma, and N. Kaur, "Comparative analysis of leach and its descendant protocols in wireless sensor network," *International Journal of P2P Network Trends and Technology*, vol. 3, no. 1, pp. 51–55, 2013.

[82] N. Wang and H. Zhu, "An energy efficient algrithm based on leach protocol," in *2012 International Conference on Computer Science and Electronics Engineering*, vol. 2, pp. 339–342, IEEE, 2012.

[83] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on wireless communications*, vol. 1, no. 4, pp. 660–670, 2002.

[84] C. S. J. Rabaey, K. Langendoen, *et al.*, "Robust positioning algorithms for distributed ad-hoc wireless sensor networks," in *USENIX technical annual conference*, pp. 317–327, 2002.

[85] D. Niculescu, "Positioning in ad hoc sensor networks," *IEEE network*, vol. 18, no. 4, pp. 24–29, 2004.

[86] G. Han, J. Jiang, C. Zhang, T. Q. Duong, M. Guizani, and G. K. Karagiannidis, "A survey on mobile anchor node assisted localization in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2220–2243, 2016.

[87] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*, vol. 344. John Wiley & Sons, 2009.

[88] P. J. Rousseeuw and L. Kaufman, "Finding groups in data," *Hoboken: Wiley Online Library*, vol. 1, 1990.

[89] L. Buttyán and P. Schaffer, "Position-based aggregator node election in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 6, no. 1, p. 679205, 2010.

[90] O. Younis and S. Fahmy, "Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on mobile computing*, vol. 3, no. 4, pp. 366–379, 2004.

[91] J.-g. Jia, Z.-w. He, J.-m. Kuang, and Y.-h. Mu, "An energy consumption balanced clustering algorithm for wireless sensor network," in *2010 6th international conference on wireless communications networking and mobile computing (WiCOM)*, pp. 1–4, IEEE, 2010.

[92] S. H. Kang and T. Nguyen, "Distance based thresholds for cluster head selection in wireless sensor networks," *IEEE Communications Letters*, vol. 16, no. 9, pp. 1396–1399, 2012.

[93] A. Manjeshwar and D. P. Agrawal, "Teen: Arouting protocol for enhanced efficiency in wireless sensor networks.," in *ipdps*, vol. 1, p. 189, 2001.

[94] A. Manjeshwar and D. P. Agrawal, "Apteen: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks," in *ipdps*, p. 0195b, Citeseer, 2002.

[95] S. Lindsey and C. S. Raghavendra, "Pegasis: Power-efficient gathering in sensor information systems," in *Proceedings, IEEE aerospace conference*, vol. 3, pp. 3–3, IEEE, 2002.

[96] A. Rais, K. Bouragba, and M. Ouzzif, "Routing and clustering of sensor nodes in the honeycomb architecture," *Journal of Computer Networks and Communications*, vol. 2019, 2019.

[97] N. D. Tan and N. D. Viet, "Sstbc: Sleep scheduled and tree-based clustering routing protocol for energy-efficient in wireless sensor networks," in *The 2015 IEEE RIVF International Conference on Computing & Communication Technologies-Research, Innovation, and Vision for Future (RIVF)*, pp. 180–185, IEEE, 2015.

[98] M. Demirbas, A. Arora, V. Mittal, and V. Kulathumani, "Design and analysis of a fast local clustering service for wireless sensor networks," in *First International Conference on Broadband Networks*, pp. 700–709, IEEE, 2004.

[99] B. Tatham and T. Kunz, "Anchor node placement for localization in wireless sensor networks," in *2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 180–187, IEEE, 2011.

[100] D. K. Yadav, P. Mishra, and S. Behera, "Optimization of anchor nodes in wireless sensor network," in *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*, pp. 1–5, IEEE, 2017.

[101] D. Yi and H. Yang, "Heer–a delay-aware and energy-efficient routing protocol for wireless sensor networks," *Computer Networks*, vol. 104, pp. 155–173, 2016.

[102] R. C. Prim, "Shortest connection networks and some generalizations," *The Bell System Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957.

[103] A. Youssef, M. Younis, M. Youssef, and A. Agrawala, "Wsn16-5: Distributed formation of overlapping multi-hop clusters in wireless sensor networks," in *IEEE Globecom 2006*, pp. 1–6, IEEE, 2006.

[104] Y. K. Sia, S.-Y. Liew, H. G. Goh, and M.-L. Gan, "Spanning multi-tree algorithms for load balancing in multi-sink wireless sensor networks with heterogeneous traffic generating nodes," 2014.

[105] A. T. Erman, T. Mutter, L. van Hoesel, and P. Havinga, "A cross-layered communication protocol for load balancing in large scale multi-sink wireless sensor networks," in *2009 International Symposium on Autonomous Decentralized Systems*, pp. 1–8, IEEE, 2009.

[106] D. Kominami, M. Sugano, M. Murata, and T. Hatauchi, "Controlled potential-based routing for large-scale wireless sensor networks," in *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pp. 187–196, 2011.

[107] G. Pantziou, A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and B. Mamalis, "Mobile sinks for information retrieval from cluster-based wsn islands," in *International Conference on Ad-Hoc Networks and Wireless*, pp. 213–226, Springer, 2009.

[108] C. Konstantopoulos, G. Pantziou, D. Gavalas, A. Mpitziopoulos, and B. Mamalis, "A rendezvous-based approach enabling energy-efficient sensory data collection with mobile sinks," *IEEE Transactions on parallel and distributed systems*, vol. 23, no. 5, pp. 809–817, 2011.

[109] H. Zhao, S. Guo, X. Wang, and F. Wang, "Energy-efficient topology control algorithm for maximizing network lifetime in wireless sensor networks with mobile sink," *Applied Soft Computing*, vol. 34, pp. 539–550, 2015.

[110] S. K. Singh and P. Kumar, "A mobile sinks based data collection scheme for isolated wireless sensor networks," in *Proceedings of 3rd International Conference on Internet of Things and Connected Technologies (ICIoTCT)*, pp. 26–27, 2018.

[111] A. A. Somasundara, A. Kansal, D. D. Jea, D. Estrin, and M. B. Srivastava, "Controllably mobile infrastructure for low energy embedded networks," *IEEE Transactions on mobile computing*, vol. 5, no. 8, pp. 958–973, 2006.

[112] Y.-C. Tseng, F.-J. Wu, and W.-T. Lai, "Opportunistic data collection for disconnected wireless sensor networks by mobile mules," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1150–1164, 2013.

[113] Y. Gurevich and S. Shelah, "Expected computation time for hamiltonian path problem," *SIAM Journal on Computing*, vol. 16, no. 3, pp. 486–502, 1987.

[114] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of heuristics*, vol. 2, no. 1, pp. 5–30, 1996.

[115] I. BoussaïD, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information sciences*, vol. 237, pp. 82–117, 2013.

[116] K. Sorensen, M. Sevaux, and F. Glover, "A history of metaheuristics," *arXiv preprint arXiv:1704.00853*, 2017.

[117] L. Davis, "Handbook of genetic algorithms," 1991.

[118] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[119] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[120] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," 1991.

[121] M. Dorigo, "Optimization, learning and natural algorithms," *PhD Thesis, Politecnico di Milano*, 1992.

[122] C. Darwin and W. F. Bynum, *The origin of species by means of natural selection: or, the preservation of favored races in the struggle for life*. AL Burt New York, 2009.

[123] F. Fausto, A. Reyna-Orta, E. Cuevas, Á. G. Andrade, and M. Perez-Cisneros, "From ants to whales: metaheuristics for all tastes," *Artificial Intelligence Review*, vol. 53, no. 1, pp. 753–810, 2020.

[124] C. Karr and L. M. Freeman, *Industrial applications of genetic algorithms*, vol. 5. CRC press, 1998.

[125] S. O. Tasan and S. Tunali, "A review of the current applications of genetic algorithms in assembly line balancing," *Journal of intelligent manufacturing*, vol. 19, no. 1, pp. 49–69, 2008.

[126] H. G. Goren, S. Tunali, and R. Jans, "A review of applications of genetic algorithms in lot sizing," *Journal of Intelligent Manufacturing*, vol. 21, no. 4, pp. 575–590, 2010.

[127] U. Mehboob, J. Qadir, S. Ali, and A. Vasilakos, "Genetic algorithms in wireless networking: techniques, applications, and issues," *Soft Computing*, vol. 20, no. 6, pp. 2467–2501, 2016.

[128] G.-S. Yao, Z.-X. Dong, W.-M. Wen, and Q. Ren, "A routing optimization strategy for wireless sensor networks based on improved genetic algorithm," *Journal of Applied Science and Engineering,*, vol. 19, no. 2, pp. 221–228, 2016.

[129] J.-L. Liu and C. V. Ravishankar, "Leach-ga: Genetic algorithm-based energy-efficient adaptive clustering protocol for wireless sensor networks," *International Journal of Machine Learning and Computing*, vol. 1, no. 1, p. 79, 2011.

[130] S. Bayraklı and S. Z. Erdogan, "Genetic algorithm based energy efficient clusters (gabeec) in wireless sensor networks," *Procedia Computer Science*, vol. 10, pp. 247–254, 2012.

[131] E. A. Khalil and A. A. Bara'a, "Energy-aware evolutionary routing protocol for dynamic clustering of wireless sensor networks," *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 195–203, 2011.

[132] Y. Wu and W. Liu, "Routing protocol based on genetic algorithm for energy harvesting-wireless sensor networks," *IET Wireless Sensor Systems*, vol. 3, no. 2, pp. 112–118, 2013.

[133] M. Kumar, M. Husain, N. Upreti, and D. Gupta, "Genetic algorithm: Review and application," *Available at SSRN 3529843*, 2010.

[134] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: A review of representations and operators," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

[135] A. Umbarkar and P. Sheth, "Crossover operators in genetic algorithms: a review.," *ICTACT journal on soft computing*, vol. 6, no. 1, 2015.

[136] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM computing surveys (CSUR)*, vol. 45, no. 3, pp. 1–33, 2013.

[137] A. E. Eiben and C. A. Schippers, "On evolutionary exploration and exploitation," *Fundamenta Informaticae*, vol. 35, no. 1-4, pp. 35–50, 1998.

[138] M. A. Kaliakatsos-Papakostas, A. Floros, and M. N. Vrahatis, *Swarm Intelligence and Bio-Inspired Computation: 10. Intelligent Music Composition*. Elsevier Inc. Chapters, 2013.

[139] T. Camilo, C. Carreto, J. S. Silva, and F. Boavida, "An energy-efficient ant-based routing algorithm for wireless sensor networks," in *International workshop on ant colony optimization and swarm intelligence*, pp. 49–59, Springer, 2006.

[140] Z.-w. Shen, Y.-h. Zhu, X.-z. Tian, and Y.-p. Tang, "An ant colony system based energy prediction routing algorithms for wireless sensor networks," in *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1–4, IEEE, 2008.

[141] L. Wang, R. Zhang, and S. Geng, "An energy-balanced ant-based routing protocol for wireless sensor networks," in *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1–4, IEEE, 2009.

[142] G. Wang, Y. Wang, and X. Tao, "An ant colony clustering routing algorithm for wireless sensor networks," in *2009 Third International Conference on Genetic and Evolutionary Computing*, pp. 670–673, IEEE, 2009.

[143] K. Guleria and A. K. Verma, "An energy efficient load balanced cluster-based routing using ant colony optimization for wsn," *International Journal of Pervasive Computing and Communications*, 2018.

[144] J. Yang, M. Xu, W. Zhao, and B. Xu, "A multipath routing protocol based on clustering and ant colony optimization for wireless sensor networks," *Sensors*, vol. 10, no. 5, pp. 4521–4540, 2010.

[145] M. Ziyadi, K. Yasami, and B. Abolhassani, "Adaptive clustering for energy efficient wireless sensor networks based on ant colony optimization," in *2009 Seventh Annual Communication Networks and Services Research Conference*, pp. 330–334, IEEE, 2009.

[146] S. Kuan, H. Ong, and K. M. Ng, "Solving the feeder bus network design problem by genetic algorithms and ant colony optimization," *Advances in Engineering Software*, vol. 37, no. 6, pp. 351–359, 2006.

[147] C. M. Coleman, E. J. Rothwell, and J. E. Ross, "Investigation of simulated annealing, ant-colony optimization, and genetic algorithms for self-structuring antennas," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 4, pp. 1007–1014, 2004.

[148] E. Mazloumi, M. Mesbah, A. Ceder, S. Moridpour, and G. Currie, "Efficient transit schedule design of timing points: a comparison of ant colony and genetic algorithms," *Transportation Research Part B: Methodological*, vol. 46, no. 1, pp. 217–234, 2012.

[149] J. M. Kahn, R. H. Katz, and K. S. Pister, "Emerging challenges: Mobile networking for "smart dust"," *Journal of Communications and Networks*, vol. 2, no. 3, pp. 188–196, 2000.

[150] A. Benzerbadj, B. Kechar, A. Bounceur, and B. Pottier, "Energy efficient approach for surveillance applications based on self organized wireless sensor networks," *Procedia Computer Science*, vol. 63, pp. 165–170, 2015.

[151] F. Lalem, R. Kacimi, A. Bounceur, and R. Euler, "Boundary node failure detection in wireless sensor networks," in *2016 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–6, IEEE, 2016.

[152] T. Clausen and U. Herberg, "Comparative study of rpl-enabled optimized broadcast in wireless sensor networks," in *2010 Sixth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pp. 7–12, IEEE, 2010.

[153] M. Saoudi, A. Bounceur, R. Euler, T. Kechadi, and A. Cuzzocrea, "Energy-efficient data mining techniques for emergency detection in wireless sensor networks," in *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing,*

Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), pp. 766–771, IEEE, 2016.

[154] P. G. Namboothiri and K. M. Sivalingam, "Performance of a multi-channel mac protocol based on ieee 802.15. 4 radio," in *2009 IEEE 34th Conference on Local Computer Networks*, pp. 313–316, IEEE, 2009.

[155] Y. M. Yusof, A. M. Islam, S. Baharun, M. G. Hamza, K. A. A. Basit, and N. A. Ahmad, "An environmental sensing experiment using ieee 802.15. 4 radio hop of the wsn telosb motes,"

[156] J. D. Peter, A. H. Alavi, and B. Javadi, *Advances in Big Data and Cloud Computing: Proceedings of ICBDCC18*, vol. 750. Springer, 2018.

[157] J. Llosa, I. Vilajosana, X. Vilajosana, and J. M. Marquès, "Design of a motion detector to monitor rowing performance based on wireless sensor networks," in *2009 International Conference on Intelligent Networking and Collaborative Systems*, pp. 397–400, IEEE, 2009.

[158] J. M. Mora-Merchan, D. Larios, J. Barbancho, F. J. Molina, J. L. Sevillano, and C. León, "mtossim: A simulator that estimates battery lifetime in wireless sensor networks," *Simulation Modelling Practice and Theory*, vol. 31, pp. 39–51, 2013.

[159] B. Jang, J. B. Lim, and M. L. Sichitiu, "An asynchronous scheduled mac protocol for wireless sensor networks," *Computer Networks*, vol. 57, no. 1, pp. 85–98, 2013.

[160] A. Zouinkhi, K. Mekki, and M. N. Abdelkrim, "Application and network layers design for wireless sensor network to supervise chemical active product warehouse," *arXiv preprint arXiv:1501.01193*, 2015.

[161] A. Varga, "Omnet++ discrete event simulation system. user manual," *Omnet Community*, 2005.

[162] M. Dorigo, E. Bonabeau, and G. Theraulaz, "Ant algorithms and stigmergy," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 851–871, 2000.

[163] S. Goss, S. Aron, J.-L. Deneubourg, and J. M. Pasteels, "Self-organized shortcuts in the argentine ant," *Naturwissenschaften*, vol. 76, no. 12, pp. 579–581, 1989.

[164] E. Bonabeau, M. Dorigo, D. d. R. D. F. Marco, G. Theraulaz, G. Théraulaz, *et al.*, *Swarm intelligence: from natural to artificial systems*. No. 1, Oxford university press, 1999.

[165] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Inspiration for optimization from social insect behaviour," *Nature*, vol. 406, no. 6791, pp. 39–42, 2000.

[166] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[167] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," in *2011 sixth annual ChinaGrid conference*, pp. 3–9, IEEE, 2011.

[168] A. S. Simaria and P. M. Vilarinho, "2-antbal: An ant colony optimisation algorithm for balancing two-sided assembly lines," *Computers & Industrial Engineering*, vol. 56, no. 2, pp. 489–506, 2009.

[169] J. Betancourt, F. Bachoc, T. Klein, and Gamboa, "Technical report: Ant colony based model selection for functional-input gaussian process regression. Ref. D3.b (WP3.2), *RISCOPE project*." https://hal.archives-ouvertes.fr/hal-02532713, Apr. 2020. hal-02532713.

[170] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on evolutionary computation*, vol. 1, no. 1, pp. 53–66, 1997.

[171] Z. A. Othman, H. M. Rais, and A. R. Hamdan, "Embedding malaysian house red ant behavior into an ant colony system," *Journal of Computer Science*, vol. 4, no. 11, p. 934, 2008.

[172] R. T. Neto and M. Godinho Filho, "Literature review regarding ant colony optimization applied to scheduling problems: Guidelines for implementation and directions for future research," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 150–161, 2013.

[173] "Chapter 6: Medium Access Control Layer ." https://www3.nd.edu/~cpoellab/teaching/cse40815/Chapter6.pdf, 2010. Fundamentals of Wireless Sensor Networks: Theory and Practice.

[174] "Rescaling and shifting." https://www.soph.uab.edu/sites/edu.ssg/files/People/MBeasley/Courses/LinearTransformation.pdf.

[175] K. Chang, "Multiobjective optimization and advanced topics," *Design Theory and Methods Using CAD/CAE*, pp. 325–406, 2015.

[176] J. Arora, "Multiobjective optimum design concepts and methods," *Introduction to optimum design. Elsevier, Amsterdam*, pp. 657–679, 2012.

[177] O. Grodzevich and O. Romanko, "Normalization and other topics in multi-objective optimization," 2006.

[178] L. N. Veldt, *Optimization Frameworks for Graph Clustering.* PhD thesis, Purdue University Graduate School, 2019.

[179] F. Della Croce and R. Scatamacchia, "The longest processing time rule for identical parallel machines revisited," *Journal of Scheduling*, vol. 23, no. 2, pp. 163–176, 2020.

[180] D. Petrovic, R. C. Shah, K. Ramchandran, and J. Rabaey, "Data funneling: Routing with aggregation and compression for wireless sensor networks," in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.*, pp. 156–162, IEEE, 2003.

[181] C. Luo, F. Wu, J. Sun, and C. W. Chen, "Compressive data gathering for large-scale wireless sensor networks," in *Proceedings of the 15th annual international conference on Mobile computing and networking*, pp. 145–156, 2009.

[182] H. M. Ammari, A. Shaout, and F. Mustapha, "Sensing coverage in three-dimensional space: A survey," in *Sensor Technology: Concepts, Methodologies, Tools, and Applications*, pp. 989–1015, IGI Global, 2020.

[183] C.-Y. Chang, C.-C. Chen, and C.-C. Liu, "A novel approximation algorithm for minimum geometric disk cover problem with hexagon tessellation," in *Advances in Intelligent Systems and Applications-Volume 1*, pp. 157–166, Springer, 2013.

[184] I. Khoufi, P. Minet, A. Laouiti, and S. Mahfoudh, "Survey of deployment algorithms in wireless sensor networks: coverage and connectivity issues and challenges," 2017.

[185] List of longest runways, "List of longest runways — Wikipedia, the free encyclopedia," 2020. [Online; accessed 05-May-2020].

[186] R. Budampati and S. Kolavennu, *Industrial Wireless Sensor Networks: Monitoring, Control and Automation.* Elsevier, 2015.

[187] M. Randall and A. Lewis, "A parallel implementation of ant colony optimization," *Journal of Parallel and Distributed Computing*, vol. 62, no. 9, pp. 1421–1432, 2002.

[188] B. Barán and M. Schaerer, "A multiobjective ant colony system for vehicle routing problem with time windows.," in *Applied informatics*, pp. 97–102, 2003.

[189] S.-T. Lo, R.-M. Chen, Y.-M. Huang, and C.-L. Wu, "Multiprocessor system scheduling with precedence and resource constraints using an enhanced ant colony system," *Expert Systems with Applications*, vol. 34, no. 3, pp. 2071–2081, 2008.

[190] R Core Team, *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2017.

[191] R. Soua, *Wireless sensor networks in industrial environment: energy efficiency, delay and scalability.* PhD thesis, 2014.

[192] M. Rajesh and J. Gnanasekar, "Hop-by-hop channel-alert routing to congestion control in wireless sensor networks," *Control Theory and Informatics*, vol. 5, no. 4, pp. 1–11, 2015.

[193] O. Yilmaz, S. Demirci, Y. Kaymak, S. Ergun, and A. Yildirim, "Shortest hop multipath algorithm for wireless sensor networks," *Computers & Mathematics with Applications*, vol. 63, no. 1, pp. 48–59, 2012.

[194] D. Carels, N. Derdaele, E. De Poorter, W. Vandenberghe, I. Moerman, and P. Demeester, "Support of multiple sinks via a virtual root for the rpl routing protocol," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, pp. 1–23, 2014.

[195] S. Isik, M. Y. Donmez, and C. Ersoy, "Multi-sink load balanced forwarding with a multi-criteria fuzzy sink selection for video sensor networks," *Computer Networks*, vol. 56, no. 2, pp. 615–627, 2012.

[196] H. Yoo, M. Shim, D. Kim, and K. H. Kim, "Global: A gradient-based routing protocol for load-balancing in large-scale wireless sensor networks with multiple sinks," in *The IEEE symposium on Computers and Communications*, pp. 556–562, IEEE, 2010.

[197] Silva, Ermenando - Airports Council International (ACI), "Aerodrome Operational Management." *Aerodrome Certification Workshop, Airport Excellence in Safety programme (APEX)*, 02, 2019. Retrieved on 29/05/2020 from shorturl.at/cwOPY.

[198] Brussels Airport, "Traffic Rules." *Brussels Airport Handbook*, 10, 2019. Retrieved on 29/05/2020 from https://media.brusselsairport.be/bruweb/default/0001/26/6afb200ed2862087c14ec46597e1f81abce5041b.pdf.

[199] Winnipeg James Armstrong Richardson International Airport, "Airport Traffic Directives for the Operation of Vehicles Airside." *Airport Traffic Directives*, 02, 2018. Retrieved on 29/05/2020 from https://www.waa.ca/uploads/ck/files/Airport_Traffic_Directives_for_the_Operation_of_Vehicles_Airside_Feb2018.pdf.

[200] MBJ Airports Limited - Sangster International Airport, "Airport Traffic Directives for Sangster International Airport," 01, 2012. Retrieved on 29/05/2020 from http://www.mbjairport.com/pdfs/Airside%20Traffic%20Directives(1).pdf.

[201] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. K. Haneveld, T. E. Parker, O. W. Visser, H. S. Lichte, and S. Valentin, "Simulating wireless and mobile networks in omnet++ the mixim vision," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pp. 1–8, 2008.

[202] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, p. 60, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2008.

[203] T. Gamer and M. Scharf, "Realistic simulation environments for ip-based networks," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pp. 1–7, Citeseer, 2008.

[204] Y. A. Sekercioglu, A. Varga, and G. K. Egan, "Parallel simulation made easy with omnet++," in *European Simulation Symposium 2003*, pp. 493–499, SCS Europe Publishing House, 2003.

[205] M. Youssef, A. Youssef, and M. Younis, "Overlapping multihop clustering for wireless sensor networks," *IEEE transactions on parallel and distributed systems*, vol. 20, no. 12, pp. 1844–1856, 2009.

[206] M. Saleem, G. A. Di Caro, and M. Farooq, "Swarm intelligence based routing protocol for wireless sensor networks: Survey and future directions," *Information Sciences*, vol. 181, no. 20, pp. 4597–4624, 2011.

[207] N. Brinis, L. A. Saidane, *et al.*, "Context aware wireless sensor network suitable for precision agriculture," *Wireless Sensor Network*, vol. 8, no. 01, p. 1, 2016.

[208] T. Ojha, S. Misra, and N. S. Raghuwanshi, "Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges," *Computers and Electronics in Agriculture*, vol. 118, pp. 66–84, 2015.

[209] D. Tacconi, D. Miorandi, I. Carreras, F. Chiti, and R. Fantacci, "Using wireless sensor networks to support intelligent transportation systems," *Ad Hoc Networks*, vol. 8, no. 5, pp. 462–473, 2010.

[210] F.-J. Wu, C.-F. Huang, and Y.-C. Tseng, "Data gathering by mobile mules in a spatially separated wireless sensor network," in *2009 tenth international conference on mobile data management: systems, services and middleware*, pp. 293–298, IEEE, 2009.

[211] C. Wang and W. Wu, "A load-balance routing algorithm for multi-sink wireless sensor networks," in *2009 International Conference on Communication Software and Networks*, pp. 380–384, IEEE, 2009.

[212] R. Misra and C. Mandal, "Ant-aggregation: ant colony algorithm for optimal data aggregation in wireless sensor networks," in *2006 IFIP International Conference on Wireless and Optical Communications Networks*, pp. 5–pp, IEEE, 2006.

Appendices

Appendix A: ACME timing

This section explains the computation of the time windows assigned for the different procedures in ACME. All these time windows consider worst case boundaries on the time to execute the tasks, preventing thus any overlapping or conflict between them. Along this section we consider the following notation:

- $\diamond$ n number of regular nodes.
- $\diamond$ K number of subsinks.
- $\diamond$ N total number of nodes ($n + K$).
- $\diamond$ π maximum number of retransmissions.
- $\diamond$ μ_p packet delivery time[1].
- $\diamond$ μ_a ACK delivery time.
- $\diamond$ μ_w DIO or ADV holding time.
- $\diamond$ t_{pa} packet delivery time plus ACK delivery time ($\mu_p + \mu_a$).
- $\diamond$ t_{pw} packet delivery time plus DIO or ADV holding time ($\mu_p + \mu_w$).

The time windows implemented in ACME are summarized below:

During ACME's initialization

- $\diamond$ ***Time to discover the number of regular and subsink nodes:*** as the first action in Step 1.2, the checkpoint launches the construction of an RPL tree which the nodes use to make it aware of their role as either regular node or subsink. The checkpoint needs to know for how long to wait for replies of the nodes before proceeding with the next actions. Since at this point it is not aware of the number of nodes in the subnetwork, the checkpoint sets a timer timer sufficiently long to allow the formation of an RPL tree in a considerably large subnetwork. More precisely, it sets a timer of 5 s for this

[1]Recall, from Section Section 4.3.3, that in OMNeT++ the packet delivery time is computed as the sum of the: (i) backoff time, (ii) time for clear channel assessment, (iii) time to switch from reception to transmission state, and (iv) transmission time (packet size/bitrate).

procedure, which is enough to let one packet travel more than 200 hops up the tree and 200 hops back to the checkpoint[2].

◇ ***Time between launching of RPL trees:*** as the second action in Step 1.2, the checkpoint asks the other subsinks, one by one, to launch an RPL tree. At this point, the checkpoint already knows the number of nodes in the subnetwork, and thus, it uses this information to set a timer between the formation of each pair of RPL trees. The timer is set equal to $N * t_{pw}$ which is an upper bound for the construction time on an RPL tree computed as if the DIO passed through all the nodes sequentially forming a chain instead of a tree.

◇ ***Time for neighbor discovery:*** during this procedure, carried out in Step 1.3, every node announces itself to its neighbors so that these last can consider it as a potential next hop for any packet they receive. To mitigate collisions during this phase, the nodes set random timers to send their messages. The range for the generation of those random timers should be larger for more densely populated subnetworks. Since at this point the nodes do not know the number of neighbors they have, we set the range of the timers as a function of the total number of nodes in the subnetwork. More precisely, we set the range $[0, N * t_{pw}]$ where the upper limit considers the packet sent by all the nodes in the subnetwork as a way to give room to the process and prevent conflicts between neighbor nodes.

◇ ***Time to set up*** $\bar{\Delta}_2^{\max}$***:*** during neighbor discovery, the checkpoint informs the nodes the time they should wait to perform pheromone initialization and launch the optimization. The time until the optimization initialization must take into account the procedure for setting up the scaling factor for O_2, which takes place at the beginning of Step 1.4. During this procedure, each node sends a tiny packet to its nearest subsink, which counts the number nodes it has assigned and transfers this information to the checkpoint for the consolidation of the information and the subsequent computation of the scaling factor. We use the value $(N * t_{pa} * \pi) + (N * t_{pa} * \pi)$ as an upper bound for this time. The first term of the expression denotes the largest time a regular node could take to reach its G-subsink. The second term, on the other hand, denotes the largest time a subsink could take to reach the checkpoint.

◇ ***Time to perform B-Sub:*** the time required to perform this auxiliary protocol should also be taken into account by the checkpoint when indicating the nodes how long to wait to launch the optimization. We use the value $(N*t_{pa}*\pi*K)+(N*t_{pa}*\pi*3)+(N*t_{pw}*\pi)$ as an upper bound for this time. The first term of the expression denotes the largest

[2]The timer for the discovery of the number of regular and subsink nodes is set considering a t_{pa} of 0.004 s and a t_{pw} of 0.02 s.

time a regular node could take to reach all the subsinks. The second term denotes the largest time a subsink could take to reach the closest subsink; this time is multiplied by 3 which is the maximum number of sweeps that will be executed in B-Sub. Finally, the third term denotes the largest time the checkpoint could take to indicate the nodes the result of this procedure.

Within ACME's exploration

◇ ***Time between ants:*** in order to test candidate solutions, the nodes generate ants in a coordinated manner and send them through the subnetwork in direction of the corresponding G-subsinks. The nodes need to know how much time to wait between the generation of ants from consecutive solutions. In the worst case, the node must wait for all the ants to reach their G-subsinks, for all the subsinks to share with the checkpoint the information they have about this solution, and for the Fpacks to reach all the nodes during global pheromone update. We use the value $(N * t_{pa} * \pi) + (N * t_{pa}) + (N * t_{pw})$ as an upper bound for this overall time. The first term of the expression denotes the largest time a regular node could take to reach its G-subsink. The second term denotes the largest time a subsink could take to reach the checkpoint. Finally, the third term denotes the largest time the checkpoint could take to indicate the nodes the result of this procedure.

Within B-Sub

◇ ***Time to start the first sweep:*** when B-Sub starts, the first action is that all the nodes send a packet to each subsink indicating the number of hops to reach it. Once this action is complete, the checkpoint must trigger the first sweep of packet assignment. The checkpoint thus needs to know for how long to wait for all the subsinks receiving their packets. We use the value $N * t_{pa} * \pi * K$ as an upper bound for this time. This product denotes the largest time a regular node could take to reach all the subsinks.

▷ ***Remark: no need for clock synchronization.*** We have made an effort to make the system highly autonomous with support on ACME. Part of this autonomy comes from the fact that there is no need for clock synchronization between the nodes. When the MULE has the first contact with the subnetwork, the checkpoint starts orchestrating the actions making part of network discovery. During neighbor discovery, in Step 1.3, the checkpoint informs the nodes the time they should wait to perform pheromone initialization and launch the optimization. The nodes receive this information from the checkpoint at slightly different times, and thus they start the optimization a little out of sync. This lag is by no means a problem since anyway, we set a random timer r_i at each node for the generation of the first

ant aimed to prevent collisions at the source nodes, and the lag, being very short, can be considered as part of this timer. From that moment and on, each node starts sending ants each T_{ants} units as displayed in Figure 6.1.

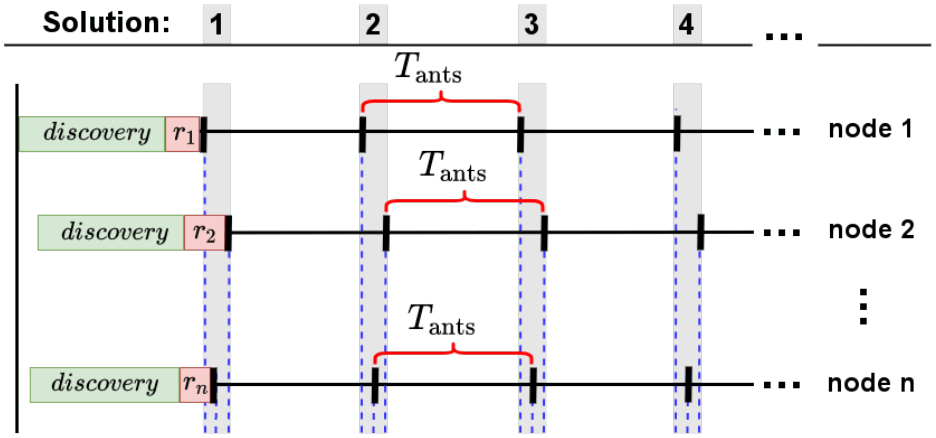


Figure 6.1: Timing for initialization and generation of candidate solutions in ACME.

Appendix B: Generation of random subnetworks

Algorithm 3 is used for the generation of the random base topologies used in the experiments of Chapter 5.

Algorithm 3 Generation of random base topologies

- **n:** number of nodes to locate; **xlim:** limits of the sensing field
- **R:** communication range; **CD:** connectivity degree

```
1: procedure randGen
2:     select an initial coordinate and locate a node
3:     update coordinates vector
4:     for <i=2:n> do
5:         select a node u from the list of nodes already located
6:         while < TRUE > do
7:             ε ← rand() : generate a random number
8:             if < ε ≤ CD > then
9:                 compute distance to all the other nodes
10:                identify the nodes within a distance ≤ 2R : nodes feasible for matching
11:                if < at least one feasible node > then
12:                    randomly pick a feasible node v as matching node
13:                    compute the coordinate of a new node : common neigbor to u and v
14:                else
15:                    aleat ← TRUE : used later to trigger selection of random coordinate
16:                end if
17:            else
18:                aleat ← TRUE
19:            end if
20:            if < aleat > then
21:                pick a random angle in the range of node u
22:                locate a new node at the random angle
23:            else
24:                break
25:            end if
26:        end while
27:    end for
28: end procedure
```

We remark that this algorithm is only used for the generation of the instances involving a random base topology. It does not make part of the procedures performed by ACME.