Optimization of Communications in Multi-Sink Wireless Sensor Networks

Lucas de Araujo Marques Leão

UBFC
UNIVERSITÉ
BOURGOGNE FRANCHE-COMTÉ

UFC
UNIVERSITÉ
DE FRANCHE-COMTÉ

THÈSE DE DOCTORAT DE L'ÉTABLISSEMENT UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

PRÉPARÉE À L'UNIVERSITÉ DE FRANCHE-COMTÉ

École doctorale n°37
Sciences Pour l'Ingénieur et Microtechniques

Doctorat d'Informatique

par

LUCAS AUGUSTO DE ARAUJO MARQUES LEÃO

Optimization of Communications in Multi-Sink Wireless Sensor Networks

Thèse présentée et soutenue à Besançon, le 30 novembre 2018

Composition du Jury :

BOUMERDASSI SELMA	Maître de Conférence HDR au CNAM Paris	Rapporteur
FLAUZAC OLIVIER	Professeur à l'Université de Reims-Champagne-Ardenne	Rapporteur
HILT BENOIT	Maître de Conférence HDR à l'Université de Haute-Alsace	Examinateur
PHILIPPE LAURENT	Professeur à l'Université de Franche-Comté	Président
FELEA VIOLETA	Maître de Conférence à l'Université de Franche-Comté	Co-directeur de thèse
GUYENNET HERVÉ	Professeur à l'Université de Franche-Comté	Directeur de thèse

N° | X | X | X |

école doctorale **sciences pour l'ingénieur et microtechniques**

Title: Optimization of Communications in Multi-Sink Wireless Sensor Networks

Keywords: wireless sensor networks, multiple sinks, routing, unicast, anycast, multicast, reliability, longevity, timeliness, Contiki, COOJA, FIT IoT-LAB, WSN430

Abstract:

The design of a wireless sensor network may present numerous challenges, such as scalability, reliability, longevity and timeliness. The existence of multiple sinks may increase the network reliability and facilitates the scalability. However, this improvement is dependent on the routing approach, that must be tailored to help achieving the desired performance goals. From this perspective, the objective of this work is to find ways of optimizing the communication in multi-sink wireless sensor networks considering the problems related to scalability, longevity (network lifetime), reliability (packet delivery) and timeliness (latency). We investigate the trades among data delivery time and energy consumption as key metrics for communication quality and efficiency. For that matter, we propose different routing algorithms, covering all three main communciations schemes (unicast, anycast and multicast). The executed simulations show that our approaches are capable of optimizing the communication, especially in terms of latency and network lifetime. Experiments on the FIT IoT-Lab platform also provide meaningful insights of the performance of our multicast solution in real environment conditions.

Titre : Optimization of Communications in Multi-Sink Wireless Sensor Networks

Mots-clés : réseaux de capteurs sans fil, multiples points de collecte, routage, unicast, anycast, multicast, fiabilité, durée de vie, latence, Contiki, COOJA, FIT IoT-LAB, WSN430

Résumé :

La conception d'un réseau de capteurs sans fil peut présenter de nombreux défis, tels que le passage à l'échelle, la fiabilité, la longévité et la latence. L'existence de plusieurs points de collecte peut augmenter la fiabilité du réseau et facilite le passage à l'échelle. Toutefois, cette amélioration dépend de l'approche de routage, qui doit être adaptée pour atteindre les objectifs de performance souhaités. Dans cette optique, l'objectif de ce travail est de trouver des moyens pour optimiser la communication dans les réseaux de capteurs sans fil à multiples points de collecte en tenant compte des problèmes liés au passage à l'échelle, à la durée de vie du réseau, à la fiabilité (livraison des paquets) et à la minimisation de la latence. Nous étudions les points d'équilibre entre le délai et la consommation d'énergie en tant que paramètres clés pour la qualité et l'efficacité de la communication. Pour ce faire, nous proposons différents algorithmes de routage, couvrant les trois principaux schémas de communication (unicast, anycast et multicast). Les simulations effectuées montrent que nos approches sont capables d'optimiser la communication, notamment en termes de latence et de durée de vie du réseau. Des expériences sur la plateforme FIT IoT-Lab fournissent également des indications significatives sur les performances de notre solution multicast dans des conditions réelles.

UBFC UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

Université Bourgogne Franche-Comté
32, avenue de l'Observatoire
25000 Besanon, France

ACKNOWLEDGMENTS

I express here my gratitude to...

My dear supervisors, in particular to Dr. Hervé Guyennet, who patiently passed on his knowledge and guidance during this process of growth. And especially to my co-supervisor Dr. Violeta Felea for the patience, the support, the unconditional dedication, the great expertise and absolute care during the development of this work.

Dr. Selma Boumerdassi and Dr. Olivier Flauzac, for their willingness in revising and commenting on my work. And the members of the jury, Dr. Benoit Hilt and Dr. Laurent Philippe for accepting our invitation to be part of this evaluation committee.

The DISC department and the doctoral school SPIM for the financial support in conferences and workshops.

The Brazilian National Council for Scientific and Technological Development (CNPq) for the scholarship that made this work possible.

The staff from the Mésocentre de calcul de Franche-Comté for all the support.

Maxime, Vincent, Paola, Zhouyang, Yiwei, Qianqian for the great help during the simulations and experiments.

The DISC department staff, for the absolute help with all the paper work and equipment.

The french people in general for receiving me so well and so warmly.

My dear department colleagues, PhD students and interns, for the support and for the good time every Wednesday.

My dear friends, always present in my life, supporting and encouraging all my dreams and giving me the strength to move on and overcome each obstacle. Especially to Mauricio for the extra incentives, for believing me, and for standing by me. To Ivelize for the amazing talks and memorable trips. To Vania, Lilian, Beatriz, Edgar, André and Leticia for the friendship. To Bruna, who was my guardian angel when I first arrived in Besaçon. To Kizzyy, Lemia and Lydia, for being amazing and friendly. To Florent and Lisa, for the many good moments, the support and friendship. To Thomas and Sophie, for the fun play. And of course to my amazing friend, work colleague, french teacher, psychologist, confident and so on, Karla - you made my life much easier here.

Finally, to my parents Alberto and Elisabete, to whom I owe my life, for the infinite support, for being sensitive and understanding, and for standing by me. And to my brothers André and Pedro, who helped me become who I am. You all are my safe haven and my source of energy. We are stronger as a family, and no physical distance can destroy this.

CONTENTS

INTRODUCTION

GENERAL CONTEXT

There are many definitions for Wireless Sensor Networks (WSN), but the most frequent descriptions point to a joint characterization of hardware and software. The WSN is defined as a group of small electronic devices with limited processing resources, energy, and radio range, organized as an ad-hoc wireless network, tasked with monitoring or tracking activities.

Research in wireless sensor networks has been a hot topic for many years now. Many advances in WSN have been proposed with numerous researches being carried out in different domains [1]. The continuous development of the WSN solutions is directly associated to the evolution of correlated technologies (microelectronics, sensory, communications, etc.) along with the combination of WSN with other research areas. For instance, the work in [2] points out WSN as an enabling technology for the IoT (Internet of Things). It means that the development and evolution of IoT applications may also trigger interesting researches in WSN.

In some application fields there is a large perimeter to be monitored, as for example in Smart Cities, oriented to intelligent buildings, where sensors are deployed in every apartment to monitor metrics as water, energy consumption or other different interesting metrics. All these information are collected and transferred to a central node, meant to compute statistical consumption information, thus allowing adaptation of energy or water flow. The deployment of a WSN to monitor such application would lead to the existence of hundreds to thousands of sensor nodes, with scalability featuring as a key aspect of the implementation [3].

In this context, one way to tackle the scalability problem is to consider the deployment of multiple sinks. The scalability must be managed because of two reasons: congestion and increased path length. In single sink networks, all the traffic is directed to one point. The existence of several sinks allows a better distribution of the traffic, helping to increase the network lifetime.

When the area of deployment increases, using the same sensor technology for the communication range, longer path lengths from nodes to single sinks are generated. Adding extra sinks causes the average length of a routing path to decrease, because geographic distances between nodes and sinks are smaller. Consequently, packet latency and energy consumption are also decreased as benefit of a shorter path length. Moreover, reliability of communications is improved. Especially when packets are addressed to multiple sinks, generating a beneficial redundancy, which helps to guarantee the delivery of the packets. In theses cases, the number of packets increases and the routing strategy must be adapted in order to cope with the new demand.

In this work we focus on the optimization of the communication in terms of data delivery

time and network lifetime, using the existence of multiple sinks as a precondition to the optimization strategy. We also focus on the differences and challenges that multiple sinks introduce at the routing level.

RESEARCH PROBLEM

One important communication optimization objective is the increase of the network lifetime, with solutions trying to reduce the energy consumption through numerous strategies. Another relevant objective is related to the time sensitive communication, with solutions trying to reduce the transmission delays in order to reduce the overall latency. When these two optimizations are associated, a conflict is created. That is because strategies to extend the network lifetime may also lead to an increase of the overall latency. In that sense, finding a point of balance between the two objectives becomes a challenge.

According to [4] the design and implementation of WSN may have many challenges, including the scalability problem. A large WSN may be impracticable considering the existence of a single sink. The amount of data gathered by the nodes and forwarded to the sink may overload the network capacity, increasing the latency, and leading the nodes in the sink neighborhood to an early depletion of the energy. Therefore, Multi-Sink WSN (MS-WSN) is presented as an important solution for a large group of challenges (scalability, reliability, real-time communication, network lifetime optimization). Nevertheless, [4] states that the class of routing protocols traditionally applied to single sink WSN may not cope with the complexities and particularities associated to the multiple sinks WSN. Depending on the application demand, packets may be forwarded to one specific sink, to any of the sinks, or even to all sinks. In that sense, it is possible to list different communication schemes for WSN with multiple sinks. From a routing perspective, we may list three communication schemes: unicast, anycast and multicast. For each one of the communication schemes, the routing protocol must be tailored to achieve the desired goals.

The optimization of the communication at the level of the routing protocol for WSN having multiple sinks depends on the communication scheme. The optimization strategy applied to a unicast routing protocol can not be directly applied to a multicast approach. In a unicast solution the packets are forwarded to a single sink among all sinks, the solution strategy may consider selecting the path to the sink node presenting the best network metrics. However, when the packet has to be sent to all sinks, the strategy changes and the selected paths may also consider balancing the load, distributing the packets through different paths or merging paths to a group of sinks.

Most of the existing solutions for multi-sink wireless sensor networks consider the packet delivery to only one sink among the group of sinks. The strategies vary between unicast and 1-anycast communication schemes. These are good strategies for the scalability problem and latency optimization, but they are not as adequate to problems involving reliability. In that sense, increasing the number of destination sinks also increases the reliability, since the packet is duplicated and sent to different sinks.

We can summarize the challenges in three groups: the design of a routing protocol considering the existence of multiple sinks; the implementation of a solution strategies considering the reliability problem; and the optimization of the communication in terms of data delivery time and network lifetime.

Objectives and Contributions

The main objective of our work is to find ways of optimizing the communication in multi-sink wireless sensor networks. We address three main issues of the WSN: longevity (network lifetime), reliability (packet delivery) and timeliness (latency). We investigate the trades among data delivery time and energy consumption as key metrics for communication quality and efficiency. We also focus our efforts on studying the specificity connected to the existence of multiple sinks in a wireless sensor network. For that matter, we propose three strategies for packet routing considering different communication schemes (unicast, anycast and multicast). The routing decisions take into account the particularities of each communication scheme in order to optimize the network.

Our contributions include three different solutions, considering each of the communication schemes and optimizations on data delivery time and energy consumption. We detail the evaluation of the approaches using both simulation and experimentation. Furthermore, we propose a classification criteria for the solutions on multiple sink wireless sensor networks based on the communication scheme and the solution type. We present the characteristics and particularities of MS-WSN solutions, along with a discussion on some research opportunities in the field of multiple sinks WSN.

The first solution is a unicast approach, focused on reducing the packet latency by means of a MAC-aware strategy that considers the transmission delay as a decision metric for the routing path. In this solution, we work on the aspect of the packet forwarding to specific sinks and the reuse of routes for future transmissions as an attempt to predict the final latency and to provide packet delivery under a predefined deadline. Due to the need of timeliness communication, congestion and buffer size are also considered important metrics, leading us to the definition of a buffer control strategy.

For the second approach we proposed a k-anycast protocol. The k-anycast class of solution is extremely flexible. It may behavior as 1-anycast when $k = 1$ or even as a multicast approach when $k = |S|$, with S representing the set of all sinks. In our solution, we are focused on increasing the network reliability by means of forwarding the packets to k sinks. We also focus on reducing the network latency and increasing the network lifetime. Our strategy considers the aggregation of network metrics as a decision process for the next hop. It is a distributed solution that requires low network knowledge and that assures the packet forwarding to exactly k different sinks.

In the third solution we proposed a multicast protocol that is as well capable of reducing the network latency and increase the network lifetime. The objectives were at one hand to provide a solution capable of delivering packets to all sinks as a way to increase the reliability and the information availability, and at the other hand to verify the particularities of the multicast communication scheme as a counterpoint to the k-anycast with $k = |S|$. We show that it is possible to take advantage of the knowledge that packets must be delivered to all sinks to improve the routing decision and achieve better latency and network lifetime.

All solutions were validated with simulations and compared to other existing solutions. Our unicast solution was validated using a homemade simulator in SciLab [5]. Both k-anycast and multicast solutions were implemented on top of the Contiki operating system [6] and validated through simulations using the COOJA network simulator. The simulation computations were performed on the supercomputer facilities of the Mésocentre de calcul de Franche-Comté. Additionally, we provide real-life results for our multicast solution and

another existing protocol. Both solutions were tested on the FIT IoT-LAB testbed [7]. We demonstrate not only the behavior of our solution in real conditions, but also its correlation to the simulation results.

THESIS PLAN

This document is divided in three parts. The first part is dedicated to the definition of Wireless Sensor Networks composed of both single sink and multiple sinks. The second part is dedicated to the state of the art in MS-WSN. In the third part we present our solutions and the final considerations in the conclusion.

The Chapters composing the first part are:

- **Chapter 1 Wireless Sensor Networks** - in this chapter we introduce the basic concepts of wireless sensor networks.
- **Chapter 2: Multi-Sink Wireless Sensor Networks** - here we present the particularities related to the existence of multiples sinks in a WSN.

The Chapters composing the second part are:

- **Chapter 3: Routing solutions in MS-WSN** - in this chapter we present a classification for the routing solutions designed for multi-sink WSN, along with a brief description of each solution.
- **Chapter 4: Other solutions in MS-WSN** - here we enumerate and describe the existing solutions considering multi-sink WSN at the MAC and application level.
- **Chapter 5: System Validation** - in this chapter we survey the validation methods applied on the described solutions.
- **Chapter 6: Conclusion** - here we present a summary of the second part and some research opportunities in MS-WSN.

The Chapters composing the contribution part are:

- **Chapter 7: Unicast** - here we describe our first solution, dealing with the optimization of the data delivery time, in a cross-layer fashion and using the unicast communication scheme.
- **Chapter 8: Anycast** - in this chapter we present our second solution, focused on the reliability and the trade-off between data delivery time and network lifetime, using a k-anycast communication scheme.
- **Chapter 9: Multicast** - in this chapter we present our third solution, also focused on reliability and data delivery time versus network lifetime, but using a multicast communication scheme.
- **Chapter 10: Real-Life experiment** - here we present the details related to the validation of our multicast solution in a real-life environment.

We conclude this document with the conclusion chapter, where we present the final arguments, the perspectives and work opportunities.

The source code of the implemented approaches are available at:
https://gitlab.com/lucas.augusto/geo_multi-sink

I

MULTI-SINK WIRELESS SENSOR NETWORKS

1

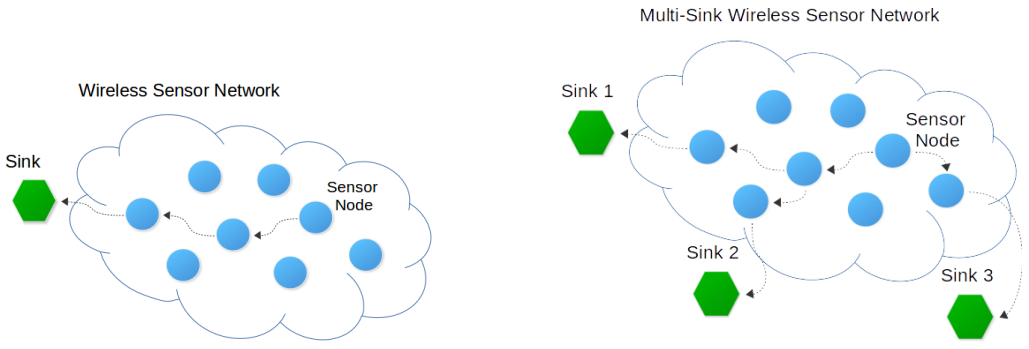
WIRELESS SENSOR NETWORKS

Contents

Wireless Sensor Network (WSN) rises rich research topics that have been deeply explored. It is the platform option for numerous applications, such as building management systems, healthcare monitoring systems, military and environment surveillance, and an enabling technology for the Internet of Things. A wireless sensor network can be defined as a network composed of small electronic devices, with limited processing and energy capacity, capable of performing sensory tasks, and exchange messages through wireless medium in order to feed a central entity - the sink - with sensory data.

1.1/ NETWORK COMPOSITION

A WSN is, in general terms, composed of two distinct entities: the sink and the sensor node. The number of sinks and sensor nodes may vary according to the application, and there may be wireless sensor networks with several sinks and sensor nodes, as exemplified in Figure 1.1.



(a) Single sink (b) Multiple sinks

Figure 1.1: Single sink and Multi-sink Wireless Sensor Networks

The sink acts as a gateway to the sensor nodes, collecting and centralizing the monitored information. The sink may be connected to another network, for example an IP network with Internet access, and acts as an interface between the WSN and the external network, where the data is routed and processed. Communications between nodes is done via radio, with the sink receiving data packets from each sensor node. The number of deployed sinks varies according to the application objectives, network structure and system architecture [8]. Applications of Smart Cities to monitor electricity consumption in households, for example, may require that several sinks are deployed, covering a given region, and centralizing information for packaging and shipping to a processing plant through an external network. Generally, it is expected from the sinks a better processing capacity and greater autonomy compared to the other sensor nodes.

A WSN may be composed of hundreds of sensor nodes, working collaboratively to monitor one or more characteristics of the environment. The sensor nodes are, in general, limited devices, with strong restrictions regarding communication range, memory and processing. Sensor nodes can communicate with each other or only with the sink depending on the application and the network structure. In this way, it is necessary to establish the rules controlling the forwarding process, and the routing protocol is responsible for that task. With the proper routing protocol, the data packets can be correctly forwarded through the network towards the destination [9].

1.2/ TOPOLOGY STRUCTURE

According to [8], WSN can be classified based on the network architecture. The deployment may follow different topologies, such as:

- **Point-to-Point**: Sensor nodes communicate with the sink through a chain of sensors. The sensor node sends a packet to its neighbor node, and the packet is forwarded from node to node until the destination is reached, as shown in Figure 1.2. The advantage of this strategy is the simplicity of implementation, however, it is unreliable, since any failure in the path prevents the final target to be reached. The packet is forwarded in a multihop manner the sink node.

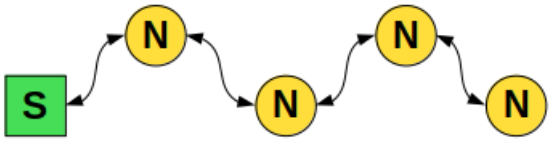


Figure 1.2: Point-to-point topology

- **Point-to-Multipoint**: Sensor nodes communicate directly with the sink. The sink is able to send packets to each of the sensor nodes, as shown in Figure 1.3. However, the sensor nodes can not communicate directly with the other nodes of the network. The advantage of this strategy is linked to energy savings, since there is no need for more sensor nodes to get involved in the routing chain. However, the size of the network is limited to the radio range of the sink and sensor nodes.

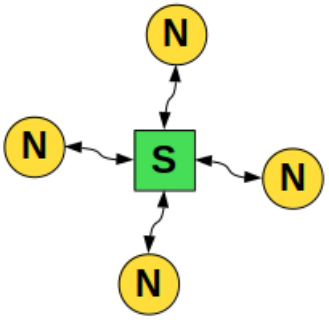


Figure 1.3: Point-to-multipoint topology

- **Mesh network**: The sensor nodes are able to communicate with the sink or any other sensor node in the network directly as long as they are within the radio range, as seen in Figure 1.4. This type of network presents a better fault tolerance, since the network generally remains functional even after a sensor node failure. However, the implementation becomes more complex and the challenge for energy savings is even greater.

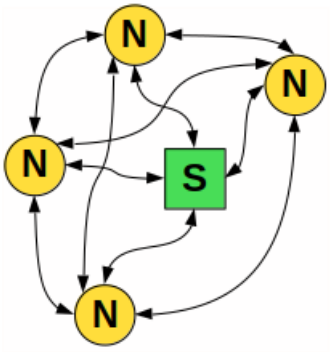


Figure 1.4: Mesh topology

- **Hybrid network**: The sensor nodes communicate with the sink directly or via another sensor node. In this configuration, we may have different routing structures, such as clusters and completely ad-hoc (graph-based). In clustering, a sensor node acts as a group leader, taking the responsibility of routing packets between several sensor nodes and the sink. In hierarchical, sensor nodes may follow a tree-based structure, with a node acting as a root and other nodes organized in the form of parent nodes and child nodes, as shown in Figure 1.5. In ad-hoc, the sensor nodes are vertices of a graph, with communication links being represented by edges between vertices. The hybrid network topology allows the combination of Point-to-multipoint and Mesh network, increasing fault tolerance, reducing energy consumption and enabling the implementation of high density networks. However, the routing protocol becomes more complex.

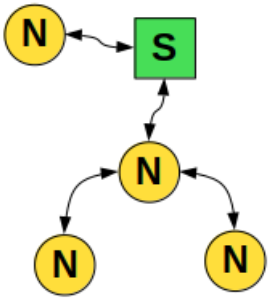


Figure 1.5: Hybrid topology

1.3/ LAYERED MODEL

WSN are typically structured in a five layer stack: Application, Transport, Network, Data Link and Physical [10], as shown in Figure 1.6. In the layered model, the information is processed at one layer and moved to the adjacent layer for further processing, following a strict communication interface. Each layer has a specific responsibility and acts on a particular metric within the scope of this responsibility. This way, the layered model provides modularity and flexibility, allowing different couplings of protocols for each layer.

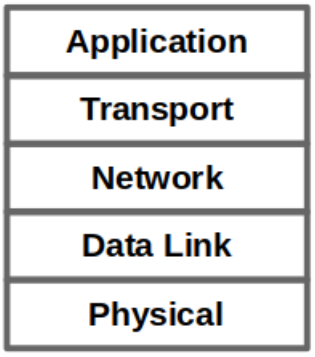


Figure 1.6: Layered model

In this structure, each layer plays a specific role:

- **Application Layer**: responsible for the monitored information, providing services according to the type of application;
- **Transport Layer**: responsible for controlling the communication flow;
- **Network Layer**: responsible for the packet forwarding, with the addressing and path definition;
- **Data Link Layer**: responsible for controlling the medium access and the energy management (duty cycle);
- **Physical Layer**: responsible for the radio signal modulation, channel and transmission power.

According to [11] strategies to optimize the communication in wireless sensor networks that follow the traditional layered model may achieve their purposes when strictly analyzed by the metrics of each individual layer. This means that although a particular problem is well addressed in a specific layer, the overall network performance does not benefit from a global optimization. For that reason, it is suggested that wireless sensor networks should be designed in a cross-layer fashion, allowing significant improvements in different network metrics. However, it is important to ensure that the gains with the cross-layer design pay off the loss of modularity inherent from this type of strategy.

Within a cross-layer strategy, layers share information in an integrated way, allowing decisions from one layer to be influenced by information from another layer. Cross-layer approaches may be divided in two major groups: inter-layer and intra-merged-layers. In the first, the well-known layered composition is maintained, but interactions within adjacent and non-adjacent layers are developed so protocol decisions at one layer are based on the decisions taken by the protocols from other layers in a back-and-forward way. The second category stands for a different approach. All layers or a group of layers are combined in order to create a single super layer able to respond for the functions of the merged layers.

- **Inter-layer**: The decisions in one layer are based on the information extracted from another layer (Figure 1.7). As for instance, a MAC solution on the data link layer may use the information of the routing tree from the network layer in order to optimize the slot scheduling and reduce the latency [12, 13, 14, 15, 16, 17]. That would be a MAC protocol based on routing. The same way, the routing protocol would use the information of the slot scheduling shared by the MAC in order to build an optimized routing tree that also reduces the network latency [18, 19]. This approach can be described as routing based on MAC. And in a third option, both routing and MAC could interact in an enchained way in order to achieve better performance in terms of routing and scheduling [20, 21, 22], finally leading to a routing and MAC back-and-forth cross-layer solution.

Figure 1.7: Inter-layer representation

- **Intra-merged-layers**: Merging of adjacent layers. A common (or super) layer taking simultaneous decisions, as for example, MAC and routing decision are done at the same time (Figure 1.8). A solution may be designed in a way that the route is created at the same time as the slot allocation takes place, for TDMA-based solutions. Instead of basing the allocation of slots on a predefined route, or create the routes based on an existing slot schedule, the decisions of next hop and slot allocation are taken simultaneously.

Figure 1.8: Intra-layer representation

1.4/ APPLICATION TYPE

According to [9], WSN applications can be classified into two categories: tracking and monitoring.

- **Tracking**: this type of solutions are related to applications which purpose is to identify objects, animals or people in certain locations. For example, a solution for tracking objects in a house can use sensor nodes scattered in all rooms to detect objects. Objects must be properly identified with RFID tags so that they can be located. The application issues the command for object tracking, and the sensor nodes next to the tagged object identify it, informing the central application of the location where the RFID label of the object was detected.
- **Monitoring**: they are characterized by the monitoring of phenomena, characteristics or events. Monitoring applications can be related to the monitoring of natural events, such as rainfall, seismic and volcanic activities, as well as characteristics of closed environments, such as temperature, ventilation, luminosity, smoke detection, energy consumption, etc. Monitoring applications can also be classified in terms of information criticality and time sensitiveness. Real-time applications require robust monitoring solutions with fault tolerance and redundancy. Differently, low criticality data monitoring solutions may not require fault tolerance mechanisms, since the loss of a data does not compromise the overall functioning of the system.

2

MULTI-SINK WIRELESS SENSOR NETWORKS

Contents

The Multi-Sink Wireless Sensor Network (MS-WSN) is a particular case of the WSN containing multiple sinks. Compared to single sink solutions, the use of several sinks normally leads to better network performance. On top of that, it also improves the network manageability, providing more flexibility and continuity [23]. By increasing the number of sinks, the number of hops a packet has to travel before reaching any of the sinks should normally decrease. It has a direct impact on the performance of metrics such as network lifetime, energy balance, latency, congestion, etc.

Solutions considering multiple sinks are designed over a number of different criteria. The decisions taken at the implementation level must account for the objectives to be achieved and the limitations imposed by the chosen solution strategy. Thus, it is relevant to point out the particularities associated to the existence of multiple sinks in WSN.

2.1/ PROCESSING TYPE

In general, WSN applications and protocols may work in a centralized or distributed way. A centralized algorithm for the routing protocol may find a global optimum solution faster than a distributed one. However, the level of knowledge on the network topology is far more elevated for centralized approaches than for distributed solutions, and in MS-WSN the coordination among sinks may increase the complexity in both cases (centralized and distributed processing).

- **Centralized:** in centralized solutions the decisions are taken in a central entity based on the fact that the central entity has a higher knowledge of the network state in comparison to other network entities. In the case of WSN, the sink is normally the central entity. However, in MS-WSN, since there are more than one sink,

the decisions in a centralized solution may be taken in three different ways: a) at each sink with the information being shared among them. It is still considered a centralized solution because the network information is centered at the sinks, and they coordinate together as a single entity; b) at a central processing entity which gathers the information from each sink through a wired connection or another reliable link; b) at each sink independently, since each sink is considered to be the central entity of a sub-WSN.

- **Distributed:** in distributed solutions the decisions are taken at the sensor nodes in two different ways. Either it is independently, when the sensor nodes dispose of enough information to take decisions, or collaboratively, when they are able to gather the necessary data to support the decision process. As for instance, in geographic routing, the sensor nodes may possess information of their geographic location and that from their 1-hop neighbors and sinks; this way the sensor node may decide to forward the packet to the neighbor node that offers the best progress in terms of distance to the closest sink.

- **Hybrid:** in hybrid solutions the decision process may happen in many different ways. As for example, at first a non-optimal routing may be created in a distributed way and once the central entity gathers the necessary information, an optimal route is created. Another option is to create an initial route in a centralized way, with the routes maintenance happening in a distributed way, with nodes exchanging data and deciding for new forwarders to reach the sink. The applied technique depends on the application demand and on the final objective. The common aspect is that the decision process is neither entirely taken centrally nor sparsely.

2.2/ COMMUNICATION SCHEMES

The final application drives the way communication takes place. The information is retrieved following different models in order to cope with the application objectives and requirements. For instance, in order to meet a specific requirement such as reliability, the communication can follow a many-to-many scheme, meaning that sensor nodes send their data to a number of destinations, which can be a set of sinks or all sinks in the network. At the application perspective, the important aspect is the number of destinations and if the destination target is predefined or not.

The routing protocol must obey the requirements imposed by the application. It means that if the application requires a many-to-many communication type, the routing protocol must be capable of forwarding the packet to a number of sinks in the network. However, the communication scheme may be described in a different way at the routing level. The problem of reaching all sinks in the network may be solved by multiple implementation strategies considering different communication schemes, such as unicast and multicast communication schemes. At the routing perspective, the forwarding mechanism is the most important aspect.

Therefore, we identify a twofold view of the communication schemes: at the application level and at the routing level.

2.2.1/ APPLICATION PERSPECTIVE

Many-to-one

- **Predefined target**: one particular sink must be reached depending on the information type. An application monitoring three different parameters can be implemented in a way that each parameter must be sent to a specific sink. The nodes share the same environment and responsibilities (sensing and forwarding data), but information is forwarded to different destinations according to the application's objective. That may be the case of a Building Management System, with sensors monitoring luminosity and temperature. The information about the environment lightening may be relevant only to a specific sink, which will also manage the actuators that will increase or decrease the lightening.

- **Undefined target**: one of the sinks must be reached, but no specific sink is addressed. A system seeking better performance in both latency and energy consumption may require any sink to be able to receive the data, leaving the destination decision to the routing strategy. This way the network is able to achieve longer lifetime and reasonable latency. An example is a fire detection system in a forest. The network must be active for as long as possible, but once an event is detected, the data must be delivered in a reasonable time, to any of the sinks available.

Many-to-many

- **Predefined target**: a number of specific sinks in the network must be reached. The information is forwarded and replicated until it reaches all predefined sinks. Generally, it is related to specific application demands, such as reliability and data availability. This is particularly interesting for a system requiring the sensed information to be available at all sinks, and where sinks are not interconnected through an external network. A system tracking buses in a city may require the information of the actual position of the bus to be available on every bus stop, so users will be informed about the waiting time.

- **Undefined target**: a group of sinks must be reached. The information must be replicated and forwarded to a number of sinks. But the final destinations are not defined. The need to reach multiple sinks may be connected to a reliability requirement. As an example, in a Healthcare Monitoring System, a sensor network monitoring patients may forward the sensed data to one of the sinks in a normal situation, but in case of an emergency, the information may be forwarded to several sinks in order to make sure the event will be acknowledged.

2.2.2/ ROUTING PERSPECTIVE

- **Unicast**: the information is addressed to one sink and it is not changed during the forwarding process. The sink selection may be predefined by the application or decided based on the routing structure. Figure 2.1 illustrates the composition.

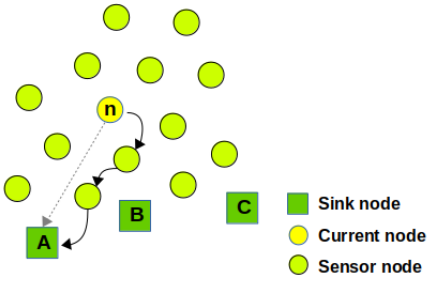


Figure 2.1: Information from source **n** is addressed to sink **A**

- **Anycast**: information is addressed to any of the sinks, so the destination may change during the forwarding process. It can be 1-anycast, when information is forwarded to any sink in the group, or k-anycast when information must reach any k sinks, as illustrated in Figure 2.2. The anycast communication scheme may be applied to either increase network reliability (k sinks) or extend the network lifetime (any one sink).

(a) 1-anycast: any sink must be reached (b) k-anycast: $k = 2$ sinks must be reached

Figure 2.2: Information from source **n** is addressed to any of the sinks **A**, **B** and **C**

- **Multicast**: information is addressed to a number of sinks. It may be all the sinks in the network or a number of predefined sinks. The destination sinks are defined at the source node, and the destinations are not changed during the forwarding process. The information of a single sensor node is replicated and forwarded to each one of the predefined sinks, as shown in Figure 2.3. Multicast approaches are focused on network reliability and information availability.

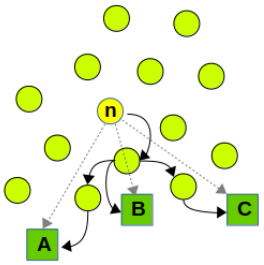


Figure 2.3: Information from source **n** is addressed to all sinks **A**, **B**, or **C**

The communication strategy at the application perspective can be implemented using different strategies of the communication scheme at the routing perspective. Table 2.1

illustrates how each of the communication strategies of the application perspective can be designed at the routing perspective.

Table 2.1: Application vs Routing implementation strategies

Application perspective		Routing perspec.	Implementation strategy
Comm. scheme	Target		
Many-to-one	Predefined	Unicast	The packet is forwarded to one specific sink. The destination is defined at source.
	Undefined	Unicast	The packet is forwarded to one specific sink. The destination is defined at source as part of the routing strategy (disjoint spanning trees rooted at each sink).
		1-Anycast	The packet is forwarded to any of the sinks. The final destination is decided during the forwarding process.
Many-to-many	Predefined	Unicast	The packet is replicated at source and each packet is forwarded to a different sink. The destinations are defined at source.
		Multicast	The packet is forwarded to a group of specific sinks. The destinations are defined at source. The duplication of the packet takes place during the forwarding process.
		k-Anycast	We consider S the set of all sinks. It only works if the k value represents the set of all predefined sinks (S'), which means that $k = \|S'\|$ and $S' \subseteq S$. In this case, S' is the set of destination sinks. Otherwise, it is not possible to assure that a specific sink receives the packet. The packet duplication takes place during the forwarding process.
	Undefined	Unicast	The destinations are defined at source as part of the routing strategy (spanning trees rooted at each sensor node with the sinks as leaf nodes). The packet is replicated at source and forwarded to one specific sink.
		1-Anycast	The packet is replicated at source and each packet is forwarded to any of the sinks. The final destination is decided during the forwarding process. It is difficult to assure that different sinks receive the packet.
		k-Anycast	The packet is forwarded to a group of sinks. The final destinations are decided during the forwarding process, along with the duplication of the packet. In this case, it is possible to assure that k different sinks receive the packet.

2.3/ IMPLEMENTATION OBJECTIVES

Most of the time the performance objectives of a WSN are driven by the application demands. The solutions are designed in a way to accomplish the imposed requirements. The performance objectives of a WSN may include different targets varying from energy optimization to real-time communications. However, the simple design of a WSN with several sinks is not enough to achieve the desired performance. For that reason, solutions in MS-WSN must be tailored using different metrics in order to reach the needed objective.

- **Longevity**: the main goal is to prolong the network lifetime. The lifetime of a network can be described in different ways, such as the first node to die, the last node to die and the moment when the network becomes disconnected. It depends on the requirements defined by the application. In all cases, many factors may influence the network lifetime, such as the occurrence of collisions, congestion and re-transmissions, the unbalanced traffic and energy consumption, the paths with excessive amount of hops, etc. Since WSN are normally composed of devices with limited energy capacity, the longevity of the network is crucial. According to [24], in MS-WSN the distance traveled by a packet is reduced since the possibility of a node to be close to a sink is higher, consequently reducing the overall consumed energy. However, specific techniques must be applied to organize the packet transmission and routes, as well as to define the sink positioning in order to profit from the existence of multiple sinks.
- **Reliability**: data availability and network resilience are the two main aspects of the reliability. The focus is to assure that data will be available for the application when necessary. For that purpose, other metrics are considered during the solution implementation, such as redundancy of nodes and/or paths, the packet delivery rate, collisions, congestion, energy consumption etc. The reliability class of solutions covers approaches centered on techniques such as fault tolerance, multi-path, packet duplication etc.
- **Timeliness**: this main objective is related to the techniques for communication optimization seeking the minimization of the end-to-end delay. Applications with specific deadlines for message delivery require the solutions to take into account metrics such as queue delay, transmission delay, latency, collisions, congestion, hop count and other metrics that have influence on the delivery time. Normally, it is the main objective of low latency applications, but many solutions consider the metrics related to timeliness as secondary optimization objectives.

II

STATE OF THE ART ON MULTI-SINK WIRELESS SENSOR NETWORKS SOLUTIONS

3

ROUTING SOLUTIONS IN MS-WSN

Contents

In general, a routing solution is an implementation of a set of responsibilities connected to the network layer in the OSI model. The main task of a routing solution is that of the discovery of routes towards a destination. In Multi-Sink WSN this task must account to the fact that multiple sinks are available, and therefore multiple routes should be considered. Moreover, depending on the application's objective, the communication scheme is such that the routing solution must find a path to one sink (unicast), to any of the sinks (1-anycast), to a set of any of the sinks (k-anycast), or even to all target sinks (multicast). The routing strategy becomes more complicated when compared to single sink solutions, and many particularities arise, such as the necessity of duplicating packets (for k-anycast and multicast).

Our classification criteria for Multi-Sink WSN routing approaches are presented in Section 3.1 followed by a synthesis of each solution in Section 3.2. In Section 3.3 we present a brief conclusion with the main characteristics of the routing solutions in MS-WSN.

3.1/ CLASSIFICATION OF ROUTING SOLUTIONS

We focus the classification of the MS-WSN routing solutions on the multi-sink aspect. The objective is to present the particularities raised by the existence of multiple sinks. The solutions are classified through five different categories: communication scheme, path strategy, sink decision, route structure and packet duplication, as presented in Table 3.1.

The routing perspective of the *Communication Scheme* (already detailed in Subsection 2.2.2) focuses on bringing evidence to how the multi-sink aspect was considered in the solutions in terms of packet destination.

Comm. Scheme	*Path Strategy*	*Sink Decision*	*Route Structure*	*Packet Duplic.*	*References*
Unicast	Single path	Fixed	Tree	No	[25][26][27][28][29] [30][31][32][33][34]
Unicast	Multi-path	At source	Tree	No	[35][36][37][38]
Unicast	Multi-path	At source	Graph	No	[39][40][41][42]
Unicast	Multi-path	At source	Tree/Graph	No	[43][44]
1-Anycast	Multi-path	On the way	Graph	No	[45][46][47][48][49] [50][51][52][53]
k-Anycast	Multi-path	On the way	Tree	Source	[54]
k-Anycast	Multi-path	On the way	Graph	Source	[55]
k-Anycast	Multi-path	On the way	Graph	On the way	[56]
Multicast	Multi-path	Fixed	Tree/Graph	On the way	[57][58]

Table 3.1: Summary of routing solutions in MS-WSN

The *Path Strategy* indicates the robustness of the solution in terms of connectivity. It is more likely for a multi-path solution to sustain the network connectivity for a longer time than a solution that considers a single path strategy. Although the concept of multiple paths and single path is also found in single sink WSN, the definition of multi-path and single path changes slightly in multi-sink WSN:

- **Single path**: the node is connected to a single sink through a single path. Despite the fact that the network is enabled with multiple sinks, nodes are part of a single disjoint tree rooted at one sink.
- **Multi-path**: there are essentially two different cases. The first is when the node is connected to multiple sinks through different trees or routes. It is considered multi-path because the packet can be forwarded to a sink through more than one path; in this case each path leads to a different sink. The second case is when the packet can be forwarded through multiple paths to the same sink. We may find both cases in the same solution when the communication scheme is anycast.

The *Sink Decision* identifies the moment the destination sink is chosen. Since in Multi-Sink WSN there is more than one sink able to receive the packet, a decision of which sink will effectively receive the packet needs to be taken. It depends on the communication scheme and also on the path strategy.

- **Fixed**: the destination is fixed and predefined. The destination may be imposed by the routing structure, as for the case of disjoint trees rooted at each sink, or an application requirement. The packet must be delivered to the predefined sink, and there is no change in the destination during the forwarding.
- **At source**: the sinks are decided at the source node, before the first hop. The source node is connected to more than one sink and according to specific metrics, the choice for a sink is made and respected through the entire forwarding task.
- **On the way**: the sink decision is taken on the way instead of at the source node. The packet is forwarded to a neighbor with the best progress rate according to the

used metrics. This way, any of the sinks may be reached, depending on the network state in terms of the used metrics.

The *Route Structure* intends to show how the paths are organized. Similarly to single sink WSN, we may have tree or graph structures. However, there are some particularities to these structures when applied to Multi-Sink WSN.

- **Tree**: there are three different cases of tree structures in MS-WSN. The first is the case of disjoint spanning trees rooted at each sink and having sensor nodes composing the structure in an exclusive way. It means that a node is part of only one tree. The second case considers almost the same scenario as the first case, with multiple trees rooted at each sink, but with all sensor nodes being part of all trees. Finally, the third case considers trees rooted at the source nodes with each of the sinks as leaf nodes.
- **Graph**: with the graph structure it is considered that the network paths can assume any form. For instance, it can be a DAG (Directed Acyclic Graph), a graph based on geographic distance to any sink, or even a graph based on the nodes schedule/availability at a given moment. The graph structure is the most used strategy for Multi-Sink WSN running with anycast, since it gives much more options in terms of paths to reach any of the sinks.
- **Tree/Graph**: it considers a tree structure that at a certain level can also have graph characteristics. For example, a tree with a graph structure at the level of the leaf nodes, or among the sink neighbors. Also at the borders of different trees so that they can be inter-connected by a bridge node. We consider the DODAG (Destination-Oriented Directed Acyclic Graph) to be a Tree/Graph structure.

The *Packet Duplication* is an underlying characteristic of MS-WSN running with k-anycast ($k > 1$) and multicast. Since the data must be forwarded to k sinks for the case of k-anycast or to all sinks for the case of multicast, the packet must be duplicated at a given moment. The decision of when the packet is duplicated is highly connected to the sink decision, but not entirely determined by it.

- **At source**: the packet is duplicated at source. If k sinks must be reached, exactly k transmissions of the same packet will be made. There are two possible addressing options. Either the packet takes the exact address of the sink to be reached or a set of potential addresses. The set of addresses are disjoint from one packet to another. For instance, if some data must be delivered to k sinks, with $k = 2$, and the network is covered by 4 sinks, the first packet may have the destination sink set $\{1, 2\}$ and the second packet the set $\{3, 4\}$. So, the first packet may be delivered to any sink in the set $\{1, 2\}$ and the second packet to any sink in the set $\{3, 4\}$. With this strategy, the algorithm works as k-anycast during the duplication, and as 1-anycast during the forwarding process after the set of sinks is defined.
- **On the way**: the packet is duplicated during the forwarding task. The duplication moment happens at the same time as the occurrence of a fork in the forwarding path. The need of a fork in the forwarding path is caused by the existence of disjoint paths towards the addressed sinks in relation to the current node. Initially,

the packet travels without duplication through a single path. Then, because of the disjoint paths, the packet must be duplicated and sent towards different directions. This is the case of a hard duplication, when two or more packets are created and forwarded. However, another possible case is when the packet is not effectively duplicated, but sequentially forwarded. The packet has a number of sinks to be reached and it decrements this value each time it encounters a sink.

3.2/ ROUTING SOLUTIONS DESCRIPTION

In this section we present a brief description of the routing approaches considering multiple sinks. We divide the presentation of the solution based on their characteristics, mainly in regard to the applied technique in terms of communication scheme.

3.2.1/ UNICAST

We present the unicast solutions divided in two groups: multi-path and single-path. As already described in Subsection 2.2.2, in unicast communication scheme the packet is forwarded to one single sink and the destination does not change during the forwarding.

3.2.1.1/ SINGLE-PATH

In the class of solutions categorized in the single-path group, there is only one path exploited towards the destination sink. The routing structure is formed by disjoint trees, and each node participates in only one tree.

Solution [25] defines a data collection scheme that satisfies a specific end-to-end delay constraint, where nodes send collected data to a single sink with the lowest latency in a multiple-sink network. Using a heuristic algorithm and in a centralized way, the latency is evaluated for each node towards each sink and the path with the lowest latency is selected. Load balance is also considered in latency estimation in order to avoid excessive collisions, which consequently increases the latency. Disjoint trees are constructed with the sinks as roots. The nodes join the trees in a greedy way, checking the hop-count of the neighbor nodes to the closest sink. After that, the information is evaluated by the centralized algorithm and the tree is changed in order to satisfy the desired latency constraint.

In [26] the objective is to reduce the energy consumption, to balance the sink flows and to prolong the network lifetime, by means of pre-clustering and clustering techniques, considering both energy and location information in a dynamic metric that changes over time. The algorithm creates different regions in the network in order to create a pre-cluster. The sinks broadcast messages with different power ranges creating different circles representing areas of coverage. Then, within each ring, the clustering process takes place considering the nodes distance to the center of the ring, the distance to the sink, the average energy of the nodes in the cluster and the residual energy. The packets are forwarded to the cluster head that later forwards them to the cluster head of the next ring. The path to the sink is selected in a way that minimizes the relative distance to the sink. The cluster heads are changed after the execution of each round, in order to better

distribute the energy consumption. The sinks calculate the distance and angle of each cluster area and broadcast the information to all nodes, this way the nodes know to which cluster they belong. The cluster heads are chosen based on the decision metrics, and they vary according to the network stage. If it is the first round, only the distance from the node to the cluster center and the distance from the node to the sink are considered. For the other rounds, the average energy of nodes within the cluster and the residual energy are as well considered. In addition, the metrics take different weights in the decision process. The weights are dynamically regulated during the network lifetime, in order to give more importance to the energy factor in the late cycles of the process. Moreover, the relative position of clusters are adjusted every round in order to avoid nodes far away from the cluster center to deplete energy faster.

In [27] the authors define a data collection scheme with minimum latency. The routes are based on the transmission schedule. Two strategies are presented: 'LP-based' builds a time-expanded graph showing all possible links at each time and from there defining the schedule. And 'greedy BFS-based' that creates layers of sensors based on the hop-count to sinks and scheduling the transmission based on the proximity to the sink, also considering the interference of neighbor nodes.

In [28] the objective is to balance the network load, avoid the early depletion of the neighbor nodes of the sinks and increase the network lifetime. Using a mobile anchor to delimit zones in the network, the algorithm dynamically changes the size of the zones based on the network load and residual energy of the hotspot sensor nodes. The routing is defined within a region by considering the PSF function (Path Selection Factor) which evaluates the proportion between the residual energy and the node distance to the hotspot neighbor.

In [29] the objective is to define a communication protocol able to adjust the network load by distributing the nodes within different spanning trees. It creates initial spanning trees based on the shortest path and other routing metrics in case of multiple options. It evaluates the load distribution for each sink and determines the need of a balancing phase, while advertising the ideal load in each cluster. The nodes decide to switch from one cluster to another in a distributed way, based on the switching threshold.

In [30] the objective is to extend network lifetime as well as to assure the feasibility of the routing proposal. The strategy is based on the definition of a MAC-aware data aggregation strategy which directly affects the routing feasibility (path is selected among nodes that fulfill the conditions). The routing is defined by an Integer Linear Programming (ILP) solution based on flow balance, energy consumption and medium availability to forward packets to a specific sink.

In [31] the objective is to improve the energy-consumption in order to increase the network lifetime in WSN with secondary sink nodes. It creates a hierarchy based on the remaining energy. Level-1 nodes must receive and forward packets, and Level-2 nodes only forward packets. Nodes must keep the information from neighbor sensors. The routing protocol is similar to the LEACH [59] way of working, since it defines a hierarchy to organize the network. The hierarchy is constructed based on the residual energy. The packets are forwarded to higher levels of the hierarchy until reaching the sink. The secondary sink to be reached is defined by the tree structure, based on the residual energy of the nodes. Once the packet is received by one of the secondary sink nodes, it is again forwarded to the primary sink. With this strategy, the multiple sinks are organized as secondary sinks with the primary sink as the root.

In [32] the objective is to define a binding strategy based on a genetic algorithm where nodes select the optimal sink to forward the data, considering the hop distance to the sinks. The central processing entity gathers all the network information and starts a process of binding nodes to the optimal sink, respecting rules defined at the genetic algorithm level. The nodes are separated into different categories and the binding process happens respecting the order defined with the categories. The algorithm starts with an initial phase of network discovery. Then, a second phase takes place in a central station, where the nodes are bound to a sink. This process is iterative, starting with nodes 1-hop away from the sink (first exclusive nodes, then non exclusive nodes). Later, the other nodes are bound to a sink based on their neighbors already connected to a sink. At the last phase, the algorithm informs the other nodes of the defined tree.

In [33] the objective is to maximize the network lifetime in a multi-source multi-sink WSN. The basic strategy is to identify the path that maximizes the network lifetime by selecting the pair (source, node) which has the greatest lifetime among the ones with the lowest lifetime. The authors present two solutions, one centralized (LOCL) and another distributed (LONL). Both algorithms work as a stepwise process that builds trees rooted at each sink, searching for the configuration that maximizes the network lifetime.

In [34] the objective is to balance the network load by creating multiple spanning trees, distributing the traffic evenly within the branches, and to minimize the total latency by reducing the maximum hop count. The process of building the trees is centralized and considers the amount of traffic generated by nodes as well as the amount of neighbors. Once the initial trees are constructed, the algorithm reevaluates each branch and executes inter-branch adjustments to better distribute the traffic and then an intra-branch adjustment, to minimize the latency by reducing the maximum hop count. The spanning multi-tree construction starts by identifying the sink neighbors, called sink-adjacent nodes. The sink-adjacent nodes are inserted in the expandable branch list, if they have unattached nodes in their vicinity. Then, each branch is expanded by attaching a neighbor node to it from the list of unattached nodes.

3.2.1.2/ MULTI-PATH

For the solutions categorized in the multi-path group, the packet is forwarded through different paths, but the destination sink remains the same. We divide the solutions into tree-based, graph-based and tree/graph-based approaches.

Tree-based: in tree-based approaches, the route structure is composed of spanning trees rooted at each sink, with sensor nodes participating in multiple trees. Once a destination sink is defined, the packet is forwarded upwards in the tree rooted at the chosen sink.

In [35] the objective is to define a collection algorithm focused on improving the network lifetime and reliability, by means of proposing multiple trees rooted at each sink as alternative routes in case of node failure or energy depletion, in order to extend the network lifetime and balance the energy consumption. The trees are built based on the link cost and the cumulative path cost. Hop count, residual energy and neighbor distance are the metrics considered for the link cost. The nodes join a tree in a greedy way based on the cumulative cost. The sink decision is taken at the source node, before the first hop, when the path is selected and followed up to the destination. There is a periodic update of the trees, in order to keep the path cost updated in respect to the residual energy.

The occurrence of failures may also trigger a new tree reconstruction. The first phase of the routing takes place with the sinks flooding the network with the Sink Route Request control packet. This packet contains the information of the sink and the path cost. Each node has to participate in a tree to each of the sinks. Once all trees are defined, the source node selects the sink with the best path cost to forward the packets. The trees are periodically reevaluated, in order to keep the path cost up to date regarding the energy cost.

In [36] the objective is to define a routing algorithm based on ant colony optimization [60] in order to improve the network QoS, considering packet loss, re-transmissions and buffer size as decision metrics. The strategy makes use of the concept of pheromones to balance the attractiveness of routes. Each node has a path to each sink in the network in the form of multi-dimensional trees. The packets are forwarded hop-by-hop towards any of the sinks based on the decision metrics. The route establishment happens in a bottom-up fashion, with nodes sending path requests towards the sinks. The nodes receiving the request reply only if their hop count to the sink is smaller than the one of the requester.

In [37] the objective is to maximize the energy usage of the network by minimizing the impact of routing occurring repeatedly on the same set of nodes. The algorithm creates multiple trees rooted at each sink. The routing path is constructed in a distributed way, with each node deciding its next hop. However, the nodes only store the information of one neighbor per sink on the routing table. The selection of neighbors is based on the hop count and the estimated distance to the neighbor. First, nodes send hello messages in order to discover the neighborhood. Later, all sinks send the tree setup message. This message is forwarded by each node, until all nodes are covered by all sinks. The packets are forwarded to a specific sink, following the entries available in the routing table. The neighbor responsible to forward the packet is selected based on the hop count to the sink and the estimated distance.

In [38] the objective is to balance the energy consumption of the sensor nodes during the message routing in order to prolong the network lifetime, by means of transmission cost in terms of energy. Two algorithms are proposed with different strategies. For ELBR the algorithm considers the nodes' energy level, then it calculates the energy level for each path to all sinks and the path with the maximum energy level is chosen. For PBR, the algorithm considers both the energy level and the energy cost of each node, in order to balance the energy consumption. Then, for each path to each sink it calculates both the resulting energy level and energy cost and executes an evaluation with different weights for both values to compose a single metric. A routing table containing the path to each of the sinks is used as input. The algorithm then executes the process of choosing the best sink to forward the packet based on the decision metrics.

Graph-based: for the graph-based approaches, the route structure is represented by a graph with nodes capable of discovering multiple paths towards multiple sinks. Although the packets may follow different paths, the sink does not change during the forwarding.

In [39] the objective is to apply modifications to the original Directed Diffusion (DD) routing algorithm, in order to optimize the network lifetime in the case of multiple sinks. The proposal considers adapting the DD algorithm to achieve better performance in the case of multiple sinks. The strategy is to attach tags to the path to each sink and select a specific path to forward the packets, based on the hop count and energy level of the nodes in the path. First, the sinks broadcast a packet called "interest", later the source nodes receiving the packets, and having data to be delivered, broadcast a packet of

“exploratory data” (ED). The ED packet is forwarded by all neighbors until it reaches a sink. Then the sink sends a “positive reinforce” (PR) message through the inverse path made by the ED packet. Upon the receiving of the PR message, the source node labels the path and saves the information of the neighbor node which transmitted the message. Once having the path to all sinks, the algorithm selects the most efficient path based on the hop count and the energy level, and starts forwarding the data packet through that path. Once the energy level of the path reaches a threshold, the node starts considering the other saved paths for the forwarding task.

In [40] the objective is to define a routing algorithm based on the path loss from each node to all sinks and the residual energy from the 1-hop neighbors of the sinks, focusing on the optimization of the network lifetime. It is a reactive algorithm with routes being discovered upon the necessity of forwarding a packet. The first step is to define the path loss from each sink to all nodes. Then nodes start forwarding the packets to the sink with the minimum path loss and minimum reciprocal battery level for the direct neighbors. The sink address is defined at source, which prevents loops. The path is selected based on two criteria: the analyzed metrics and the node contention window. Each node receiving a RTS (Request To Send) from the source node evaluates the possibility of participating in the route, then defines a timer to respond the requester with a CTS (Clear To Send). The first node to send the CTS is selected as packet forwarder.

In [41] the objective is to balance the network load by distributing the loads among the sink neighbors and thus prolonging the network lifetime. The algorithm makes use of the round-robin scheduling process to distribute packets among all sinks neighbors (deputies), in order to avoid the early depletion of energy and collisions in the sink vicinity. Every node stores a table containing the next hop information to all sinks’ neighbors. The packets are forwarded through the next hop respecting the initial destination attribution and a forwarding metric. The metric is based on the neighbor energy level and distance to the deputy. Beacon messages are sent from sink to nodes in order to discover the network. The deputy nodes (sink neighbors) re-transmit the beacon and the other nodes in the network learn about the existence of all deputy nodes. The sensor nodes must store a table containing each existing deputy node, the hop level to that specific deputy node, the neighbor leading to that deputy node and the dirty level (error counter). In addition to that, another table must contain all the neighbor nodes, with their respective energy level and dirty level. Once these tables are filled, the node starts transmitting packets. A new packet is assigned to a deputy following the round-robin scheduling process. The node receiving a packet to a specific deputy selects within its table the best candidate for the next hop.

In [42] the objective is to increase the network lifetime and distribute the traffic from sensor nodes among the existing sinks. The algorithm makes use of the concept of potential fields to define a cost to a link. It works as a gradient-based routing algorithm, since it makes a combination of metrics to select the next hop. Nodes are attached to a single sink, based on the gradient value. The routes are created in a greedy way by checking the potential field values of the neighbor nodes. The next hop decision considers the residual energy of the neighbors and the hop-count to reach the sink.

Tree/Graph-based: the tree/graph-based approaches are adaptations of the RPL (Routing Protocol for Low-power and lossy networks) [61] to the scenario of multiple sinks. Since RPL works with DODAGs, the solutions based on the RPL protocol are categorized as tree/graph-based according to the classification presented in Section 3.1.

In [43] the objective is to adapt the existing RPL protocol to a multi-sink WSN in order to reduce the average hop count, and consequently the energy consumption and packet loss. It defines a virtual root, which acts as a ”master sink” unifying all sinks into a single DODAG (Destination Oriented Directed Acyclic Graph). Each real sink is part of a subtree of the virtual sink rooted tree. The routing strategy follows the RPL recommendations and it considers different constraints/requirements (objective function), such as the link quality and the hop count to the sink. The route is discovered by means of message exchange among neighbors and parent nodes.

In [44] the objective is to improve the packet delivery rate by proposing numerous objective functions (OFs) to help build the routes towards several sinks. The different OFs are based on cross-layer (MAC delay and queue) decisions to help the route choices in order to improve the packet delivery rate. Once again, the RPL recommendations are followed and the route construction process behaves as the standard RPL.

3.2.2/ 1-ANYCAST

In the 1-anycast communication scheme, the packet is forwarded to one sink, but the destination is not fixed. The final destination is determined along with the path taken by the packet. The path decision is based on the adopted routing strategies.

Opportunistic routing: the solutions considering opportunistic routing benefit from the broadcast nature of wireless links to forward the packet through different potential paths. Sensor nodes can overhear a packet transmission, and based on opportunistic decisions/rules forward the packet towards the next-hop [62].

In [45] the objective is to define an energy-efficient opportunistic routing protocol that takes advantage of the wireless medium characteristics to extend the network lifetime. The strategy considers defining a set of forwarder candidates, organized by priority and relaying the packet to those nodes, expecting that at least one of them will correctly receive the packet and forward the data. The priority is defined with the OEC metric, which is calculated from the sink to all nodes in a cumulative way. Nodes broadcast their existence in order to build a table of neighbors. Starting from the sink, a configuration message with the first OEC metric is sent to all sink neighbors. Then the sink neighbors calculate their own OEC and advertise it to their neighbors. This procedure is repeated, until all sensor nodes calculate their OEC. The OEC metric is based on the node’s residual energy and the energy cost to relay a packet through the optimal path. The sender then forwards the packet to the set of forwarder candidates and the node with the highest priority within the nodes that received the packet will relay the packet and send an ACK to the sender.

The objective in [46] is to define a routing protocol for rechargeable MS-WSN in order to reduce network latency and optimize the energy consumption. The strategy defines a duty cycle-aware routing protocol, that opportunistically chooses the next hop based on the latency to the best sink and the duty cycle. It means that the first node to wake up is the first node to receive/send the packet. The duty cycle is dynamic and varies according to node’s energy level. The node having a packet to be sent is able to wake up at any time if it has enough energy for the transmission. However, it has to wait for some neighbor node to wake up in order to send the packet.

Gradient-based routing: the solutions considering the gradient-based routing base the

path decisions on a calculated metric that represents the capability of a node to serve as next hop. The packet is forwarded in a greedy way to the node that presents the best value of the calculated metric.

In [47] the authors propose a distributed algorithm for multi-sink anycast communication scheme based on potential fields capable of ensuring low overhead and high resilience to the network. The algorithm is based on the concept of potential fields. The sinks attract the data packets to them through the intermediate nodes. The potential field is calculated in a distributed way and may consider different metrics according to the application objective. The algorithm starts with the exchange of message between nodes in order to achieve the potential field equilibrium. Once the potential fields are defined, the source nodes start transmitting the packets to the neighbor nodes.

Solution [48] defines a balanced routing path with the lowest possible latency and improving energy consumption. The algorithm evaluates the gradient field of a node considering three attributes: the hop distance, the buffer occupation of the 1-hop neighbors and the buffer occupation of the 2-hop neighbors, in order to select the best path. The strategy also considers selecting sibling nodes instead of parent nodes when congestion is detected. It is a 2-phase routing algorithm. The first phase is reserved to announcements, where nodes communicate their information to other nodes. The second phase is the routing definition, based on the path selection strategy. It makes use of weights to equalize the importance of the metrics on the final gradient value. The lowest gradient index is always selected. Packets can be forwarded to parents or siblings, but never to children.

Bio-inspired routing: in bio-inspired routing the packet forwarding problem is solved with the use of algorithms that represent a nature phenomena. As for instance, swarm intelligence is used to model the behavior of the packets being forwarded in the networks.

In [49] the objective is to create a routing strategy based on swarm intelligence capable of increasing the network scalability and fault tolerance, by means of network self organization and adaptation. The algorithm strategy considers taking advantage of the pheromone concept to organize the network and fast react to changes, such as node failure and the deployment of new nodes and sinks. The pheromone works as a gradient value, giving nodes a comparison parameter to choose, among the other neighbors, the one to effectively forward the packet. The network setup and maintenance work similarly. All sensor nodes and sinks transmit packets containing their pheromone levels in a variable interval. At the beginning, sensor nodes have a pheromone level of 0 and the sinks a level of 1. The nodes receiving the sink pheromone at level 1, calculate their own pheromone level and propagate it to their neighborhood. This process is followed indefinitely, since the pheromone dissemination is a constant event. In parallel, at each node there is also a process of decreasing the pheromone level of all nodes. The cadence of decreasing and advertising is dynamic and calculated based on the network state. The packets are forwarded towards the nodes with higher levels of pheromone, until the packet reaches any of the sinks.

Geographic routing: in geographic routing the forwarding process is based on the physical/virtual position of the nodes. Little network knowledge is required, usually only the positions of the one-hop neighbors and of the destination sink. The next hop is decided based on the progress towards the destination sinks. The selected neighbor node is the one with the largest geographic progress towards a sink in relation to the current node.

In [50] the objective is to balance the network load and energy consumption in order to prolong the network lifetime, by considering the next hop the node having the higher geo-

graphic progress, the higher residual energy and the lower delay. The algorithm considers a distributed solution, based on beacon message exchanges as a way to communicate to the neighbor nodes the defined metrics (location, residual energy and delay). Once having all metrics, the nodes calculate the aggregated weight for each neighbor and forwards the packet to the node that offers the best progress towards any of the sink, as well as having the highest residual energy and the lowest delay. By favoring the best geographic progress, the algorithm tries to achieve better energy consumption and latency, since fewer hops are involved in the forwarding task. The residual energy metric ensures a better distribution of the energy consumption. And the lower delay improves the traffic balance, since it distributes the traffic among nodes with small queuing time.

The work in [51] presents three solutions. The idea is to assure packet delivery while minimizing the path (GFGA) and energy consumption (COPA, EEGDA) in a MS-WSN. The strategy considers applying the anycast concept to geographic routing (GFGA), and associates energy consumption in the decision process (COPA, EEGDA). The next hop is decided based on the cost of the energy-weighted shortest path (ESP). At each hop, the algorithm calculates the cost over progress to the geographically closest sink and decides to which node the packet will be forwarded.

The objective in [52] is to reduce the packet loss due to node failure (network holes) and to increase the network lifetime. It makes use of the geographic forwarding technique with a dynamic duty cycle approach. It dynamically changes the nodes duty cycle to balance the energy consumption. Routes are decided based on the duty cycle. It means that the first node to wake up is the first node to send the packet. It works as a geographic forwarding algorithm during the first phase, when there is no fault. In case of fault, a flag field is activated and the packet may be forwarded to nodes other than the ones defined by the geographic forwarding algorithm.

Probabilistic routing: in probabilistic routing the path decision for each node is relaxed. Instead of selecting a unique forwarder, the current node evaluates the routing probabilities of multiple forwarders based on a cost function. The next hop is decided based on the optimization of the cost function.

In [53] the objective is to maximize the network lifetime based on probabilistic routing in a DAG structure containing all sinks. The solution creates a structure based on the hop count to the sinks, then calculates the routing probability based on the loads of child nodes, the generated data and the routing probabilities of the parent nodes. The routing probabilities are calculated from the top level to the bottom level. The packets are forwarded according to the routing probabilities.

3.2.3/ k-ANYCAST

As in anycast communication scheme in general, the packet is forwarded to a group of sinks, and the definition of the final destination is done during the forwarding process based on the network metrics and the path choices. This is also the basic principle of the approaches considering k-anycast. In this scenario, the packet must be delivered to exactly k different sinks. The packets my be duplicated at source or during the forwarding process.

The authors in [54] propose a data gathering algorithm with a fault recovery scheme especially designed to assure reliability, with redundancy of nodes, paths and data delivery.

The algorithm considers the construction of disjoint trees, rooted at each sink. However, the sensor nodes serving as forwarders are also disjointed, meaning that a node serves as forwarder for exactly one tree. In order to improve reliability, the solution considers forwarding the packet to k sinks. The nodes are always part of exactly k trees, in order to forward the packet to k sinks. As a way to increase the network lifetime, the network is dense and not all nodes are active. Some of the nodes are considered as replacement for future failures in the network. The routing is divided in three phases. The first is responsible for defining the nodes that will be put to sleep and used as replacement for future node failures. During the second phase, the sinks start sending token messages for network discovery. The third phase is related to nodes receiving the token and deciding to participate in the routing tree as a relay node or as a leaf node. The decision depends on the node's hop count to the sink, the residual energy and the relay status. If the node was already used as a relay node, then it cannot be used as a relay node again in a different tree.

In [55] we find a solution which is a variation of the approach presented in [46], already described in Subsection 3.2.2. The solution defines a routing protocol for rechargeable MS-WSN to reduce network latency and optimize the energy consumption, but focused on assuring the delivery by forwarding the packets to k sinks. The algorithm builds spanning trees rooted at each sink. The next hop is decided based on the nearest sink that has to be reached.

In [56] the main objective is to extend the network lifetime at the same time assuring the packet delivery to exactly k sinks. The strategy considers balancing the utilization of nodes in the sink neighborhood. Path is constructed in a greedy way. The current node builds a spanning tree to k sinks with minimum cost. The next hop is decided based on the cost of the energy-weighted shortest path (ESP). At each hop, the algorithm calculates the cost over progress and decides to duplicate the packet towards different nodes in order to reach k sinks. When a packet is duplicated, it is assigned to a set of specific sinks, in order to assure the reception by k sinks.

3.2.4/ MULTICAST

The approaches considering the multicast communication scheme are focused on forwarding the packet to a number of predefined sinks in the network. Packet duplication normally occurs during the forwarding process. The works in [57] and [58] try to find merging points in the path towards the sinks as a way to aggregate the data and to reduce the energy consumption.

In [57] the objective is to relay the messages from multiple sources to multiple sinks while reducing the number of links used to forward the messages. The strategy considers maximizing the overlap of paths from sources to sinks. This way, the same routes are used from different sources to forward messages to sinks. The next hop is defined by selecting the neighbors with more path overlaps and serving more sinks. The change of routes is an iterative process triggered by a timeout.

The objective in [58] is to identify a minimum subgraph with fewer edges than the original graph, enabling the connection of all sources to all destinations. It applies a heuristic method to identify an initial subgraph and improve the identified subgraph by consecutive search iterations. The routes are initially constructed based on the hop distance to the sink. Each source has an independent path to each sink. Then, the algorithm tries to

merge the independent paths from a source to all sinks, reusing as much as possible the same path. Later, the merge takes place with the sources, trying to determine a point of aggregation and replication. The convergence node is responsible of aggregating the message from the sources and re-split it towards the destination sinks.

3.3/ CONCLUSION

In this chapter we presented a classification criteria for routing solutions in multi-sink WSN, listing and describing the particularities related to the existence of multiple sinks in the network. We also presented a brief description of the routing solutions based on our classification criteria.

In MS-WSN solutions categorized as unicast, the multi-path aspect is well explored. The solutions alternate between a graph route structure and the existence of multiple routing trees. The packet is forwarded through one of the available paths towards the addressed sink. For the case of single path solutions, the nodes participate in a single routing tree rooted at one of the sinks.

In the 1-anycast communication scheme, we find the solutions with different routing strategies, such as opportunistic routing, gradient-based, bio-inspired, geographic and probabilistic routing. They all have in common the fact that multiple paths are possible towards any of the sinks. The sink decision is taken on the way, during the forwarding of the packet through the nodes and based on specific routing metrics, such as energy cost, energy consumption, hop count, etc.

For the k-anycast routing solutions, the number of works is reduced, with only three solutions presenting a strategy to forward k packets to k sinks. The works in [55] and [54] propose a strategy where the packets are duplicated at source. Contrary to the work in [56] which proposes the duplication of the packet during the forwarding process.

Finally, only two approaches could be classified as having a multicast communication scheme. The solutions are focused on identifying merging points in the routing paths, in order to aggregate the data and to reduce the energy consumption.

4

OTHER SOLUTIONS IN MS-WSN

Contents

In this chapter we present other solutions considering the existence of multiple sinks in the network. Our goal is to show the impacts of having multiple sinks in different types of solutions. We separate the solutions in two categories: MAC solutions, describing the approaches that consider the existence of multiple sinks as part of the optimization strategy, and the application-level solutions, which are focused on the quantification, deployment and coordination of multiple sinks in a WSN.

4.1/ MAC SOLUTIONS IN MS-WSN

According to [63], the Medium Access Control (MAC) is a sublayer of the data link layer in the OSI model. In wireless sensor networks, a MAC protocol is responsible for controlling the way sensor nodes share the wireless medium. It is charged with many functions, including collision avoidance, transmission range management, listening time etc. The design of a MAC protocol may seek the communication efficiency in different ways, such as reducing collisions and increasing the achievable throughput, providing flexibility for various applications through multi-channel communication, synchronous and asynchronous transmissions, variable duty cycle and so on [64, 63].

The MAC solutions listed in Table 4.1 are focused on wireless sensor networks with multiple sinks. They are also concerned with techniques for communication optimization in terms of latency and energy consumption. The solutions try to efficiently manage the energy consumption through optimal scheduling and low duty cycle at the same time as reducing the transmission delay. We provide a brief presentation of each approach in Subsection 4.1.2.

4.1.1/ CLASSIFICATION

Most of the MAC solutions extend the functions of well-known MAC protocols. According to [63], WSN do not strictly follow one single standard. Solutions are proposed with a mix of different concepts or modified versions of a standard protocol.

As shown in Table 4.1, we classify the MAC solutions regarding the multi-sink aspect at the routing level (communication scheme), to identify how solution evaluation considered the existence of multiple sinks; the physical and data link layers for the standard radio interface; the MAC type, in order to show the method used for medium access, with contention, without contention, duty cycled; and finally the protocol family, in order to indicate if the solution follows the same principles of a well-known family of protocols.

Comm. Scheme	*Standard*	*MAC Type*	*MAC Protocol*	*Ref.*
Unicast	IEEE 802.15.4	Contention-free	TDMA-like	[65]
Unicast	–	Contention-free	TDMA-like	[66]
Unicast	–	Duty Cycled	ALOHA + CSMA/CA	[67]
Unicast	IEEE 802.15.4	Duty Cycled	CSMA/CA	[68]

Table 4.1: Summary of MAC solutions in MS-WSN

4.1.2/ DESCRIPTION

All the solutions listed in Table 4.1 considered a unicast routing protocol as part of the evaluation process. Solutions [66], [67], [68] considered single path and disjoint trees as routing structures. The approach in [65] considered a multi-path strategy with multiple trees rooted at each sink.

In [65] the authors define an energy efficient MAC protocol based on TDMA with an interference-aware strategy adapted to multi-sink WSN, focused on time-sensitive applications. The algorithm considers not only the neighbor nodes, but also the nodes within the interference range, during the assignment of slots. The process happens iteratively, with parent nodes sending a message containing a time slot proposition to their child nodes. The child nodes verify if the slot is already taken by another node (a neighbor or another node within the interference range). If so, the message is not replied and the parent nodes wait the next available slot to resend the proposition. If this time the child node responds to the proposition, the slot is assigned and the information is advertised to other nodes.

In [66] the objective is to reduce the length of the cycle and determine the most efficient time slot schedule in a multi-channel network. The algorithm defines a scheduling table for each sensor using different transmission channels, then it minimizes the number of slots, considering a heterogeneous traffic. Each sink is responsible for different information/applications, so a specific sink may relay to the other sinks the packets it receives.

In [67] the objective is to define a duty cycled medium access strategy based on the routing tree and able to provide real-time communication in a MS-WSN. For the single-source scenario, the information is sent directly to the sink in a staggered communication process. The nodes are synchronized and the packet is raised to upper levels until the sink is reached (synchronous skewed wakeup). For the multi-source scenario, if a collision is

detected, the algorithm makes use of controlling messages to organize the forwarding. In both cases, the packet is first forwarded to a 'primary' sink and then distributed to all other sinks (if necessary).

In [68] the authors define a communication protocol that creates a very low duty cycle schedule based on the routing tree able to extend the network lifetime and assure low latency in comparison to other LPL (Low Power Listening) [69] approaches. The solution takes a sampling period as input and defines an adapted time schedule for the nodes to cooperate in the collecting task. The algorithm works as a management cross-layer system, controlling both MAC and routing layers in a way they cooperate to define a duty cycle schedule based on a routing tree previously defined by the CTP protocol. It works as a hybrid routing protocol, since it has a stateful phase as well as a stateless one. Periodic synchronizations for the duty cycle schedule take place and trigger a redefinition of the routing tree.

4.2/ APPLICATION-LEVEL SOLUTIONS IN MS-WSN

In WSN the optimization strategy can be implemented within any layer of the communication stack. It basically depends on the final objective and the applied mechanisms. If for instance the selected strategy to increase the network lifetime is related to the nodes' placement, it is more likely to be an implementation at the application-level. We call application-level approaches the class of solutions concerned with the application layer. Commonly, the solutions on the application layer are related to resource availability, coordination, communication management, data authentication, encryption etc. Since this work is focused on MS-WSN, the selected solutions are related to strategies focused on optimizing the communication with multiple sinks.

4.2.1/ CLASSIFICATION

The application-level solutions are classified in terms of the solution type, which essentially describes the purpose of the solution. We identify three types of solutions: sink coordination, sink placement and sink quantification, as presented in Table 4.2.

- **Sink Coordination**: this class of solutions focuses on establishing a communication protocol among the sinks in order to achieve better network manageability. Generally, the solution defines a clear interface, with syntax and semantics, to exchange data of any kind within the network. The objective is to organize with other sinks the necessary actions in order to achieve a better performance in terms of load sharing, network lifetime, low latency etc.

- **Sink Placement**: according to [23] the ideal placement of sinks in a MS-WSN leads to better energy consumption and lower latency, since the number of hops is reduced. This class of solutions focuses on determining the best locations for the sinks in a network with sensor nodes already placed in a random or grid way. Different metrics are considered depending on the final objective (longevity, reliability and timeliness). Normally, the position of the sinks is evaluated in conjunction with the routing objectives.

- **Sink Quantification**: determining the ideal amount of sinks in a network is directly connected to the network economical feasibility and application objective. As exemplified by [70], a WSN application may require the network to be reliable and functional for a minimum period of time. As a solution, the number of sinks may be increased, resulting in an extension of the network lifetime. However, it is important to determine the ideal number of sinks to achieve the desired minimum lifetime in order to the network to be economically feasible. Normally, the solutions focusing on quantifying the ideal number of sinks are also concerned with the best placement of the sinks, and in the same way the objective is connected to the application demand (extended network lifetime, low latency etc).

Comm. Scheme	*Solution type*	*References*
Unicast	Sink coordination	[71]
Unicast	Sink quantification and placement	[70][72]
Unicast	Sink placement	[23][73][74][75][76]
1-Anycast	Sink placement	[77]
Multicast	Sink placement	[78]

Table 4.2: Summary of application-level solutions in MS-WSN

4.2.2/ DESCRIPTION

The solutions [73], [74], [75], [23], [76], [70] and [72] considered the deployment optimization strategy as part of a unicast routing with a tree-based structure and single path. The solution [71] considered a unicast communication scheme from the routing perspective, but with a graph-based structure configuring a multi-path strategy. The solution in [77] considers an anycast strategy, with a graph-based structure and the solution [78] considers a multicast communication scheme from the routing perspective, also with a graph-based structure.

4.2.2.1/ SINK COORDINATION

In [71] the objective is to improve the network throughput by distributing the traffic among the different sinks. The sinks communicate with each other, exchanging network information in order to define the best distribution of nodes among the sinks. Each sink keeps the information of the packet delivery rate of other sinks, and after a timer, the re-balance takes place, if necessary.

4.2.2.2/ SINK QUANTIFICATION AND PLACEMENT

In [70] the primary objective is to determine the minimum number of sinks and later their best possible location, in order to enable a MS-WSN to be operational for a predefined period. The authors try to identify the optimal relation between the number of sinks and the network lifetime. The solution works as a brute-force algorithm that identifies the

necessary amount of sinks to fulfill a required network lifetime. The sinks placement makes use of the k-Means approach to identify the best locations.

In [72] the authors propose an algorithm able to determine the amount of necessary sinks and their respective positions, respecting a predefined hop-count constraint and the set of potential sink locations in order to maximize the network lifetime. The strategy consists of maximizing the network lifetime by minimizing the hop-count, at the same time as balancing the network load. The balance is achieved by minimizing the maximum amount of descendant nodes in a branch. The steps of the algorithm consider first the calculation of the sinks coverage at each potential position, respecting the hop-count constraint, then it identifies the minimal subset of sinks with maximum coverage. Finally, the algorithm builds load-balanced disjoint trees, minimizing the maximum number of descendants among the bottleneck nodes.

4.2.2.3/ SINK PLACEMENT

In [23] the objective is to define a self-organized placement algorithm for multiple sinks, able to reduce the worst-case delay scenario, based on the hop count to the closest sink. The algorithm considers dividing the network into sectors. Then, the center of gravity is calculated for each sector and the sink is placed at this position. The 1-hop neighbor nodes connect to the sink, and successively until the n-hop nodes. Once all nodes are assigned to a group, the location of the 1-hop neighbors is estimated. The 1-hop neighbors' location information is used to determine the set of candidate locations for the sink (the corner of a regular octagon). Then the sink moves through each of the candidate locations and calculates the worst-case delay scenario based on the hop count. The best location is chosen based on the minimum value for the worst-case delay.

In [77] the authors propose an LP (Linear Programming) algorithm based on the volume of transmitted data to solve the problem of maximizing the network lifetime by placing the sink nodes at the best locations. The strategy is based on the MAX-MIN scheme, where the objective is to maximize the minimum amount of data generated by a node. This way, the network lifetime is extended, at the same time assuring fairness in data flow distribution. The algorithm considers the position of the existing nodes as candidates for sink placement. The amount of sinks are fixed and the algorithm searches for the best locations.

In [78] the authors propose a method of multiple sinks placement in a gossip-based wireless sensor network, focused on the minimization of the worst case latency. The strategy considers a greedy search based on the location of the existing nodes. Two algorithms are proposed. The first one is a pure greedy (PG) strategy that divides the network into sections and identifies the worst case latency scenario. Later the sinks are placed on the inbound of the worst case scenarios. The latency is evaluated and the new positions are tested. The performance of the strategy is highly dependent of the initial position of the sinks. The second strategy considers a greedy simulated annealing (GSA), which starts as a random selection of sink positions. The probability of a specific position being selected is proportional to the quality of the position in terms of latency. After a number of iterations, the selection of positions converges to a pure greedy strategy, with the candidates weight being the most relevant aspect.

In [74] the objective is to define a sink deployment strategy based on Gene Expression Programming (GEP) to extend the network lifetime and reduce the source-to-sink latency.

Using the GEP, the algorithm places the sinks in a certain initial position. Then the positions of the sinks are evaluated against the Fitness Function, which considers the smallest values of the hop count from nodes to sinks. At this point the algorithm may decide to stop or to execute another iteration. If it decides to execute another iteration, a gene function is executed and the next generation of individuals is created for a new evaluation.

In [75] the authors propose four different placement strategies that consider not only the topological aspect of the network (routing tree), but also physical constraints (the existence of obstacles). The first strategy KSP (K-meanS Placement) is based on the classic k-Means, where the candidate positions are evaluated iteratively until the moment when no more changes on the sink position are detected. The second strategy KDP (K-meDoid Placement) is close to KSP in terms of execution, but it considers the medoid position, which may be defined as the most central object in a cluster. The third strategy is SPP (Shortest Path Placement), which considers the same execution process of the KDP algorithm, differing on the distance measure. Instead of using an Euclidean distance, the new strategy makes use of the Dijkstra algorithm to calculate the shortest path and creates a matrix of distances based on the existence of a link between two nodes. The last strategy is RMP (Routing Metric Placement), which performs the same execution process of SPP, however adding an extra cost to each entry of the distance matrix. The new cost corresponds to the link quality. If a link is found to be weak, an extra cost representing the ETX (expected number of transmissions) is added.

In [73] the authors propose a solution for the sink placement that minimizes the worst-case delay and maximizes the network lifetime, making use of a strategy based on genetic algorithm (GA). The first step of the strategy considers a technique of location discretization. In order to bring the problem to a search space of finite locations, the authors propose a method of discretization of the possible sink location by defining regions of indifference. Such a region is characterized by the fact that the changes in the sink position do not represent a change in the network routing. The GA based solution then considers as entries the nodes position and the number of sinks. The sinks are placed randomly at the discretized locations and the fitness function is calculated considering the worst-case delay. Then the genetic operations are executed and a new generation of individuals is created for the next iteration. The algorithm is stopped after a certain number of generations is reached.

In [76] the objective is to define a sink deployment strategy based on Particle Swarm Optimization (PSO) that minimizes the worst case delay and consequently extends the network lifetime. The algorithm uses the Discrete Particle Swarm Optimization (DPSO) to solve the problem of best sink location. The main strategy is to transform the sink location problem in a discrete problem, to perform a local search for the local best candidate and finally to execute the PSO to determine the global best. The discretization consists of dividing the deployment field into a grid. Each cell represents a portion of candidate locations. Changes in the sink position within a single cell result in no alterations to the routing topology.

4.3/ CONCLUSION

In this chapter we presented a classification and description of MAC and application-level solutions. We can verify that the existence of multiple sinks in the network may also

impact other layers of the OSI model. The solutions designed for MS-WSN at the level of the MAC considered basically the unicast communication scheme, and both single path and multi-path. From a practical point of view, the MAC solutions that considered multi-path could as well run under a different communication scheme, such as anycast or multicast, since the optimizations already considered the existence of multiple paths.

The solutions at application-level were mostly designed considering unicast communication scheme from the routing perspective. The position of the sinks has much influence on the performance of the routing algorithm, which means that an optimization of the sink placement considering a unicast communication scheme may not be equally optimal to a different communication scheme that considers packet duplication, such as k-anycast and multicast.

In summary, the WSN deployments with multiple sinks may benefit from optimizations in different layers of the OSI model. However, the optimizations depend on the selected communication scheme, since the network performance may vary according to the routing strategy. As for instance, if we consider a TDMA-like MAC solution to optimize the time-slot scheduling for a unicast routing protocol considering disjoint trees rooted at each sink in a single path strategy, the same solution cannot be applied to another unicast routing protocol considering multi-path, since the time-slot scheduling optimization strategy may not consider a node participating in multiple routing trees.

5

SYSTEM VALIDATION

Contents

The system validation is an important process of the development of a solution. In general, this is how solutions are proved to be effective, subject to the predefined assumptions, conditions and requirements. A new protocol in multi-sink WSN may be analyzed from a theoretical perspective, and put under test through both simulations and real-life experiments. Considering the complexity of setting up a real wireless sensor network, it is customary to evaluate the solution based on a simulation before submitting it to a real-life experiment.

5.1/ THEORETICAL ANALYSIS

The main difference between the execution of experiments (real-life or simulations) and theoretical analysis is that the simulation is not a self-contained evaluation, since it is valid only for the studied scenario - and extrapolated with trend analysis. In a theoretical analysis the results are always true and valid regardless of the studied scenario - but limited to the system modeling. However, theoretical analysis is generally difficult to perform, due to the many associated variables. Because of that, researches often simplify the task with an abstraction, relaxing some of the constraints.

We could identified three works considering theoretical analysis of their approaches, as presented in Table 5.1 .

Solution type	*Comm. scheme*	*Tool type*	*Network size*	*Ref.*
Routing	Unicast	Numerical software	Tiny	[33]
Routing	Unicast	–	Small	[27]
Routing	Anycast	–	Medium	[53]

Table 5.1: Summary of the solutions with theoretical analysis

5.2/ SIMULATIONS

Simulation is widely applied as testing method to verify the correctness of the solutions. The advantages of simulation are numerous. It is typically easier to implement and to extend testing setups. The environment is under control and highly adaptable to the necessary conditions to test all the desired scenarios. With simulation, reproducibility becomes much simpler, since saving test configurations is a common feature. Another advantage of the simulation in WSN is the network size. It is easier and cheaper to investigate the impacts of increasing the network size with simulation.

Solution type	*Communication scheme*	*Tool type*	*Network size*	*Reference*
Routing	Unicast	–	–	[38]
Routing	Unicast	–	Tiny	[41]
Routing	Unicast	–	Small	[27]
Routing	Unicast	–	Medium	[34][36]
Routing	Unicast	Homemade	Large	[28]
Routing	Unicast	Network simulation	Medium	[26][39][32][40]
Routing	Unicast	Network simulation	Small	[44][37][43]
Routing	Unicast	Network simulation	Tiny	[25][31]
Routing	Unicast	Numerical software	Massive	[35]
Routing	Unicast	Numerical software	Small	[29][30][42]
Routing	Unicast	Numerical software	Tiny	[33]
Routing	Anycast	–	Medium	[46]
Routing	Anycast	Homemade	Medium	[52]
Routing	Anycast	Homemade	Massive	[51]
Routing	Anycast	Network simulation	–	[54]
Routing	Anycast	Network simulation	Small	[56]
Routing	Anycast	Network simulation	Massive	[50][49]
Routing	Anycast	Network simulation	Medium	[45] [47][48]
Routing	Anycast	Numerical software	Medium	[53][55]
Routing	Multicast	–	Massive	[58]
Routing	Multicast	Network simulation	Medium	[57]
MAC	Unicast	Network simulation	Small	[65]
MAC	Unicast	Network simulation	Tiny	[68]
MAC	Unicast	Network simulation	Medium	[67]
MAC	Unicast	Numerical software	Small	[66]
Application	Unicast	Homemade	Medium	[73][23][76]
Application	Unicast	–	Medium	[72]
Application	Unicast	Network simulation	Small	[74][71][75]
Application	Unicast	Network simulation	Medium	[70]
Application	Anycast	Numerical software	Tiny	[77]
Application	Multicast	Homemade	Medium	[78]

Table 5.2: Summary of the simulation classification

In Multi-Sink WSN, it is observed that simulation is the most common method to validate

the solutions. The simulation tools vary largely from homemade tools to well-known network simulators. One particular feature for the simulation scenarios is the network size, from tiny to massive deployments. Table 5.2 summarizes the simulation details of the analyzed solutions.

5.2.1/ NETWORK SIZE

Most of the times the network size is a direct demand of the final application. The amount of deployed nodes is one of the requirements that must be accounted when scheming a solution. A routing protocol for energy consumption optimization in a tiny network does not necessarily work for a large network. So, the solutions have to consider tailoring the development to the amount of sensor nodes the applications require. In addition, to account with the scalability problem, some solutions try to cover the maximum range of sizes as possible, going from tiny to medium or from medium to massive networks.

From the set of studied works, we could identify five network size categories, going from tiny networks with less than fifty sensor nodes to massive networks with more than one thousand sensor nodes:

- **Tiny**: equal or less than 50 sensor nodes;
- **Small**: more than 50 and equal or less than 100 sensor nodes;
- **Medium**: more than 100 and equal or less than 500 sensor nodes;
- **Large**: more than 500 and equal or less than 1000 sensor nodes;
- **Massive**: equal or more than 1000 sensor nodes.

Nevertheless, we detected that most of the solutions, 46% of all studied papers, are validated with medium size networks, with an average of approximately 250 sensor nodes and a standard deviation of nearly 140.

5.2.2/ SIMULATION TOOLS

Most of the simulation tools are generic. Some of them are oriented towards network simulation, and others are meant to be used for every environment accepting discrete event simulation. There are also extremely specific simulation tools, which are in general homemade simulators created specifically for a particular solution to be tested. Among the network simulators used in the studied papers, only COOJA, TOSSIM and WorldSens are notably designed for wireless sensor networks. For some generic simulators, auxiliary modules have been developed to address specific features of sensor networks (such as Castalia [79] for OMNet++).

Simulators are generally designed to address different levels. For example ns-2/ns-3 simulators (networking level) are mainly designed for network without taking the hardware properties into account, while TOSSIM (operating system level) is intended particularly for simulating the behavior of the operating system TinyOS, which is also the case for COOJA. As for OMNet++, it is oriented towards the application level.

- **Homemade**: C++, JAVA and others

  In many cases, authors decide to develop a homemade simulator. We call homemade simulation tools, the class of tools developed by researchers for the specific purpose of testing their propositions. Normally, homemade tools give more freedom for researchers to explore the specificities of their solutions. The programming language choices of a homemade simulator vary according to the available APIs/modules and the implementation easiness.

- **Numerical Softwares**: MATLAB, MOSEK, Octave, SciLab

  MATLAB [80] is the high-level language and interactive environment used by engineers and scientists to analyze and design systems. The matrix-based language is a natural way to express computational mathematics. Networks are commonly modeled as graphs and MATLAB offers structures and function to manipulate directed and undirected graphs, with the objective of network analysis.

  MOSEK [81] is a tool for solving large, mathematical optimization problems. Several network problems, as the one of minimizing the overall average travel time for all network traffic, can be solved as optimization problems.

  GNU Octave [82] is a high-level language primarily intended for numerical computations. It helps solving linear and nonlinear equations, numerical linear algebra, statistical analysis, and performing other numerical experiments. It may also be used as a batch-oriented language for automated data processing. Functions relevant to network/graph analysis are developed in toolboxes (such as Octave routines for network analysis) to explore simple graph statistics, distributions, visualization and to compute common network metrics.

  Scilab [5] is a free and open source software for numerical computation providing a powerful computing environment for engineering and scientific applications. A large number of functionalities is included in SciLab: control, simulation, optimization, signal processing, graph/network manipulations etc.

- **Network Simulation**: COOJA, NS-2 / NS-3, OMNET++, OPNET, QualNET, TOSSIM, WSNet

  COOJA [83] is a network simulator for the Contiki operating system. Contiki OS [6] is a lightweight open source operating system designed for resource limited devices (sensor nodes included). Developed at the Swedish Institute of Computer Sciences by Adam Dunkels et al, it is written in C programing language and provides portability to different platforms. It offers both full IP (Internet Protocol) networking and low-power radio communication mechanisms. COOJA is a Java-based application simulating the behavior of the Contiki operating system. Moreover, it emulates various sensor nodes using dedicated emulator programs for the micro-controller units, and the radio transceivers. Currently, Avrora [84] is integrated for the emulation of Atmel AVR-based devices and MSPSim [85] for the emulation of TI MSP430-based devices. The COOJA simulator can work at different levels - called cross level simulations: networking level, operating system level and machine code instruction set level.

  Network Simulator [86] is probably the most exploited simulator for network traffic simulations, both wired and wireless. It is a discrete event simulator developed in C++ and targeted at networking research. Actions are associated with events rather than time. An event consists of an execution time, a set of actions, and a

reference to the next event. These events are connected to each other, forming a chain of events on the simulation timeline. Unlike a time-driven simulator, in an event-driven simulator, time between a pair of events does not need to be constant. When the simulation starts, events in the chain are executed chronologically. Its second version, ns-2 [86], has been replaced by ns-3 [87], which is a new simulator, designed as a set of libraries that can be combined together and also with other external software libraries. Its main goal was to improve the realism of the models, using paradigms of component-based programming. Components of ns-2 dedicated to packet processing are written in C++ and those for managing control (used to create simulation scenarios) are in OTcl. In ns-3, the simulator is written entirely in C++, with optional Python bindings. Simulation scripts can therefore be written in C++ or in Python. The main advantage of the Newtork Simulator is the large number of protocols already implemented.

OMNeT++ [88] is an object-oriented modular discrete event framework which is being used, among others, for wired and wireless network simulations. One of the fundamental ingredients of the OMNET++ infrastructure is a component architecture for simulation models. The components are developed in C++ while models use a high-level language called NED (Network Description language). Modules can be interconnected through ports and communicate by message passing. This generic architecture allows modeling and simulation of any system based on discrete events.

OPNET Modeler [89, 90] is part of the OPNET Technologies suite, a commercial software. It supports specification, modeling and simulation of communication protocols and protocols. OPNET Modeler includes a vast library of communications devices, communication mediums, and cutting-edge protocols. The engine of the OPNET Modeler is a finite state machine model in combination with an analytical model.

QualNet [91] is a commercial software, specialized in simulating different wireless applications executing on large and heterogeneous networks. It is a C++-based discrete-event simulator, with a layered protocol design: Physical, MAC, Network, Transport and Application. The simulator provides a comprehensive environment for designing protocols, creating and animating network scenarios, and analyzing their performance.

TOSSIM [92] is an event-driven simulator for the TinyOS operating system designed for sensor networks. In this feature, it is similar to the COOJA simulator. However, these tools differ in the way nodes are represented. In TOSSIM, several nodes are simulated by changing the sensor node code. In COOJA, the executed code remains unchanged. TOSSIM enables the simulation of the entire TinyOS applications by replacing few low-level components with simulation implementations. The simulation events represent hardware interrupts, high-level system events and posted tasks.

WSNet [93] is an event-driven simulator tool, part of the WorldSens integrated environment for the design, development, prototyping and deployment of wireless sensor applications. It can be used in cooperation with or independently from the second simulator tool of WorldSens, WSim. Its role is to simulate the hardware behavior and the events that occur in the platforms. Each node of the sensor network is simulated by a WSim program while the radio medium simulation is performed by WSNet. The later simulates the radio propagation, collisions, introduces errors and

returns the received packets using UDP/IP communications. Both simulation tools are developed in C.

5.3/ REAL-LIFE EXPERIMENTS

The amount of works considering real-life experiments in MS-WSN is low. Only 10% of the studied papers presented results from real-life experiments. Another observation is that different operating systems and hardwares are available (such as Contiki, Tiny OS - see Table 5.3), and the choice for one or another is not always clear.

For the operating systems, the studied papers proposed experiments using TinyOS, Contiki and VROS. Both Contiki and TinyOS were already briefly presented in Subsection 5.2.2. VROS [94] is a light-weighted operating system for wireless sensor nodes, supporting multitasking, with deadlock prevention and based on message queue scheduling.

In terms of hardware, both MICA2 (MPR400CB) [95] and MICAz (MPR2400) [96] belong to the same family of motes, based on the Atmel ATmega128L and compatible with TinyOS. The motes TelosB [97], TmoteSky [98] and SkyMote [6] are based on the Texas Instrument MSP430, compliant with IEEE 802.15.4, also compatible with TinyOS and Contiki [6]. VNODE [94] is based on the Atmel ATmega128L, designed by VRLab, compatible with VROS and TinyOS. The hardware eZ430-RF2500T [99] is a Texas Instrument MSP430, for multi-purpose low-power wireless applications.

In all solutions, the real-life deployments were fixed, with no variation of the nodes' position. The size of the networks was also reduced, with a maximum of 56 nodes.

Solution type	*Comm. scheme*	*Hardware*	*OS*	*Application*	*Number of sinks / nodes*	*Ref.*
Routing	Unicast	SkyMote	Contiki	Validation of a multi-sink routing protocol for RPL	2/56	[43]
Routing	1-Anycast	eZ430-RF2500T	–	Forest Monitoring	1/16	[46]
Routing	Unicast	VNODE	VROS	Validation of a load-balance routing algorithm	6/30	[41]
MAC	Unicast	MICA2	TinyOS	Validation of a real-time, low energy MAC protocol	3/11	[67]
MAC	Unicast	TmoteSky, TelosB, MicaZ	TinyOS	Long-term environmental monitoring	1/14	[68]

Table 5.3: Summary of solutions with real-life experiments

6

Conclusion

In this part, we presented a particular perspective of the WSN solutions in literature, considering the particularities regarding the existence of multiple sinks. In general, the range of solutions considering a multi-sink topology is still small compared to the single sink case. Additionally, most of the solutions for MS-WSN are focused on extending the network lifetime, having longevity as the main objective. The analysis on the state of the art reveals that only 10% of the solutions considered reliability as the main objective and 26% envisioned timeliness. That results in 64% of the papers dedicated to longevity. We summarize in Table 6.1 the distribution of the solutions in terms of objectives.

Solution level	*Main objective*	*Comm. scheme*	*Processing type*	*References*
Routing Layer	Longevity	Unicast	Centralized	[28][30][31][32][34][38]
Routing Layer	Longevity	Unicast	Distributed	[42][40][37][39][35][41]
Routing Layer	Longevity	Unicast	Hybrid	[43][29][26][33]
Routing Layer	Longevity	1-Anycast	Distributed	[45][47][51][50][53]
Routing Layer	Longevity	k-Anycast	Distributed	[56]
Routing Layer	Longevity	Multicast	Distributed	[57][58]
Routing Layer	Reliability	Unicast	Distributed	[36]
Routing Layer	Reliability	Unicast	Hybrid	[44]
Routing Layer	Reliability	1-Anycast	Distributed	[52][49]
Routing Layer	Reliability	k-Anycast	Distributed	[54]
Routing Layer	Timeliness	Unicast	Centralized	[27][25]
Routing Layer	Timeliness	1-Anycast	Distributed	[46][48]
Routing Layer	Timeliness	k-Anycast	Distributed	[55]
MAC Layer	Longevity	Unicast	Centralized	[68]
MAC Layer	Timeliness	Unicast	Centralized	[65]
MAC Layer	Timeliness	Unicast	Distributed	[67][66]
Application Layer	Longevity	Unicast	Centralized	[74][75][70][76][72]
Application Layer	Longevity	1-Anycast	Centralized	[77]
Application Layer	Timeliness	Unicast	Centralized	[73]
Application Layer	Timeliness	Unicast	Distributed	[23]
Application Layer	Timeliness	Unicast	Hybrid	[71]
Application Layer	Timeliness	Multicast	Distributed	[78]

Table 6.1: Summary of MS-WSN solutions

Considering the communication scheme, the solutions are divided mainly between unicast and anycast, with respectively 67% and 27% of all solutions focused on these two schemes. It shows that the multicast scheme, summing up 6% of the solutions, is largely neglected. This reveals that the potential and advantages of MS-WSN are not yet entirely explored, leaving many blank spots to be filled. Improvement opportunities connected to application requirements are distributed among the different main objectives and communications schemes. As for instance, multicast is an interesting solution for applications that consider availability of data as a key requirement.

Cross-layer optimizations are also shortly used, only 30% of the papers presented a solution considering cross-layer interactions, and none of them with multicast communication scheme. The existence of multiple sinks can be explored in different ways with cross-layer. An example is the slot allocation based on multiple routing trees rooted at each sink, focusing the latency and interference minimization.

In terms of deployments, the size of the network is another aspect that is not largely tested. The majority of the solutions are focused on small and medium networks. Approximately 70% of the studied papers considered networks between 50 and 500 sensor nodes. That leaves large and massive networks being covered by only 12% of the solutions. The scalability issue is not completely addressed, with really few works considering massive deployments.

Another interesting observation is related to the system validation method. Simulation is the most preferred method used by researchers to validate the solutions. In 90% of the studied papers the solutions were validated only with simulation, which means that real-life experiments are still rare in MS-WSN. The simulation methods are divided among network simulation tools (48%), homemade tools (14%), numerical software (10%) and process simulation (2%). Among the Network Simulation tools, only one third of the works, totaling 8 solutions, used network simulation tools primarily designed for wireless sensor networks - COOJA (3), TOSSIM (4) and WSNet (1). The other 16 solutions were validated using more generic tools - NS-2 (8), NS-3 (1), OMNET++ (3), OPNET (3) and QUALNET (1).

In summary, wireless sensor network protocols remain an important research topic with many works being presented over the years. The design of new protocols and innovative solutions in WSN enable advancements in correlated areas and in applications for different domains. It is a very dynamic field, subject to constant evolution.

III

CONTRIBUTIONS

MOTIVATION FOR OUR CONTRIBUTIONS

Up to this point, we presented an overview of the main definitions in multi-sink wireless sensor networks, along with the particularities related to the existence of multiple sinks. We proposed a classification criteria for the categorization of the existing approaches in MS-WSN, and we analyzed the existing solutions according to the defined criteria. We enlisted the solutions based on the corresponding layer in the OSI model, the communication strategy from the routing perspective and their main optimization objectives. All this work allowed us to have a better view of the state of the art in MS-WSN, and to identify the opportunities of improvement.

For the first phase of our investigation, we focus on routing strategies and we set timeliness as our main optimization objective. We consider the strategies involving cross-layer optimizations, in order to reduce the end-to-end delay. We analyze the existing solutions focused on the unicast communication scheme and having timeliness as the main objective.

In terms of objectives, we could notice that most of the solutions deal with the energy consumption and network lifetime optimizations. Because of that, we propose a solution considering the optimization of the end-to-end delay. We identify that for the unicast communication scheme, only two works with latency optimization are proposed. We select the work in [25] as a comparison reference, because of the compatibility of the assumptions. The work in [25] considers a contention-based MAC, with the latency optimization accounting for a cross-layer strategy of MAC-aware routing. We suggest improvements in the cross-layer strategy, transforming it in a back-and-forth MAC and routing strategy. We also implement a flow control strategy in order to reduce the buffer occupancy and to enable the increase of the network, since the solution in [25] considered only tiny networks of 50 sensor nodes. We evaluate the scalability capacity of the solutions, considering the case of scalability without proportion, which means that the increase in the number of sensor nodes in not followed by the increase of the number of sinks.

In a second phase, we direct our attention to the applications that require the packet forwarding to many sinks. We can verify that, for the communication scheme at the application perspective, most of the solutions are designed for the case of many-to-one communication scheme. From the routing perspective, this situation translates to approaches considering unicast and 1-anycast. Although it is possible to implement many-to-many applications using unicast and 1-anycast, as described in Subsection 2.2.2, some limitations exist and the final solution may not be as efficient as it is for the many-to-one case.

We can recall from the state of the art that only one solution [55] considering k-anycast deals with the optimization of the communication in terms of latency as main objective. However, the solution considers rechargeable nodes and requires a lot of control mes-

sages to set up the initial state of the network. The level of network knowledge required for the routing establishment is elevated and the solution is not capable of assuring that exactly k sinks receive the packets. Because of that, we select [56] as the reference solution, since the presented strategy is capable of assuring the packet forwarding to exactly k sinks. Moreover, the solution is based on geographic routing, which requires fewer controlling messages and network knowledge. However, the solution in [56] is only focused on the optimization of the energy consumption. So, we propose a k-anycast solution capable of finding a trade-off between latency and network lifetime optimizations. We apply a cross-layer strategy of MAC-aware routing, with the ordering of the packet forwarding being based on the wake-up schedule of the neighbor nodes. Additionally, by proposing a k-anycast solution, we are covering not only the case of k-anycast, but also the case of 1-anycast, since the k-anycast solution behaves as 1-anycast when $k = 1$.

Despite of the flexibility of k-anycast solutions, they are generally not well designed to send packet to a number of predefined sinks. It means that in order to replicate the packet towards all predefined sinks, when the number of predefined sinks is smaller than the number of sinks in the network, the k-anycast solution must be adapted. The k value must be set to the number of predefined sinks, and the target destinations must contain only the set of predefined sinks. Because of that, multicast solutions are better fit for the case the packets must be delivered to a set of predefined sinks. We can remark from the state of the art that only two strategies consider the multicast communication scheme, and none of them have timeliness as main objective. Moreover, they are hop-count based solutions, requiring an important amount of network knowledge in order to perform the routing strategy and optimizations. In that sense, we propose a multicast solution that is based on geographic routing and focused on optimizing both the network latency and lifetime. Our approach tries to improve the process of forwarding the packet to multiple destinations by searching for common forwarders in order to avoid the packet duplication.

Finally, for the last phase of our investigation, we explore the execution of real-life experiments. We can retrieve from the state of the art that very few works consider validating the solutions with real-world experiments and testbeds. Moreover, none of the solutions considered many-to-many applications. They are all focused on the unicast and 1-anycast communication schemes. In that sense, we submit our multicast strategy to real-life experiments using the FIT IoT-Lab testbed as the platform for the execution. We analyze and confront the results of the real-life experiments for our solution against an existing k-anycast solution. We also compare the real-life results against the results obtained from simulations.

From this point forward, we start describing our contributions and the results associated to the simulations and the real-life experiments.

7

UNICAST

Contents

In unicast communication scheme the information is addressed to one sink. The definition of the destination sink may be fixed or at source, depending on the application or the routing structure. The routing path towards the sink may change as a consequence to a change in the network, such as the occurrence of node failure or an increase in the packet latency. However, the addressed sink is not changed during the packet forwarding.

Applications demanding a many-to-one communication with predefined targets may use unicast as a way to implement the routing strategy. As for instance, in smart buildings, with heating and lighting being controlled autonomously, a sensor node may collect both temperature and luminosity information, and send them separately to different sinks, one responsible for the heating and another one responsible for the lighting (sink decision at source). In another example, the generated packet may be forwarded to the closest sink. The source node is part of a disjoint tree rooted at the closest sink. This could be the routing strategy of a smart health system, with sensor nodes collecting patient data and forwarding them to the nearest sink. The packets must be delivered to the closest sink, respecting a time constraint, because of the perishable aspect of the data.

Our objective in this chapter is to present a unicast routing algorithm based on MAC information to select efficient paths in terms of latency and to control the nodes' buffer size, avoiding the packet drop due to buffer overflow. The latency is evaluated for each node towards each sink and the path with the lowest latency is selected. The buffer size is also considered in order to reduce congestion at sink vicinity. At node level, the latency

of the neighborhood is evaluated and the back-off time is changed in order to prioritize nodes facing an increase in latency.

7.1/ MOTIVATION

The transmission delay affects the network latency, compromising the performance of the network and consequently the efficiency of the final application. It is a noteworthy issue in multi-hop WSN, especially for time sensitive networks. This situation happens because of the cumulative aspect of the delay in a path, resulting into high latency. It means that packets are held up at each hop and accumulate time lags in a node-to-sink route. This phenomenon can be caused by many factors, including collisions, channel unavailability, congestion, unbalanced traffic and prolonged back-off periods.

In that context, it is important to find ways to simultaneously tackle the majority of the mentioned problems in order to benefit from a global improvement in terms of network latency. To achieve this objective we may apply cross-layer techniques, evaluating and making decisions based on parameters in different layers. Aligned to that, the DD protocol [25] presents an interesting method to solve the high latency issue. Its cross-layer approach, especially designed to create routes based on the delay at each node, enables a dynamic reaction to paths presenting high latency. The traffic balance also contributes to the load fairness in tree branches, diminishing the effects of the channel unavailability and collisions. However, we can still notice some opportunities of improvements that can be explored:

- **Fixed threshold**: a predefined deadline value is used as threshold for routing changing decision. Because of that, some nodes may present packets with a latency near the defined deadline.
- **Corrective response**: network routes are only changed when a path latency overreaches the predefined deadline metric. This means that the network may face higher latency than the deadline from time to time.
- **No priority**: There is no mechanism to give more chances to nodes facing high delays to send packets with priority.

For these reasons, we propose improvements on the DD protocol focused on proactive measures to build routing paths with low end-to-end delay.

7.1.1/ SYSTEM MODEL AND ASSUMPTIONS

We represent a MS-WSN as a graph $G = (V, E)$, where V represents the set of all nodes and E the set of existing wireless links. Each $e \in E$ corresponds to a pair of nodes (i, j) as long as i and j, with $i \in V$ and $j \in V$, are within each other's transmission range (symmetric link). The set of neighbors of i is represented by V_i. We also specify that $V = S \cup N$ where S represents the set of sink nodes and N the set of sensor nodes, with $S \cap N = \emptyset$.

We assume that the network is coordinated by a central entity that aggregates and coordinates all sinks. We consider that a node is able to send a packet at any moment, and that the access to the medium is a contention-based strategy with collision avoidance.

The latency is the time a generated packet takes to be routed from the source node up to the destination sink. We call *node latency* the average latency of the packets generated and forwarded by the node. When a node is called to be out of deadline, it means that the node latency surpasses a predefined deadline.

7.1.2/ DD PROTOCOL

DD is a routing solution for MS-WSN proposed in [25]. It is an heuristic solution for a problem presented by the authors as *deadline-aware forest construction problem*. Its main objective is to build disjoint trees rooted at the sink to collect information in a timely way, respecting a predefined deadline. Each node participates in only one tree. The algorithm is divided in two phases: initial tree construction and tree reconstruction. Delay and latency are the two main metrics used during the process of selecting nodes of a path. The forwarding delay on a link e is estimated following the process presented in [100] and summarized through equation 7.1:

$$delay(e) = E[N_c](E[BD] + T_c + T_o) + (E[BD] + T_s) \tag{7.1}$$

where,
$E[N_c]$ represents the number of collisions until a message is successfully transferred,
$E[BD]$ represents the average back-off delay,
T_o represents the time to check channel availability,
T_c represents the time to detect a collision,
T_s represents the time to detect a successful transmission.

The latency is then evaluated for each path (node n towards sink s) considering equation 7.2:

$$SendTime(n) = \sum_{e \in path(n,s)} delay(e) \tag{7.2}$$

where,
$SendTime(n)$ represents the total latency from node n to sink s,
$path(n, s)$ represents the current path from node n to sink s.

7.1.2.1/ INITIAL TREE CONSTRUCTION

A node n advertises its existence to other nodes. Then, the neighbors of n reply to the advertisement with their identifier, and if they are already connected to a tree. The node n selects as a parent the neighbor with the smaller hop distance to the sink. If no sink information is received, then node n randomly selects a neighbor node to be its parent p_n. The node n sends to p_n the list of its neighbors. The parent p_n then forwards this information to its own parent. If a loop is detected, the information process is re-executed and the node n chooses another parent.

7.1.2.2/ TREE RECONSTRUCTION

The tree reconstruction phase begins after evaluating the result of equation 7.2, and if the evaluated node is found to be out of deadline, which means that the packets transmitted from this node are received at the sink with a latency that surpasses a specified limit. The tree reconstruction happens in two stages by firstly evaluating the possibility of connecting leaf nodes to other parents, and then by reducing the traffic load on a certain branch.

A list of leaf nodes out of deadline D_n is created using the $SendTime(n)$ evaluating process. For each leaf node n in the list D_n, the algorithm tries to find a new parent p_n among the node's neighbors V_n. Once a neighbor node $v_n \in V_n$ respecting the deadline is found, it is connected to n and becomes its parent, as presented in Figure 7.1. The old parent of node n may become a leaf node, and therefore inserted in the D_n list if it is found to be out of deadline. After these procedures, if a node is still found to be out of deadline, the next stage is started.

The objective of the second stage is to reduce the amount of traffic on a tree branch. This is made by moving nodes from one branch to another. Firstly, the algorithm identifies a common parent node i of the nodes in the D_n list. Then, the children nodes of i are moved to a new parent if they are out of deadline, as in Figure 7.2. Finally, node i is connected to a new parent. This process is executed until no more nodes are found in the D_n list.

- **Leaf reroute**: identify a new parent node that satisfies the latency constraint, as seen in Figure 7.1.

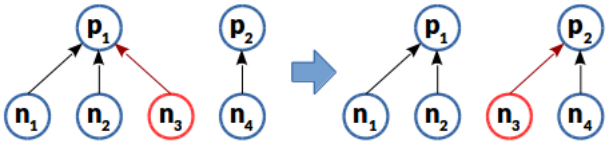


Figure 7.1: Leaf rerouting process

- **Partial tree reconstruction**: reduce the traffic load in a path in order to minimize the latency, as in Figure 7.2.

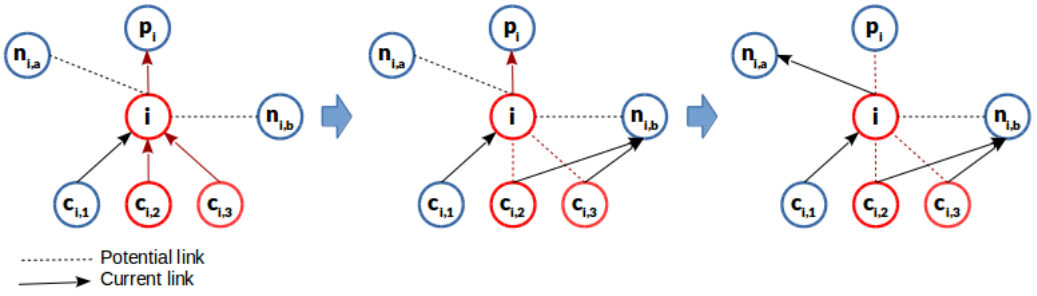


Figure 7.2: Tree reconstruction process

As already presented in Section 7.1, DD has no preventive measure to avoid out of deadline packets. It only reacts to packets that were already noticed to be out of deadline.

7.2/ DYNAMIC BACK-OFF IN DD (DB-DD)

We propose DB-DD, a multi-sink cross-layer routing protocol, working in a unicast communication scheme, with a contention-based MAC. The main goal of DB-DD is to reduce the node-to-sink latency using preventive measures at the MAC and routing layers, in order to avoid that the forwarded packets overreach a predefined latency deadline. At the same time, DB-DD foresees actions to enable scalability, taking into consideration dynamic metrics that change over time, depending on the network size.

7.2.1/ PREVENTIVE RESPONSE

Algorithm 1 High Latency Nodes (HLN) algorithm

Input: N: set of all sensor nodes, $Deadline$: defined deadline
Output: HLN: set of nodes to be rerouted

1: $HLN \leftarrow \emptyset$
2: **for all** $n \in N$ **do**
3: /* Check if n is a leaf node */
4: **if** $GetChildren(n) = \emptyset$ **then**
5: /* $SendTime(n)$ returns the latency from node n to its corresponding sink, based on the tree it belongs */
6: **if** $SendTime(n) \geq Deadline$ **then**
7: $HLN \leftarrow HLN \cup \{n\}$ /* packets forwarded by n are out of deadline */
8: **else**
9: $V_n \leftarrow GetNeighbors(n)$
10: **if** $SendTime(n) \geq LocalThreshold(V_n)$ **then**
11: /* n has a higher latency than the nodes in its vicinity */
12: $HLN \leftarrow HLN \cup \{n\}$
13: **end if**
14: **end if**
15: **end if**
16: **end for**

We call preventive response the algorithm's attempt to anticipate the nodes that will be out of deadline. The algorithm evaluates the latency of the nodes at the neighborhood of the current node. Let us consider HLN the set of nodes facing high latency, n the node to be evaluated, V_n the neighborhood of node n, and S the set of all sinks, with $s_i \in S$. The average latency of the nodes in the specified neighborhood and its standard deviation are obtained by equations 7.3 and 7.4 respectively. If a path from n to s_i is noticed to have a higher latency than the value verified in equation 7.5, then n is included in HLN and assigned to a new parent p'. Algorithm 1 describes the construction process of the HLN set. The algorithm evaluates only the leaf nodes (line 4). If the node n is out of deadline, it is included in the HLN set (line 6). Otherwise, the algorithm evaluates the average latency of the neighborhood of the node. If the packets forwarded by node n present a higher latency compared to its neighborhood, the algorithm includes n in the HLN set (lines 10 to 12). We assume that all packets have the same priority and only data packets are considered for the latency evaluation.

$$\bar{x}(V_n) = \frac{1}{|V_n|} \sum_{v \in V_n} SendTime(v) \tag{7.3}$$

$$\sigma(V_n) = \sqrt{\frac{\sum_{v \in V_n} (SendTime(v) - \bar{x}(V_n))^2}{|V_n|}} \tag{7.4}$$

$$LocalThreshold(V_n) = \bar{x}(V_n) + \sigma(V_n) \tag{7.5}$$

7.2.2/ DYNAMIC BACK-OFF TIME

Algorithm 2 Backoff variation algorithm

Input: N: list of all sensor nodes, $Deadline$: defined deadline, F_{max}: default maximum contention window

Output: W^n_{max}: new maximum contention window for $n \in N$

```
for all n ∈ N do
    /* SendTime(n) returns the latency from node n to its corresponding sink */
    if SendTime(n) ≥ Deadline then
        W^n_max ← (1/3) F_max
    else
        V_n ← GetNeighbors(n)
        if SendTime(n) ≥ LocalThreshold(V_n) then
            W^n_max ← (2/3) F_max
        else
            W^n_max ← F_max
        end if
    end if
end for
```

In order to prioritize the communication of nodes facing high latency, we vary the back-off period (F_{min} and F_{max}) by applying a factor that defines the interval in which the back-off oscillates. This way, nodes experiencing high latency problems may try to access the channel more frequently. The process is described in Algorithm 2 and applies the following rules:

- **Scenario 1**: nodes in this scenario are not facing high latency and they attempt to access the channel in normal conditions. The back-off randomly varies in the interval $[F_{min}, F_{max}]$.

$$SendTime(n) < LocalThreshold(V_n) \text{ and } SendTime(n) < Deadline$$

- **Scenario 2**: in this case, nodes must have more chances to access the channel, since they are already facing a higher latency than their neighbors. The back-off randomly varies in the interval $[F_{min}, \frac{2}{3}F_{max}]$.

$$SendTime(n) \geq LocalThreshold(V_n) \text{ and } SendTime(n) < Deadline$$

- **Scenario 3**: finally, nodes in this scenario must access the channel more frequently. The latency is already above the predefined $Deadline$. The back-off randomly varies in the interval $[F_{min}, \frac{1}{3}F_{max}]$.

$$SendTime(n) \geq Deadline$$

7.2.3/ OPERATION

The algorithm builds disjoint routing trees rooted at each sink. Each node participates in only one tree. The initial is constructed based on the minimal hop distance from node n towards one of the sinks. The process starts with a node n sending a $DiscoveryMesssage$ to the nodes in its direct vicinity V_n. Each node v_i in V_n verifies if it has already received a $DiscoveryMesssage$ from another sink or from a different neighbor node. If the minimal hop count to the sink can be reached through the neighbor node n, then n is saved as a parent node. The node v_i updates its internal routing table with the hop-distance and path to the sink. If the nodes in V_n detect two or more equally distant sinks, they randomly choose one path. Node v_i advertises the new path information to its neighbors. If node n detects it is the new path to the sink for the node v_i, it saves the v_i identification as a child node.

As already detailed in equation 7.5, the $LocalThreshold(V_n)$ is calculated based on the node n vicinity. The nodes in N periodically collect information from their neighbors, in order to be able to calculate the $LocalThreshold(V_n)$ metric. The tree reconstruction phase takes place after the detection of nodes violating either the $LocalThreshold(V_n)$ or the defined $Deadline$ bounds. The reconstruction process is divided in two steps: leaf rerouting and traffic load redistribution. A list of nodes violating the bounds (HLN) is constructed during the periodic network performance evaluation.

Algorithm 3 Leaf reroute

Input: HLN: list of nodes to be rerouted, $Deadline$: defined deadline
Output: New path for nodes in HLN, Updated HLN

```
1: R ← ∅
2: /* Leaf rerouting process */
3: for all n ∈ HLN do
4:    V_n ← GetNeighbors(n)
5:    v_min ← −1
6:    latency_min ← ∞
7:    for all v_i ∈ V_n do
8:       if SendTime(v_i) < Deadline and SendTime(v_i) < latency_min then
9:          v_min ← v_i
10:         latency_min ← SendTime(v_i)
11:      end if
12:   end for
13:   if v_min ≠ −1 then
14:      ChangeParent(n, v_min) /* Set v_min as the new parent of the node n */
15:   else
16:      R ← R ∪ {n}
17:   end if
18: end for
19: HLN ← R
```

The entire cycle starts with the leaf rerouting step, shown in Algorithm 3. For each node n in HLN, the algorithm tries to find a suitable new parent node p' where $SendTime(p')$

does not violate the imposed bounds (line 8). If a suitable neighbor node is found, the parent of n is changed (line 14). However, if no neighbor node presents a $SendTime$ that does not violate the deadline, the node n is kept in the HLN set (line 16).

Algorithm 4 Traffic load redistribution

Input: HLN: list of remaining nodes to be rerouted
Output: New path for each node in HLN

```
1:  /* Traffic load redistribution */
2:  repeat
3:    /* C is the matrix (parents, count of children): [p, |GetChildren(p) ∩ HLN|] ∀p ∈ P */
4:    /* and P ← {GetParent(h) | h ∈ HLN} */
5:    C ← CommonParents(HLN)
6:    /* Returns the parent node with the maximum number of children */
7:    c_max ← MaxChildCount(C)
8:    /* Select all the child nodes of c_max in HLN */
9:    X ← X ∪ {x_i | x_i = GetChildren(c_max), x_i ∈ HLN}
10:   /* Change the parent of the child nodes */
11:   for all x_i ∈ X do
12:     V_{x_i} ← GetNeighbors(x_i)
13:     p' ← −1
14:     latency_min ← ∞
15:     for all v_i ∈ V_{x_i} do
16:       if SendTime(v_i) < latency_min and v_i ≠ c_max then
17:         p' ← v_i
18:         latency_min ← SendTime(v_i)
19:       end if
20:     end for
21:     if v_min ≠ −1 then
22:       ChangeParent(x_i, p')
23:     end if
24:     HLN ← HLN\{x_i}
25:   end for
26:   /* Change the parent of the common parent */
27:   V_{c_max} ← GetNeighbors(c_max)
28:   p'' ← −1
29:   latency_min ← ∞
30:   for all v_i ∈ V_{c_max} do
31:     if SendTime(v_i) < latency_min and v_c ≠ GetParent(c_max) then
32:       p'' ← v_i
33:       latency_min ← SendTime(v_i)
34:     end if
35:   end for
36:   if v_min ≠ −1 then
37:     ChangeParent(c_max, p'')
38:   end if
39: until |HLN| > 0
```

After the leaf rerouting, if it was not possible to remove all nodes from HLN, the algorithm executes the traffic load redistribution step, trying to re-balance the traffic by forcibly

changing nodes from one tree branch to another, as described in Algorithm 4. Firstly, the algorithm verifies the existence of a common parent within the nodes in HLN and creates a set of common parents C (line 5). The parent node (c_{max}) having the largest number of child nodes in HLN is selected as the node to have its child nodes changed (line 7). The child nodes from the common parent node violating the imposed bounds are changed to a new parent node p', different from c_{max} (lines 11 to 25). The algorithm then checks in c_{max} vicinity a suitable new parent node p'' for c_{max} (lines 27 to 38). This process is repeated until all nodes in HLN are removed.

As a continuous process during the nodes communication, and whenever a back-off period must be set, the algorithm reuses the values detected for the nodes $SendTime(n_i)$, to evaluate the adherence to the $LocalThreshold(V_n)$ and the defined $Deadline$, in order to decide about the back-off factor to be applied. The back-off interval is changed according to the scenarios described in Subsection 7.2.2, so that a communication priority to nodes facing high latency can be applied.

7.2.4/ SIMULATION AND RESULTS

According to [25], the performance evaluation of the DD algorithm was executed through simulations and compared to AODV [101]. We implemented both DD and AODV, and executed simulations with DD and AODV in order to assure that the results presented the same behavior described in [25]. We were interested in the relative performance gain from DD in relation to AODV. Both algorithms were implemented with SciLab [5], which is the tool we also used for the network generation algorithm, described in Annex A. All solutions are centralized approaches and the network communication aspects we needed could be simulated using SciLab, such as packet collisions within the node's communication range. Moreover, the integration between the simulator and the network generator algorithm was straightforward, since both were implemented with the same tool. The performance of our solution was also evaluated through simulations using SciLab and compared to both DD and AODV. The simulation environment and details are outlined in Table 7.1.

Table 7.1: Configurations for the simulation

Simulation Settings	
Network density	~5 neighbors
Network area	2.5 km^2 to 18.5 km^2
Communication range	270 m
Number of sinks	4
Number of sensors	50, 100, 200, 300, 400
Number of generated networks	100 per scenario
Packet size	2048 bytes
Generation interval	500 ms
Deadline	100 ms
Radio type	802.11a
Transmission rate	6 Mbps
Simulation time	60 s

The test scenarios and environment were adapted from the DD solution, varying the amount of nodes in the network from 50 to 400, with four sinks in all cases. For each

scenario, 100 random networks were generated. The nodes and sinks were randomly placed, but respecting the network density. The network area varies according to the number of sensor nodes and network density. The details on the network generation process can be found in Annex A. The communication range was set to 270m as in [25] and we assume the network to be an undirected graph. The packet size was fixed to 2048 bytes, the packet generation interval was set to 500 ms, the transmission rate is 6 mbps and the deadline is 100 ms, all these parameters as in [25].

Four metrics were selected for the performance analysis. First we observe the average end-to-end delay (Latency), in order to be compared to DD and to verify the obtained improvements of our suggestions. Then, the average maximum latency, in order to compare the worst case result. Later, the number of nodes out of deadline, to identify a ratio between the network growth and the increase of nodes out of deadline. Finally, the average buffer size, to evaluate the possibility of congestion due to high packet accumulation.

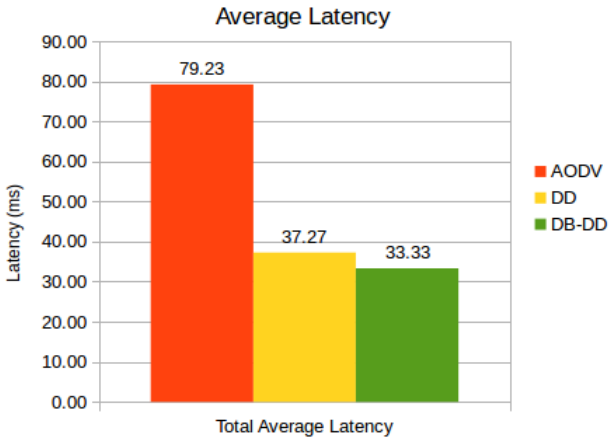


Figure 7.3: Average latency for 100 random networks composed of 50 sensor nodes and 4 sinks

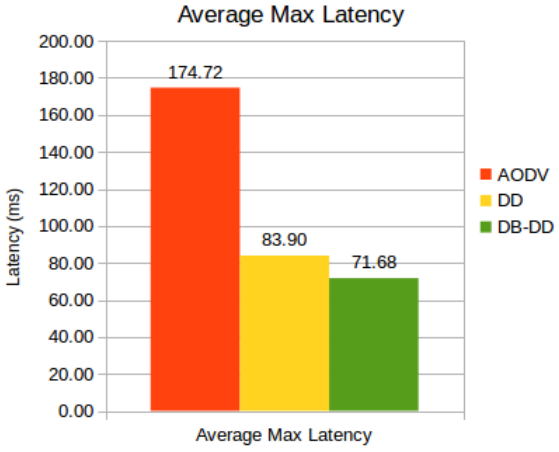


Figure 7.4: Average maximum latency for 100 random networks composed of 50 sensor nodes and 4 sinks

We start the performance analysis with the same testing scenario from the paper [25].

Since the solutions were implemented with a different simulation tool, our first objective is to assure that the obtained results are similar to the results presented in the paper. As noticed in Figure 7.3, in respect to the nodes' average latency, DB-DD outperforms both AODV and DD, showing a decrease in latency of 58% and 11% respectively. DB-DD also shows a better result regarding the average maximum latency, as presented in Figure 7.4. As in Figure 7.5, it is possible to see that DB-DD had absolutely no node out of deadline for any of the 100 networks. However, AODV presented an average of 27.64% of nodes out of deadline, and DD had 1 to 4 nodes out of deadline in 9 of the 100 networks, resulting in an average of 0.34%. These results confirm that the modifications implemented for DB-DD were able to avoid high delays and decrease the network latency.

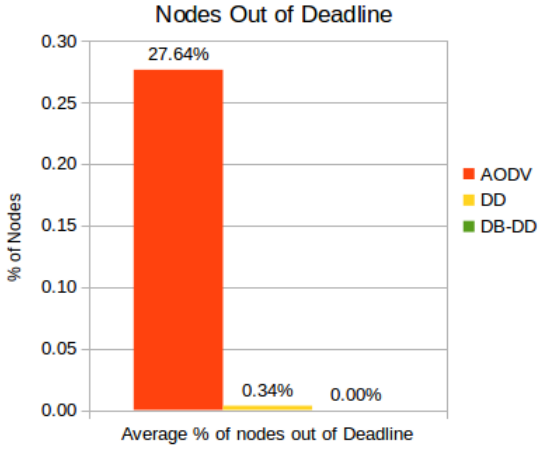


Figure 7.5: Average of the rate of out of deadline nodes for 100 random networks composed of 50 sensor nodes and 4 sinks

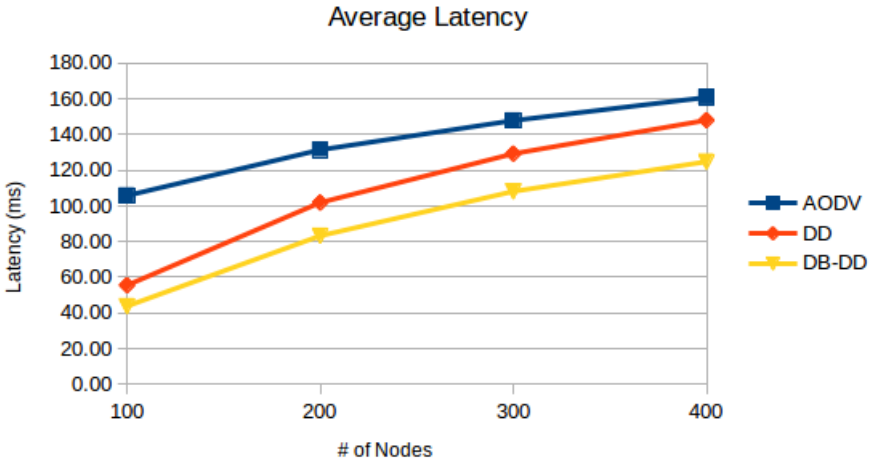


Figure 7.6: Average latency under scalability aspect - AODV, DD and DB-DD simulation for 100 random networks composed of 100 to 400 sensor nodes and 4 sinks

Subsequently, we executed the scenarios with networks composed of 4 sinks and 100 to 400 sensor nodes, in order to perform a scalability evaluation. The objective was to verify

the solution behavior when increasing the number of sensors nodes, but maintaining the same amount of sink nodes. As we can see in Figure 7.6, DB-DD presents a better behavior in terms of average latency. The increase in latency is moderately lower than DD and significantly lower than AODV. The average network latency surpasses the defined deadline at 100 sensor nodes for AODV, 200 sensor nodes for DD and only at 300 sensor nodes for DB-DD.

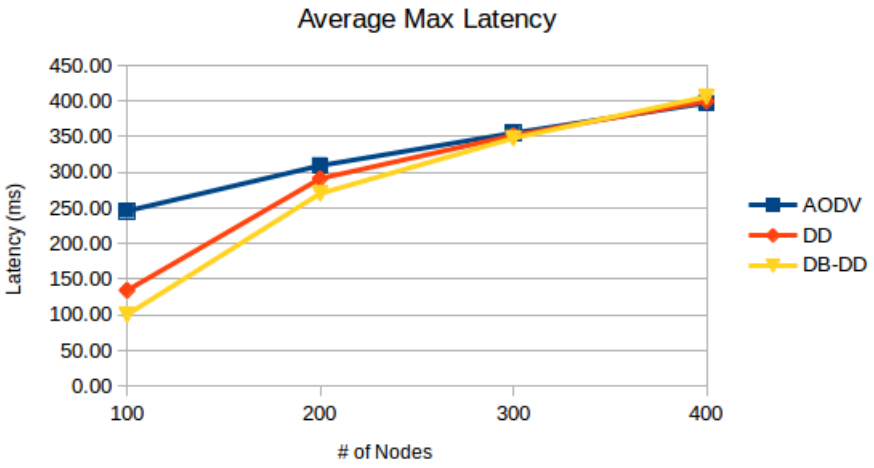


Figure 7.7: Average maximum latency under scalability aspect - AODV, DD and DB-DD simulation for 100 random networks composed of 100 to 400 sensor nodes and 4 sinks

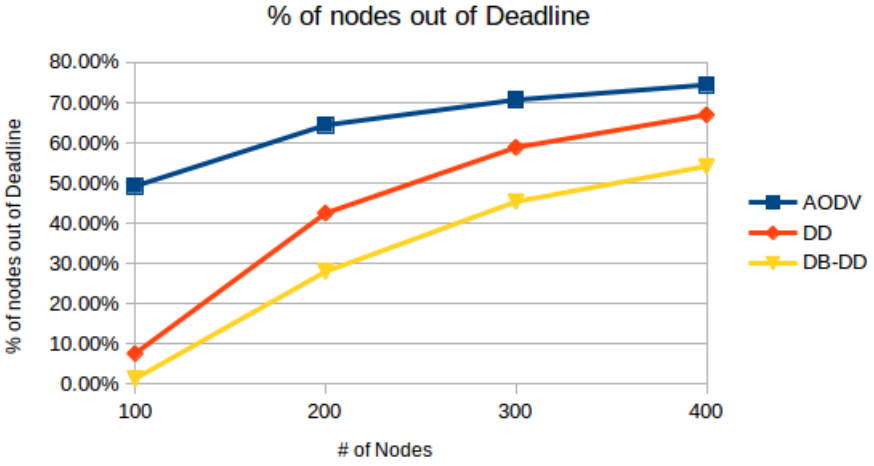


Figure 7.8: Average of the rate of out of deadline nodes under scalability aspect - AODV, DD and DB-DD simulation for 100 random networks composed of 100 to 400 sensor nodes and 4 sinks

In Figure 7.7, we can see a small difference between DD and DB-DD; even so it represents a decrease of 25% for networks with 100 sensor nodes. However, the proximity of the results in the other scenarios may indicate that the worst latency situation at each solution is similar. It may be caused by farther nodes, unable to find parents with better

node-to-sink latency. Nevertheless, when verifying the amount of nodes out of deadline in Figure 7.8, we can see that DB-DD presents a better behavior, always lower than DD by an average gap of 18%.

We also evaluated the nodes' buffer occupancy during each one of the scenarios. The target was to verify how the occupancy of the buffer evolved, in order to determine if packet drops would happen due to buffer overflow.

The results can be verified in Figure 7.9. The average maximum buffer size presented by sensor nodes in each solution is really alike and varying from approximately 120KB for 100 sensor nodes to 2600KB for 400 sensor nodes. The buffer increase shows a saturation of the network. Depending on the size of the nodes' internal measurement memory, packet loss may occur for all solutions due to buffer overflow. In order to solve this problem, we propose in Section 7.3 two additional solutions based on buffer constraint that limit the amount of stored packets and avoid buffer overflow.

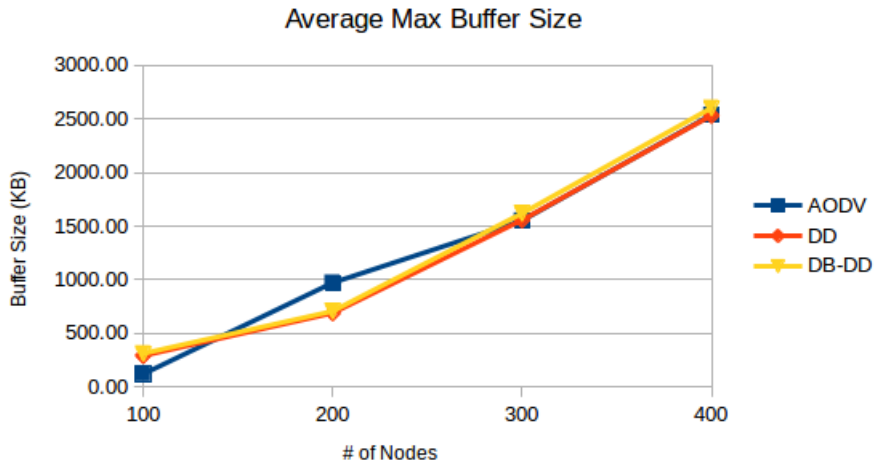


Figure 7.9: Average maximum latency under scalability aspect - AODV, DD and DB-DD simulation for 100 random networks composed of 100 to 400 sensor nodes and 4 sinks

7.3/ DB-DD WITH BUFFER CONSTRAINT

According to [102], some applications require continuous or periodical sending of information in order to maintain up-to-date data. This scenario can generate congestion, due to buffer overflow. The process of buffering information may also lead to data loss if the entire measurement memory is used and new measures are to be collected or received. Because of that it is interesting to create a buffer constraint to delimit the amount of received information stored on the node's memory.

In [102] authors state that lower buffer sizes may lead to better latency, since channel contention is reduced. However, one consequence of limiting the buffer size may be the increase in packet loss, due to packets being dropped as result of an overflow. Although this statement may be verified for AODV, it is not necessarily observed for DB-DD.

To verify the applicability of the above statement, we developed two tests considering both AODV and DB-DD algorithms. The first execution was made with the original versions of

both algorithms, in order to have a view of the nodes latency when there is no buffer constraint. The second execution intended to verify the latency when buffer constraints were applied. For that, it was necessary to implement a modification on the original algorithms. Initially, nodes should start monitoring their buffer sizes. Then, for the nodes already surpassing the predefined limit, a process of rejecting new packet incomes would start, without compromising the storage of the nodes' own generated packets. In this way it would be possible to verify the statement regarding the channel contention reduction.

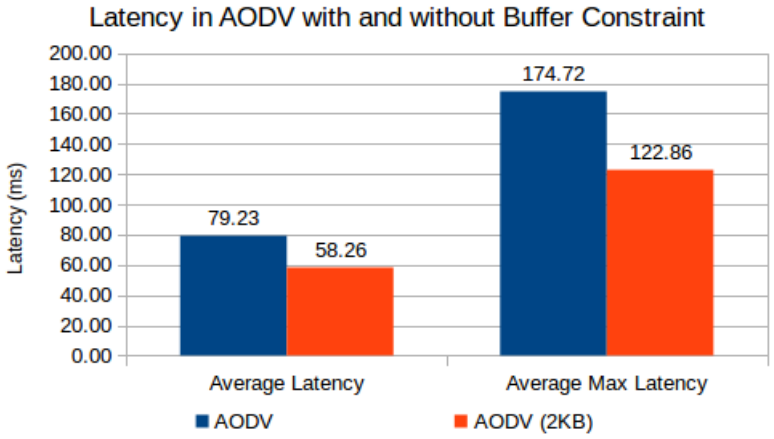


Figure 7.10: Latency in AODV without buffer constraint and with 2KB of maximum buffer size (networks of 50 sensor nodes and 4 sinks)

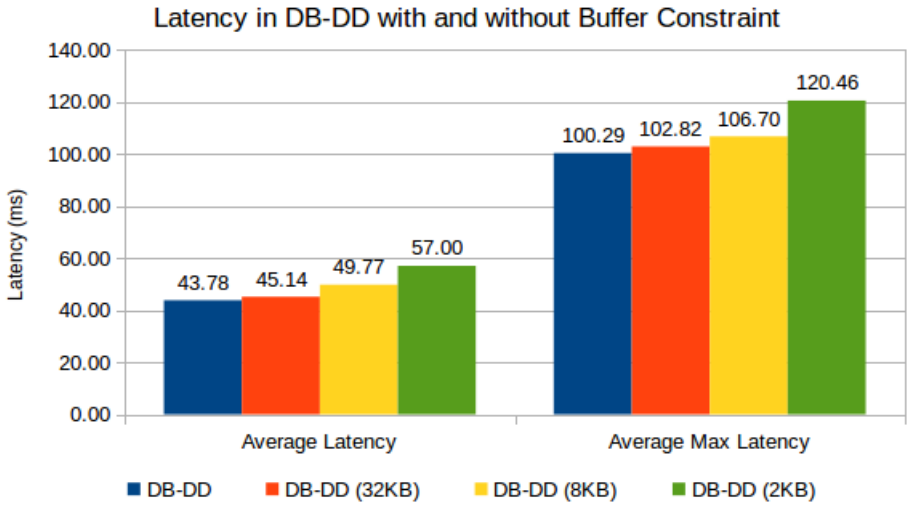


Figure 7.11: Latency in DB-DD without buffer constraint and with different buffer sizes - 2KB, 8KB and 32KB (networks of 100 sensor nodes and 4 sinks)

As expected, the test results show that the statement presented by [102] is verifiable for AODV, but not observed for DB-DD as presented in Figures 7.10 and 7.11. Channel contention is already reduced in DB-DD, due to the variable back-off time. Moreover, routes may change from time to time, because of the periodic evaluation of the nodes latency. The reduction of the buffer size in DB-DD leads to the opposite result found in

AODV. Child nodes in DB-DD with limited buffer have to wait longer periods to transmit data to their parents, triggering an increase in nodes' delay and leading to more tree reconstructions.

However, large buffer sizes is still an issue, as evidenced by Figure 7.9. Because of that, it is important to define a method to limit the buffer increase in larger networks, in order to avoid packet drops due to the lack of space in the measurements memory area. In this context, we present two more contributions: DB-DD with Fixed Buffer Constraint (DB-DD-FBC) and DB-DD with Dynamic Buffer Constraint (DB-DD-DBC).

7.3.1/ DB-DD WITH FIXED BUFFER CONSTRAINT (DB-DD-FBC)

As already presented, it is important to find a way to limit the buffer occupancy, in order to avoid packet losses. On account of that, we propose a second contribution, focused on limiting the buffer occupancy, favoring the channel access as well as prioritizing the transmission over receptions, and assuring enough buffer space to store the nodes' own generated packets.

7.3.1.1/ OPERATION

Algorithm 5 Buffer/Backoff Algorithm

Input: N: list of all sensor nodes, $Deadline$: defined deadline, F_{max}: default maximum contention window
Output: W^n_{max}: new maximum contention window for $n \in N$

for all $n \in N$ **do**
  $V_n \leftarrow GetNeighbors(n)$
  /* B list of buffer sizes */
  $B \leftarrow \emptyset$
  for all $v_i \in V_n$ **do**
    $B \leftarrow B \cup \{BufferSize(v_i)\}$
  end for
  /* $\bar{x}(B)$ represents the mean of the buffer sizes from the neighbors of n */
  if $SendTime(n) \geq Deadline$ **or** $BufferSize(n) \geq \bar{x}(B)$ **then**
    $W^n_{max} \leftarrow \frac{1}{3} F_{max}$
  else
    if $SendTime(n) \geq LocalThreshold(V_n)$ **then**
      $W^n_{max} \leftarrow \frac{2}{3} F_{max}$
    else
      $W^n_{max} \leftarrow F_{max}$
    end if
  end if
end for

DB-DD-FBC follows the same operation as DB-DD, already detailed in Subsection 7.2.3. However, an additional step is executed in order to enable the buffer limitation. The metric $BufferConstraint$ is taken into consideration during communication. If a node n needs to send a packet to its parent node p, a new verification will be performed to confirm that

the parent node p is able to receive more packets. The metric $BufferConstraint$ is a fixed value, defined during the network configuration, and it stipulates the limitation of the buffer in terms of storage. Although the buffer may be limited, the constraint is only used as a metric to deny new receptions, leaving the remaining space for the nodes' own generated packets. The act of denying new receptions automatically prioritizes the transmission, since the node will no longer be busy with incoming packets.

Another modification is related to the back-off time factor. Considering a node n and its neighbor nodes in V_n, we call the function $BufferSize(n)$ the operation which returns the node's current buffer size. The Algorithm 5 presents the adaptation to the back-off process considering the buffer constraint. The modification considers retrieving the buffer size of the neighbors of n (line 5), and evaluating the buffer size of n against the mean of the buffer sizes of its neighbors (line 9). If the buffer size of n is bigger than mean of the buffer size of its neighbors, then the back-off time is changed.

7.3.1.2/ RESULTS

The tests were executed considering the scenarios with 100, 200, 300 and 400 sensor nodes and 4 sinks. We defined the $BufferConstraint$ metric to be 32KB, since it presented the best performance compared to the execution without buffer limit, as indicated in Figure 7.11. The objective is to verify if the same improvements obtained in DB-DD would be repeated when limiting the buffer size. We compared DB-DD-FBC with AODV, DD and DB-DD.

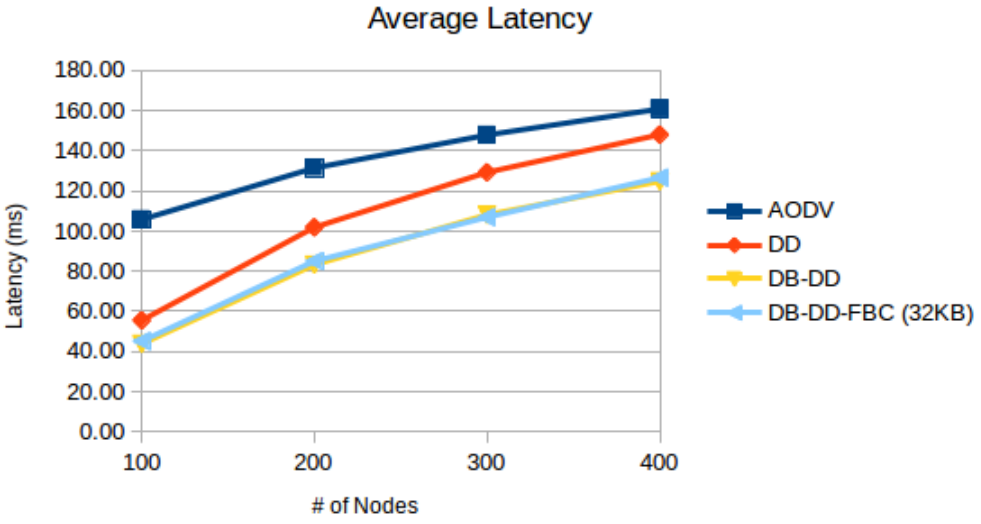


Figure 7.12: Average latency - AODV, DD, DB-DD and DB-DD-FBC simulation for 100 random networks composed of 100 to 400 sensor nodes and 4 sinks

The results show that we were able to achieve practicably the same results as DB-DD for the average latency as displayed in Figure 7.12. In terms of average maximum latency, we were able to perform as better as DB-DD for 100 sensor nodes and even better for 400 sensor nodes, with a decrease of 13% as shown in Figure 7.13. However, regarding the amount of nodes out of deadline, DB-DD-FBC performed slightly worse than DB-DD,

with a maximum difference of 4%, as presented in Figure 7.14. We suppose that this phenomenon is due to the fact that the nodes in the lower levels of the routing tree have to wait longer before being able to transmit the packets to their parent nodes.

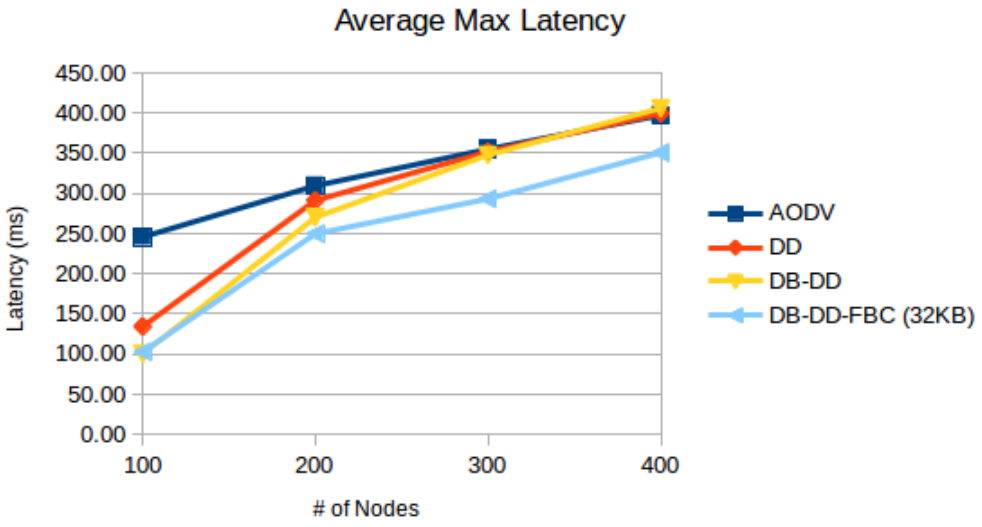


Figure 7.13: Average maximum latency - AODV, DD, DB-DD and DB-DD-FBC simulation for 100 random networks composed of 100 to 400 sensor nodes and 4 sinks

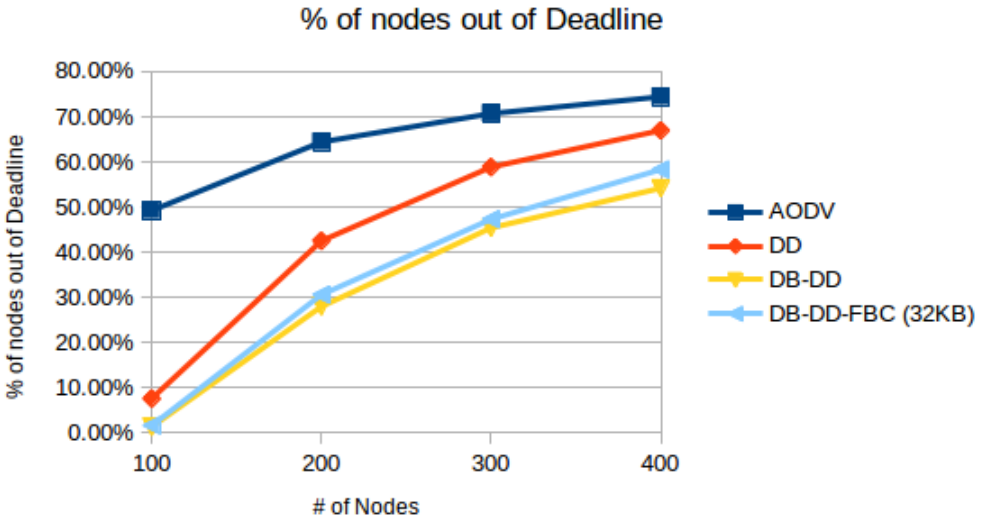


Figure 7.14: Average of the rate of out of deadline nodes - AODV, DD, DB-DD and DB-DD-FBC simulation for 100 random networks composed of 100 to 400 sensor nodes and 4 sinks

As expected, it is possible to observe in Figure 7.15 an important reduction of the buffer occupancy, with a maximum difference of 89% when compared to DB-DD. On top of that, the curve for DB-DD-FBC is fairly stable, showing a small increase when varying the network size. It means that the proposed modifications to reduce the buffer occupancy

were satisfactory, and it was possible not only to reduce the buffer occupancy, but also to avoid the problem of increasing the network latency.

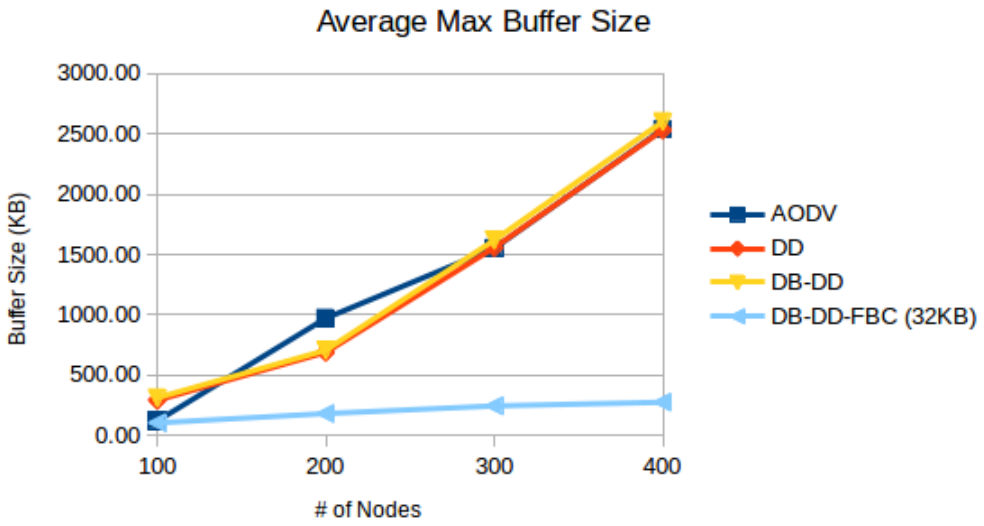


Figure 7.15: Buffer occupancy - AODV, DD, DB-DD and DB-DD-FBC simulation for 100 random networks composed of 100 to 400 sensor nodes and 4 sinks

7.3.2/ DB-DD WITH DYNAMIC BUFFER CONSTRAINT (DB-DD-DBC)

Although the solution with the fixed buffer constraint already shows good results when increasing the network size, it would be better for scalability purposes to have a dynamic constraint. It means that the increase of the network size is also followed by the constraint, providing the same performance as the DB-DD-FBC solution and maintaining the buffer occupancy in lower levels. In that sense, it was developed a third contribution, in order to offer a generic solution in terms of network size and to extend the scalability capacity.

7.3.2.1/ OPERATION

The same process implemented for DB-DD-FBC is also executed for DB-DD-DBC. The only difference comes from the fact that the $BufferConstraint$ metric is no longer a fixed value. The metric becomes a dynamic value, varying from node to node, and subject to the tree structure. The new metric is represented by $BufferConstraint(n)$.

Let us consider n the node for which the buffer constraint will be determined. The buffer limit depends on the position of the node n in the tree. If n is a leaf node, the buffer constraint is not relevant, because the node does not receive packets from other nodes. If n has a child node, then the buffer limit becomes important and it is defined as a multiple of the packet size, based on the number of descending nodes. The buffer constraint is calculated from the leaf node, up to the root of the tree. The parent node accumulates the buffer size of its child, adding it to its own limit. The general process to calculate the buffer constraint is presented in Algorithm 6. The algorithm starts by selecting the leaf nodes and retrieving the parent node of each leaf node. The parent node of each leaf

node is set with a buffer limit based on the number of child nodes. If all child nodes of the current parent are covered, the parent node is marked as visited and its parent node is selected as the next node to be visited.

Algorithm 6 Buffer Limit Algorithm

Input: N: list of all sensor nodes, S list of all sink nodes
Output: $BufferConstraint(n)$: buffer constraint for n, $\forall\ n \in N$

```
1:  /* C current list of nodes to be visited */
2:  C ← ∅
3:  for all n ∈ N do
4:      BufferConstraint(n) ← PacketSize
5:      if GetChildren(n) = ∅ then
6:          C ← C ∪ {n}
7:      end if
8:  end for
9:  /* V list of visited nodes */
10: V ← C
11: /* T next list of nodes to be visited */
12: T ← ∅
13: while C ≠ ∅ do
14:     for all c_i ∈ C do
15:         p ← GetParent(c_i)
16:         if p ∉ S then
17:             BufferConstraint(p) ← BufferConstraint(p) + BufferConstraint(c_i)
18:             if p ∉ T and GetChildren(p) ⊂ V then
19:                 T ← T ∪ {p}
20:             end if
21:         end if
22:     end for
23:     V ← V ∪ T
24:     C ← T
25:     T ← ∅
26: end while
```

The process of calculating the $BufferConstraint(n)$ is executed only when the routing tree is reconstructed. If a leaf node changes its parent, then the $BufferConstraint(n)$ metric must be updated for the entire new and old branches.

7.3.2.2/ RESULTS

Similarly to the tests executed for DB-DD-FBC, we considered the scenarios with 100, 200, 300 and 400 sensor nodes and 4 sinks once again to validate this new proposal. We defined $PacketSize$ as 2KB, since it represents the size of a single packet in all previous simulations. The objective this time was to verify if DB-DD-DBC behaved better then DB-DD and DB-DD-FBC in terms of latency, at the same time keeping the buffer occupancy low.

The results show that DB-DD-DBC was able to perform slightly better than DB-DD and DB-DD-FBC for 400 sensor node networks, with a decrease in latency of 4% in com-

parison to DB-DD-FBC, as presented in Figure 7.16. Regarding the average maximum latency, DB-DD-DBC performed even better, reaching a decrease of 20% in comparison to DB-DD, as noticed in Figure 7.17. The weak point in DB-DD-DBC was found in the average number of nodes out of deadline, where it performed marginally worse than DB-DD, with a difference of 2%, and slightly better than DB-DD-FBC, with a difference of 2%, as presented in Figure 7.18. This behavior is due to the fact that the nodes in the lower levels of the routing tree have to wait longer before being able to transmit the packets to their parent nodes.

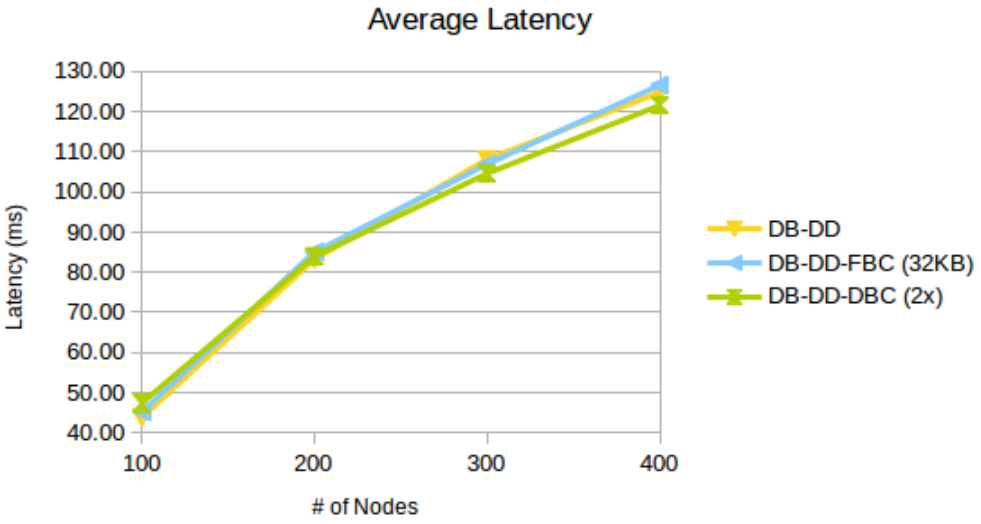


Figure 7.16: Average latency for DB-DD, DB-DD-FBC and DB-DD-DBC simulation for 100 random networks composed of 100 to 400 sensor nodes and 4 sinks

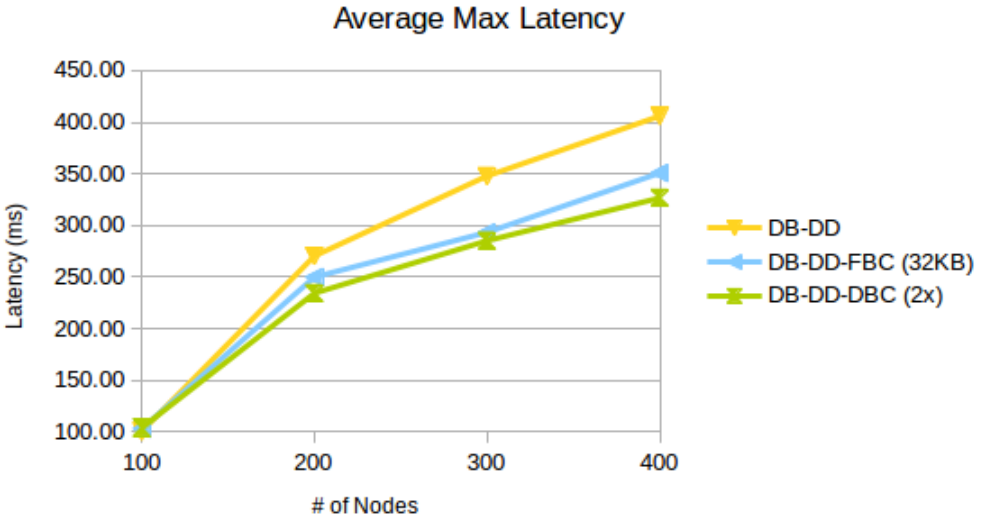


Figure 7.17: Buffer size for DB-DD, DB-DD-FBC and DB-DD-DBC simulation for 100 random networks composed of 100 to 400 sensor nodes and 4 sinks

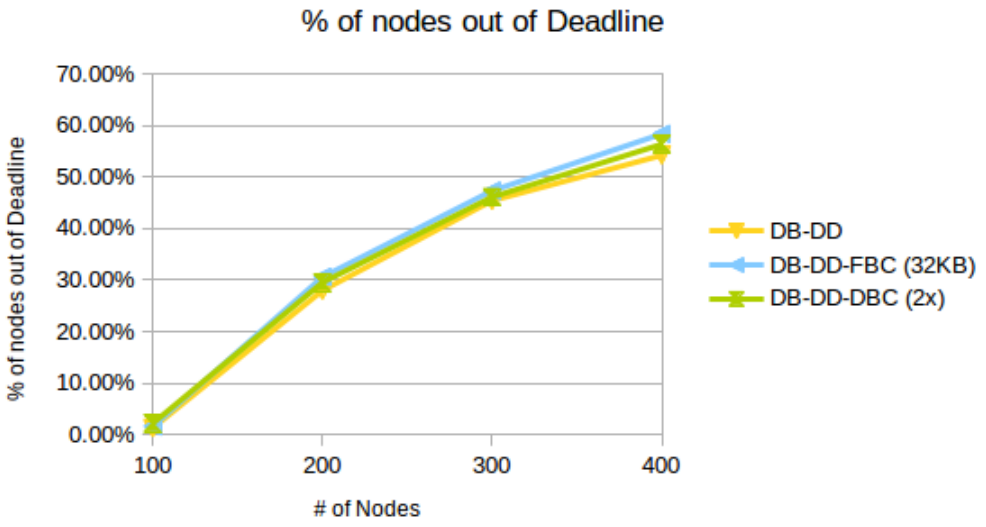


Figure 7.18: Average of the rate of out of deadline nodes for DB-DD, DB-DD-FBC and DB-DD-DBC simulation for 100 random networks composed of 100 to 400 sensor nodes and 4 sinks

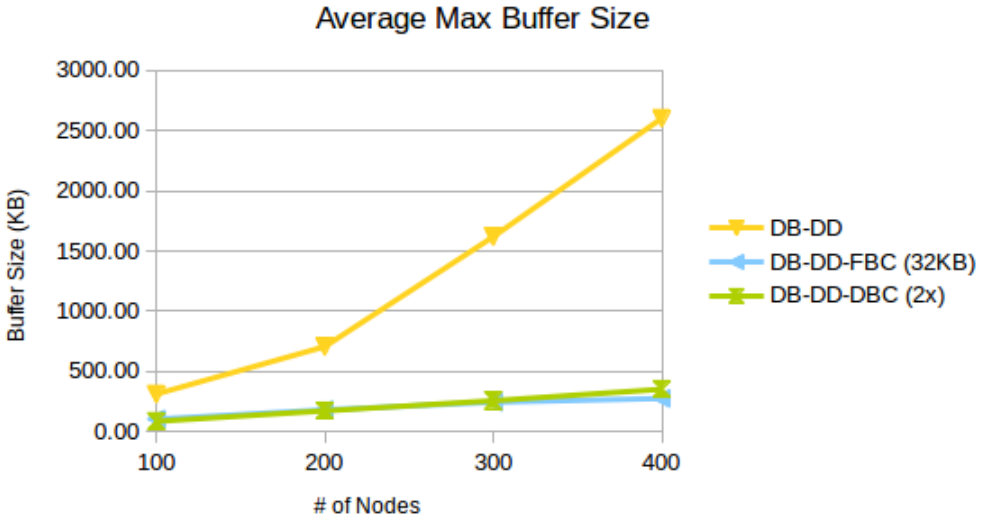


Figure 7.19: Buffer occupancy for DB-DD, DB-DD-FBC and DB-DD-DBC simulation for 100 random networks composed of 100 to 400 sensor nodes and 4 sinks

In terms of buffer occupancy, DB-DD-DBC performed largely better than DB-DD, and with a reasonably stable curve with the increase of the network size. However, for 400 sensor node networks, DB-DD-DBC started to degrade in performance when compared to DB-DD-FBC, reaching a difference of 21%, as shown in Figure 7.19. This behavior was expected, since DB-DD-DBC has a dynamic constraint influenced by the amount of nodes of a tree. When the network increases in size, the $BufferConstraint(n)$ also increases, especially for nodes on the top of the tree. Nevertheless, it is possible to say that in

terms of buffer occupancy DB-DD-DBC is as stable as DB-DD-FBC, keeping the buffer occupancy low. Besides, it outperforms DB-DD-FBC on all the other evaluated metrics. Thus, it is plausible to say that DB-DD-DBC has a better performance than DB-DD-FBC.

7.4/ CONCLUSION

In this chapter we focused on presenting solutions considering unicast communication scheme, with a strategy of selecting the forwarding path based on the transmission delays, aiming at the minimization of the latency, at the same time as enabling the increase of the network size. Three proposals were presented, evaluated through simulations and finally compared to existing solutions, with results showing improvements in terms of network latency and buffer occupancy. The presented solutions can be summarized as centralized, MAC-aware routing protocols, for time sensitive applications, in Multi-Sink WSN, with packets being routed through low latency paths in a unicast fashion, and with a preventive method to avoid high latency. Additionally, two of the solutions also presented mechanisms for buffer control.

The obtained simulation results show that our suggestions produced improvements in terms of end-to-end delay. For the tiny network size (50 nodes), DB-DD has no nodes out of deadline, differently from AODV that presented an average of 27.64% of the nodes out of deadline, and DD that presented an average of 0.34% of nodes out of deadline. In terms of average latency, DB-DD presented a performance gain of approximately 58% compared to AODV and 11% compared to DD. When we increase the network size, with scenarios considering medium-sized networks (100 to 400 nodes) and the same number of sink nodes, the average number of nodes out of deadline increases in all solutions. However, DB-DD presents fewer nodes out of deadline in all scenarios, compared to AODV and DD. For the scenario with 100 sensor nodes, DB-DD presents 1.29% of the nodes out of deadline, AODV has 49.24% of the nodes out of deadline and DD has 7.56%.

The increase in the network size with a fixed number of sinks also increases the number of buffered packets. In a real network, the nodes have a limit for the buffering memory, and when this limit is reached, packets are dropped. In order to avoid dropping the packets, we proposed a strategy of buffer occupancy control, limiting the number of stored packets and increasing the priority of the packet transmission. The results show that is was possible to reduce the average buffer occupancy with a small degradation of the average latency performance. For the scenario with 100 sensor nodes, the performance gain in relation to DD for the average latency is 21.12% for DB-DD, 18.66% for DB-DD-FBC and 14.67% for DB-DD-DBC. However, the maximum buffer occupancy was $312KB$ for DB-DD, $104KB$ for DB-DD-FBC and $85KB$ for DB-DD-DBC.

In summary, it was possible to reduce the average latency and the number of nodes out of deadline when the network increases with a fixed number of sinks. We were able to limit the buffer occupancy, without dropping the packets, and with a small degradation of the latency results.

8

ANYCAST

Contents

In anycast communication scheme the information is addressed to any of the sinks. The decision towards which destination sink the packet will be forwarded is taken on the way, based on the routing path. A particular case of the anycast communication scheme is the k-anycast. In this case, the packet must be forwarded to any k different destinations, with k representing the number of sinks to be reached ($k \leq |S|$). Since the destination sinks may be in disjoint paths, it also implies the need of packet duplication at some point of the routing process.

An anycast solution may be the strategy to respond the many-to-one and many-to-many demands of an applications. The particularity is that the target sink is not predefined. An application of forest fire detection may be an example where k-anycast may be applied. The sensor nodes are spread in a large area, normally with difficult access, which makes the maintenance of nodes a real challenge. In case of a fire, the warning must the raised and effectively delivered to the concerned authority. With anycast, it is possible to set a number of targets, increasing the chances of delivering the packet, without flooding the network. Since the sinks are not predefined, the routes may change and packet may be delivered to different sinks at each transmission. Another example of application is a tracking system in a warehouse. Sensor nodes read RFID tags from the stored goods, and send the data to any of the sinks. Based on the position of the source node, the sink

is capable of identifying the position of the goods. There is no need to address the packet to a specific sink.

In this chapter we focus our effort on the definition of a k-anycast routing protocol, capable of forwarding packets to exactly k different sinks, at the same time as reducing the latency and increasing the network lifetime.

8.1/ MOTIVATION

The k-anycat is the most flexible communication scheme in MS-WSN. A k-anycast routing protocol may behave as 1-anycast routing solution when $k = 1$ or even as a multicast routing protocol when k equals the number of all sinks. However, the complexity of the algorithm increases, especially when the target sinks and packet duplication are decided on the fly, during the packet forwarding. In these cases, the packet is not addressed to one specific sink, but to a group of sinks, and the final destination is decided on the way based on the network metrics at each hop.

The number of works existing dedicated to k-anycast routing is limited. Most of the anycast routing solutions are designed as 1-anycast routing protocols (see Section 3.1). Nevertheless, a 1-anycast routing solution could be used as k-anycast if we assume that packets are duplicated at source and forwarded in sequence. The problem with this strategy is that there is no guarantee that k different sinks will be reached. Depending on the forwarding criteria, chances are that the k packets will be forwarded to the exactly same sink. In that sense, k-anycast solutions are essentially different from 1-anycast protocols, with added complexity and special objectives.

To the best of our knowledge, only three works in literature consider k-anycast communication scheme (already described in Subsection 3.2.3). The routing solutions [54] and [55] are based on the hop-count distance to the sinks. This strategy implies either higher network topology knowledge or an important setup phase, with nodes discovering their hop-count distance to each sink. The work in [56] is a geographic routing protocol, which requires a lesser level of network knowledge, compared to the hop-count based routing protocols. It is focused on reducing the energy consumption, while assuring the packet forwarding to exactly k sinks. However, the routing decision is based on the energy cost of the transmission, which is a fixed value. It means that the routes towards the same group of sinks are always the same, which increases the energy consumption of the nodes involved in the forwarding process. This may eventually lead to an early depletion of the nodes' battery.

In this chapter, we propose GeoK, a k-anycast geographic routing algorithm, capable of assuring the forwarding of packets to exactly k different sinks, focused on network lifetime and latency optimizations. Our algorithm incorporates several techniques to reduce both the latency and the maximum energy consumption. The next hop decision is taken using a linear combination of network metrics, along with dynamic weights. The metrics weights vary according to the number of targeted sinks. Because of that, the paths are constantly changed during the lifetime of the network, which helps distributing the energy consumption. Both duplication and target sinks are decided on the fly. Since our algorithm is based on geographic routing and greedy forwarding, the existence of void areas may be expected (local minimum). Because of that, we also combine and adapt two different void handling techniques, making use of the right-hand rule and the passive participation

[103]. Finally, when packets must be duplicated, we benefit from MAC-level information to create an ordered sending list based on the duty-cycle schedule of the neighbor nodes participating in the forwarding task.

8.1.1/ SYSTEM MODEL AND ASSUMPTIONS

Similarly to the model presented in 7.1.1, we represent a MS-WSN as a graph $G = (V, E)$, where V represents the set of all nodes and E the set of existing wireless links. Each $e \in E$ corresponds to a pair of nodes (i, j) as long as i and j, with $i \in V$ and $j \in V$, are within each other's transmission range (we consider here that all links are symmetric). The set of neighbors of a node i is denoted by V_i. We also specify that $V = S \cup N$ where S represents the set of sink nodes and N the set of sensor nodes, with $S \cap N = \emptyset$.

Specifically to this chapter, we have that the number of sinks to be reached is denoted by k, with $0 < k \leq |S|$. Additionally, we assume that every node is aware of its own geographic position and the geographic position of all sinks. For simplification, the geographic positions are represented by the cartesian coordinates (x, y), and the distance is always the euclidean distance represented as $|ij|$ with $i, j \in V$. We also assume that packets are generated by sensor nodes only, and that k is given and remains constant for each packet generation. For the energy consumption, we follow the radio model in [59], defining the consumed energy during transmission:

$$E_{T_x} = E_{elec} \times p + \epsilon_{amp} \times p \times d^2 \tag{8.1}$$

and the consumed energy during reception:

$$E_{R_x} = E_{elec} \times p \tag{8.2}$$

where E_{elec} is the dissipated energy for the transmitter/receiver electronics (we assume it to be $50\ nJ/bit$ as in [59]), ϵ_{amp} is the dissipated energy for the transmit amplifier (we assume it to be $100\ pJ/bit/m^2$ as in [59]), p is the amount of bits of the message and d is the transmission distance.

We consider two types of networks: with void areas and without void areas. A network with void areas is defined by the existence of a specific node out of the transmission range of a particular sink and for which there is no other neighbor node that presents a better geographic progress towards that sink than the node itself. In summary, $\exists\{n, s_i\}, n \in V, s_i \in S, (n, s_i) \notin E$ such that $|ns_i| < |v_j s_i|$ for all $v_j \in V_n$. In a MS-WSN it is possible for a node to be in the void area of one or more sinks. The amount of void areas in a network is a fixed number based on a particular node and the set of sinks. However, since k sinks must be reached, the probability of entering a void area increases with a higher k. It is because the algorithm has to forcibly select a sink in a void area, if all the other sinks were already selected.

For a packet being forwarded from a source node towards a sink, we say that the next hop offers a positive progress towards the sink if it is geographically closer to the sink in relation to the current node.

8.1.2/ KANGUROU

KanGuRou is the work proposed in [56]. It is a geographic routing algorithm, which means that it requires little knowledge on the network topology. Because of the similar assumptions, we selected KanGuRou as the reference solution for the comparison with the results of our solution.

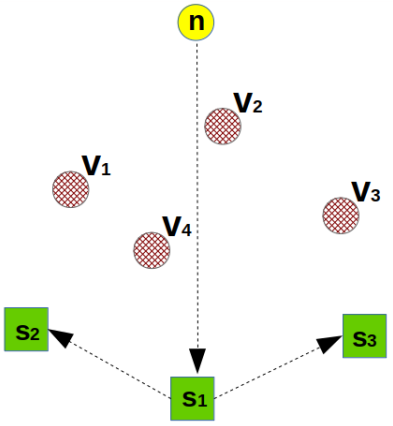


Figure 8.1: Spanning tree towards $k = 3$ sinks

The main objective in KanGuRou is to guarantee the packet forwarding to exactly k sinks and at the same time to reduce the overall energy consumption. The strategy considers a geographic routing towards a set of sinks. The path is constructed in a greedy way. The current node builds a spanning tree to k sinks, based on the cost over progress, which considers a trade-off between the geographic progress and the energy cost of the transmission, as displayed in Figure 8.1. Since the current node knows the existence and the position of all sinks, it calculates a spanning tree having exactly k sinks. At this point, the algorithm decides if a duplication will be necessary, based on the number of branches of the spanning tree from the current node towards the sinks.

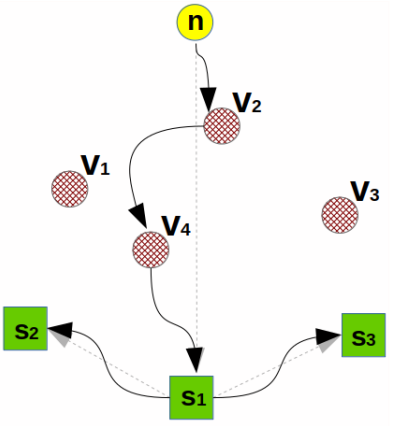


Figure 8.2: Next hop decision based on the previous spanning tree

The next hop is also decided based on the cost over progress towards each branch of the spanning tree having the current node and the k sinks. The solution assumes an adaptable transmission range in order to reduce the energy cost. In the example of

Figure 8.2, the nodes v_1, v_2, v_3 and v_4 are within the communication range of the node n (with maximum radio power). Instead of sending the packet directly to node v_4, the node n decreases its radio power and send the packet to node v_2. The node v_2, at its turn, also decreases its radio power, in order to send the packet to node v_4. Finally, node v_4 sends the packet to the sink s_1.

The decision process is repeated at each node. The algorithm calculates the cost over progress and decides to duplicate the packet and send it through different paths in order to reach k sinks. When a packet is duplicated, it is targeted to a set of specific sinks, in order to assure the reception by k sinks.

During the process of searching a suitable forwarder towards the groups of sinks, the KanGuRou protocol may not find a valid node with a positive progress. This is the case of network void areas, which are inherent to geographic routing. KanGuRou handles the existence of voids using a recovery mode with face routing [104], so packets can be forwarded out of the problematic area, as exemplified in Figure 8.3. In face routing, a planar graph is partitioned into faces with a traversal line having the current node at one point and the destination node at the other point. The packet travels through the faces using the right-hand rule until the point it finds a node which offers a positive progress towards the destination. The right-hand rule consists of an analogy to a technique used to exit a maze. The idea is to place the right hand (or the left) on the wall and walk through the maze until the exit is found.

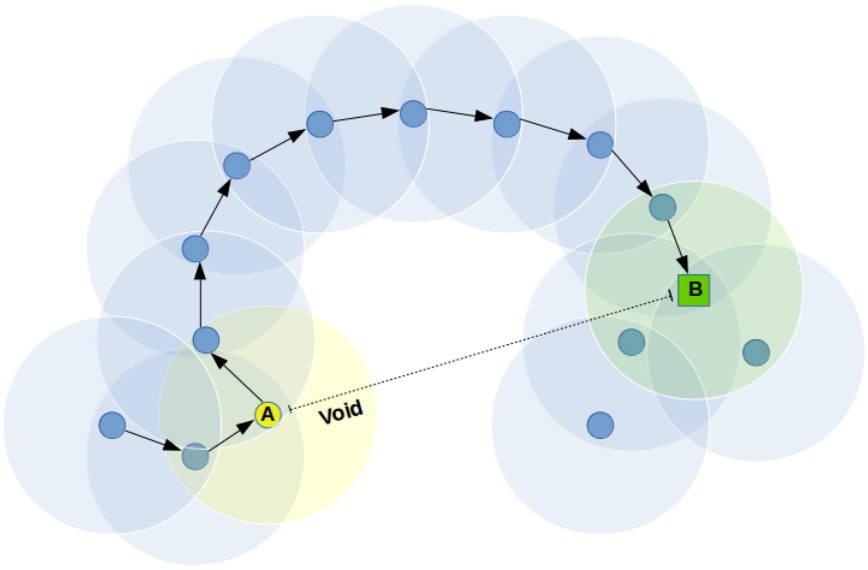


Figure 8.3: Packet being forwarded from node A to node B around the void area

KanGuRou is focused on reducing the energy consumption, favoring transmissions to closer nodes. Since the transmission range is variable, the energy cost to transmit a packet to a closer node is reduced. However, the number of hops is increased and consequently the latency. Moreover, KanGuRou does not consider the neighbor nodes during the selection of the target sinks. This behavior may lead to an early duplication of the packets, as well as the forwarding of packets into void areas. There is no preventive measure to avoid void areas.

The authors in [56] implemented and tested KanGuRou thorugh simulations using the

WSNet [93] simulator and C programming language. KanGuRou was compared to another solution called EEGDA [51], which is a 1-anycast solution focused on reducing the energy consumption.

8.2/ GEOGRAPHIC K-ANYCAST ROUTING (GEOK)

We propose GeoK, a geographic routing protocol for multi-sink wireless sensor networks that assures the delivery of k packets to exactly k sinks using a k-anycast communication scheme, focused on reducing the overall latency and maximizing the network lifetime. The next hop is selected by the current node based on the calculation of weighted metrics. The selection of the target sinks is done based on the ordered list of energy cost to reach each of the available sinks.

The algorithm for the next hop decision can be divided into five steps with a preprocessing at the beginning of the algorithm, followed by the candidates filtering, the forwarders selection, the remaining sink distribution and the actual routing at the end.

- The preprocessing is responsible for retrieving the packet information and triggering the GeoK processing. Commonly, during the packet routing, relevant information (source address, target destination, etc.) are inserted in the packet as a header in order to help the forwarding process. At each hop, the information in the packet is updated. The number of targeted sinks may change due to the decisions of the next hop. In that sense, the current value of k is denoted as k_{curr} and the set of available sinks is denoted as S_a.
- The second processing step is responsible for filtering the neighbor nodes, in order to create a list of candidate forwarders. The filtering takes place by eliminating the neighbor nodes that are not able to offer a positive progress towards the sink, and the neighbor nodes in void areas. If no candidate forwarder is found, the recovery mode is triggered and a neighbor is selected using a void handling technique.
- The third processing step is dedicated to effectively selecting the forwarders. The candidates are evaluated based on the weighted metrics, and a forwarder is selected for exactly k_{curr} sinks.
- The fourth processing step is triggered only if $k_{curr} < |S_a|$, and it is responsible for distributing the remaining sinks throughout the selected forwarders, based on their position and the position of the sinks.
- The actual routing is responsible for the packet transmission. If multiple forwarders were selected, a packet duplication takes place. Each forwarder receives a copy of the packet with a different header. The header contains the set of target sinks and the new value of k.

We define here the concept of *positive progress* towards a set of sinks for a multi-sink geographic routing protocol. In contrast to the concept of progress towards one sink already presented in Subsection 8.1.1, the progress of a node towards a group of sinks considers a tree structure rooted at the current node and having all the sinks attached to the tree. In that sense, to calculate its progress, the current node builds a spanning tree composed of only itself and all the target sinks. For the construction of the spanning

tree, the current node considers the geographic distances towards the sinks. The first selected sink is the closest sink to the current node. The edge connecting the two nodes is represented by the euclidean distance between them. All other sinks are added to the tree based on their distances towards the closest node in the tree, with each edge labeled with the corresponding euclidean distance. The progress value for a node is the inverse of the sum of all euclidean distances of the spanning tree. A progress for a node is called positive if it is superior to the progress of the previous node from which the packet was received.

8.2.1/ PREPROCESSING

Once a node has a packet (either generated or received) to be forwarded, it triggers the execution of the preprocessing, described in Algorithm 7. The first step is to check if the current node is a sink itself and if it is in the list of target sinks (line 4). If the current node is one of the target sinks, the k_{curr} value must be decremented, and the current node must be removed from the list of available sinks (lines 5 and 6). If k_{curr} is still greater than zero, the multi-hop process continues with the candidate filtering process (line 9).

Algorithm 7 $[F, p_{new}, packet] = MultiHop(n, packet, V_n, H_s^n)$

Input: n: current node, $packet$: packet to be forwarded, V_n: the set of neighbors of n, H_s^n: set of neighbor nodes of n in a void area for the sink s

Output: F: structure with the set of forwarders with the respective sinks and k (neighbor node, [sinks], k), p_{new}: progress offered by the current node towards S_a, $packet$: packet to be forwarded

1: $S_a \leftarrow GetSinks(packet)$ /* Set of target sinks */
2: $k_{curr} \leftarrow GetK(packet)$ /* Current k */
3: $p_{prev} \leftarrow GetProgress(packet)$ /* The progress offered by the previous hop */
4: **if** $n \in S_a$ **then**
5: $\quad k_{curr} \leftarrow k_{curr} - 1$
6: $\quad S_a \leftarrow S_a \backslash \{n\}$
7: **end if**
8: **if** $k_{curr} \geq 1$ **then**
9: $\quad [F, p_{new}] = GeoK(n, k_{curr}, S_a, V_n, H_s^n, p_{prev})$
10: **end if**
11: **return** $[F, p_{new}, packet]$

8.2.2/ CANDIDATES FILTERING

The candidate filtering step is responsible for creating a set of candidate forwarders (L). The set can be composed entirely of candidates with positive progress, or of candidates issued from the recovery mode, or a mix of both. Since the algorithm has to find a suitable forwarder towards k sinks, it is possible that for networks with void areas, only part of the k can be covered by candidates with positive progress. Because we need to assure the delivery to exactly k sinks, it is necessary to complete the candidate set with other nodes, which are the neighbor nodes issued from the recovery mode. The recovery mode is the process of selecting a candidate forwarder with the face routing technique. The filtering process is described in Algorithm 8, explained afterwards.

The filtering starts by checking if the current node n offers a positive progress towards the set of available sinks (S_a) (line 4). If the current node does not offer a positive progress,

it means that the packet was in recovery mode and no positive progress has yet been found. In this case the packet must keep the recovery mode status and the next hop is selected using the face routing technique [103]. The recovery algorithm follows the same principles of the works in [105], [51] and [56]. If the current node effectively offers a positive progress, the normal candidate selection is started.

Only the neighbor nodes with a positive progress to at least one sink are selected. The algorithm excludes the neighbors that have already announced being in a void area (line 8). The values in H_s^n represent the neighbor nodes of n in a void area for the sink s.

Algorithm 8 $[F, p_{new}] = GeoK(n, k_{curr}, S_a, V_n, H, p_{prev})$

Input: n: current node, k_{curr}: current number of sinks to be reached, S_a: set of available sinks, V_n: set of neighbors of n, H_s^n: set of neighbor nodes of n in a void area for the sink s, p_{prev}: progress offered by the previous hop towards S_a

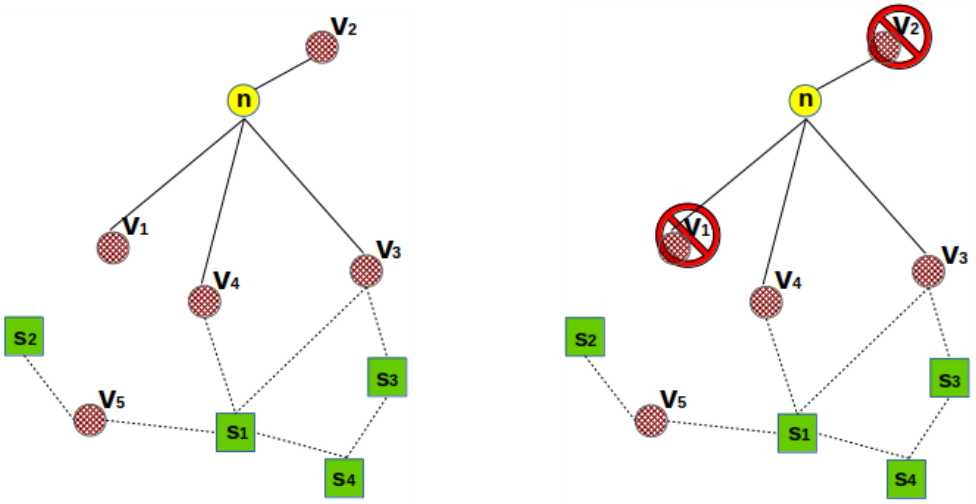
Output: F: list with the set of forwarders with the respective sinks and k, p_{curr}: progress offered by the current node towards S_a

```
1:  F ← ∅
2:  L ← ∅ /* Set of pairs [candidate, sink] */
3:  p_new ← ProgressTowards(n, S_a)
4:  if p_new > p_prev then
5:     S' ← ∅ /* Set of found sinks */
6:     for all v_j ∈ V_n do
7:        for all s_i ∈ S_a do
8:           if dist(n, s_i) > dist(v_j, s_i) and v_j ∉ H^n_{s_i} then
9:              L ← L ∪ {[v_j, s_i]}
10:             if s_i ∉ S' then
11:                S' ← S' ∪ {s_i}
12:             end if
13:          end if
14:       end for
15:    end for
16:    k' ← |S'|
17:    if k' < k_curr then
18:       /* Lack of candidates, void handling is triggered */
19:       if k' > 0 then
20:          F ← Forwarders(n, L, S', k') /* Select the forwarders among the found candidates */
21:          k_curr ← k_curr − k'
22:       end if
23:       S' ← S_a \ S'
24:       SendVoidNotification(n, V_n, S') /* Notify neighbors about the existence of a void area */
25:       L = Recovery(V_n, S', k_curr) /* Start Recovery mode using right-hand rule */
26:    end if
27: else
28:    /* Current node does not offer a positive progress */
29:    /* Start recovery mode using face routing */
30:    L = Recovery(V_n, S_a, k_curr)
31:    p_new ← p_prev
32: end if
33: F ← F ∪ Forwarders(n, L, S_a, k_curr)
34: return [F, p_new]
```

The algorithm creates a second set of sinks, with the sinks for which a neighbor with positive progress was detected (line 11). The size of this set represents the maximum possible number of attainable sinks, represented by k'. If k' is smaller than the k_{curr}, it

means that it was not possible to find a positive progress towards all necessary sinks (line 19). In this case, the algorithm tries to create a preliminary list of forwarders for the found sinks and neighbors (Algorithm 8, line 20 - detailed in Algorithm 9), and completes the list with the candidates issued from the recovery mode (line 25). At the same time, the void announcement is triggered (line 24) for the set of sinks the algorithm cannot find a suitable neighbor candidate. The void announcement contains a list of sinks for which the current node is in a void area. The announcement is periodically broadcasted in order to inform the neighbor nodes about the void area. Each node receiving the broadcasted message updates its own H_s^v set with the information from n.

Let us consider an example. The network represented in Figure 8.4a has six sensor nodes (n, v_1 to v_5) and four sinks (s_1 to s_4). The nodes v_1, v_2, v_3 and v_4 are direct neighbors of the node n. The node n must send a packet to three sinks ($k = 3$). As previously described, the filtering process eliminates the nodes with no positive progress towards the sinks (v_2 in relation to all sinks and $v3$ in relation to sink s_2) and the nodes knowingly in void areas (v_1). The final list of nodes and sinks is ($[v_4, s_1]$; $[v_4, s_2]$; $[v_4, s_3]$; $[v_4, s_4]$; $[v_3, s_1]$; $[v_3, s_3]$; $[v_3, s_4]$).



(a) Initial state (b) Filtered neighbors

Figure 8.4: Filtering process (solid lines represent neighbor connections and dotted lines represent paths)

8.2.3/ FORWARDERS SELECTION

The forwarders selection stage is focused on selecting the most suitable forwarders (the F list in Algorithm 8) based on the decision metrics (distance, consumed energy and duplication avoidance) and the defined weights for each metric. Each element of F is a structure that represents a different forwarder for the packet. It means that if $|F| > 1$ a duplication takes place. The structure in F comprises the neighbor node responsible for the forwarding task, the list of target sinks and the new k value. The list of target sinks may contain exactly k sinks or more. It depends on the amount of available sinks S_a in comparison to k_{curr} and the proximity of the already selected target sinks (S_{used}) in relation to the ones not yet selected (S''). The distribution of the remaining sinks (S'') is detailed in Subsection 8.2.4.

The target sinks must be chosen in a way to reduce the energy consumption. To cope with that, the solution creates an ordered list of sinks based on the energy cost of the transmission (Algorithm 9, line 5). The first element in the list is the sinks to which the energy cost of the transmission from the current node is the smallest. The subsequent elements are the sinks selected based on the minimum energy cost of the transmission from either to the current node or an already selected sinks.

Once the sink list is created, the algorithm starts the process of searching for the most suitable forwarder. The set of forwarder candidates having the sink s_i as target is extracted from L (algorithm 9, line 8), and the selection of the best forwarder v out of them is started (Algorithm 9, line 9) - the selection process being described in Algorithm 10.

Algorithm 9 $F = Forwarders(n, L, S_a, k_{curr})$

Input: n: current node, L: set of candidate forwarders with respective sinks, S_a: set of available sinks, k_{curr}: number of sinks to be reached
Output: F: list with the set of forwarders with the respective sinks and k

1: $F \leftarrow \emptyset$
2: $S_{used} \leftarrow \emptyset$ /* Set of selected target sinks */
3: $pk \leftarrow k_{curr}/|S_a|$ /* Ratio between k_{curr} and the amount of available sinks S_a */
4: /* S': set of sinks ordered by energy cost considering the transmission from the n */
5: $S' \leftarrow OrderSinksByEnergyCost(n, S_a)$
6: **for all** $s_i \in S' | i \leq k_{curr}$ **do**
7: $S_{used} \leftarrow S_{used} \cup \{s_i\}$
8: $R \leftarrow \{v_j \mid [v_j, s_i] \in L\}$
9: $v \leftarrow SelectForwarder(n, R, F, s_i, pk)$
10: $f \leftarrow find(F, v)$ /* returns the index of v in F or -1 if nonexistent */
11: **if** $f < 0$ **then**
12: $f \leftarrow |F|$
13: **end if**
14: $F[f].neighbor \leftarrow v$
15: $F[f].sinks \leftarrow F[f].sinks \cup \{s_i\}$
16: $F[f].k \leftarrow F[f].k + 1$
17: **end for**
18: /* S'': set of other possible target sinks */
19: $S'' \leftarrow S' \backslash S_{used}$
20: **if** $S'' \neq \emptyset$ **then**
21: /* Distributes the remaining sinks through the selected forwarders based on the energy cost from a used sink to an available sink */
22: $F = DistributeRemainingSinks(F, S'')$
23: **end if**
24: **return** F

The selection of the most suitable forwarder is executed using the decision metrics and their respective weights. The weights vary according to the ratio between the k_{curr} and the amount of available sinks $|S_a|$ (Algorithm 9, line 3). Initially, the value of the metrics for each candidate is computed (Algorithm 10, lines 5 to 14). Each metric has a different objective. As for instance, the distance metric (α) is focused on selecting the candidate with the highest positive progress towards the target sink. This is important in order to reduce the overall hop count and consequently the latency. The consumed energy metric (β) regards the selection of the node with the smallest total energy consumption, which in time balances the energy consumption in the neighborhood of the current node, prolonging the network lifetime. The total energy consumption is advertised along with other parameters (geographical position and void areas) in the broadcast messages. Finally, the

duplication avoidance (δ) accounts for both latency and energy consumption, since the increase of packets in the network not only intensifies the energy consumption, but also multiplies the possibility of congestion and delays. As already mentioned, the weights for each metric are dynamic and adjustable depending on the situation. For a scenario where k is much smaller than S, only few replications may be triggered, and the focus must be on progressing towards the closest sinks, so the weights for the distance metric and energy consumption are higher. On the other hand, if $k = |S|$ the objective changes, and avoiding replications becomes more important.

The aggregated decision metric ω is calculated using the relative values of the metrics from each candidate (Algorithm 10, lines 15 to 20). The candidate node having the smallest ω is selected as forwarder (Algorithm 10, line 21). If the selected forwarded v is already part of the forwarders list F, it means that v is already a forwarder to a different sink. In this case, the k value of the existing entry is incremented and the target sink s_i is added to the list of sinks of the forwarder v. This is important in order to avoid the packet duplication. If v is not yet part of the F list, a new entry is created with the new forwarder, which ultimately triggers a duplication (Algorithm 9, lines 10 to 16).

Algorithm 10 $v = SelectForwarder(n, R, F, s, pk)$

Input: n: current node, R: set of candidate neighbors, F: list with the set of forwarders with the respective sinks and k, s: current target sink, pk: value for the selection of the weight
Output: v: neighbor node selected as forwarder
1: /* $\alpha(pk)$: weight for the distance metric */
2: /* $\beta(pk)$: weight for the consumed energy metric */
3: /* $\delta(pk)$: weight for the duplication avoidance metric */
4: /* Weights vary according to pk ($k_{curr}/|S_a|$) */
5: **for all** $v_j \in R$ **do**
6: $D[v_j] \leftarrow dist(v_j, s)$
7: $E[v_j] \leftarrow ConsumedEnergy(v_j)$ /* Total consumed energy of the neighbor node v_j */
8: /* Check if v_j is already a forwarder */
9: **if** $find(F, v_j) = -1$ **then**
10: $G[v_j] \leftarrow 1$
11: **else**
12: $G[v_j] \leftarrow 0$
13: **end if**
14: **end for**
15: **for all** $v_j \in R$ **do**
16: $x \leftarrow \alpha(pk) \times \frac{D[v_j]-Min(D)}{Max(D)-Min(D)}$
17: $y \leftarrow \beta(pk) \times \frac{E[v_j]-Min(E)}{Max(E)-Min(E)}$
18: $z \leftarrow \delta(pk) \times G[v_j]$
19: $\omega[v_j] = x + y + z$
20: **end for**
21: $v \leftarrow v_j | \omega[v_j] = Min(\omega)$
22: **return** v

The forwarders selection process is illustrated in Figure 8.5 considering the same network from the example in Figure 8.4. The ordered list of sinks contains (s_3, s_4, s_1, s_2). The sink s_3 is the closest to the current node n, consequently it presents the smallest transmission energy cost in relation to the node n. The sink s_4 is the second in the list, since it presents the smallest energy cost in relation to s_3, that is already part of the list. Sinks s_1 and s_2 are added following the same process. Let us suppose that n needs to send a packet to three sinks ($k = 3$). The evaluation process starts with sink s_3, since it is the first sink in

the ordered list. The algorithm starts searching for a suitable forwarder. The weighted metric is calculated for v_4 and v_3, having s_3 as target sink. The node v_3 is selected and associated to s_3. The process is repeated, and the sink s_4 is selected from the ordered list of sinks. After evaluating the weighted metrics for nodes v_3 and v_4, the candidate node v_3 is selected again. Because it is already a forwarder for a different sink, the node v_3 had an extra weight in the evaluation, improving its final ω score. The process is repeated one more time in order to have the $k = 3$ sinks covered. The sink s_1 is selected and the candidates are evaluated. The neighbor node v_3 is selected as the final forwarder for all sinks.

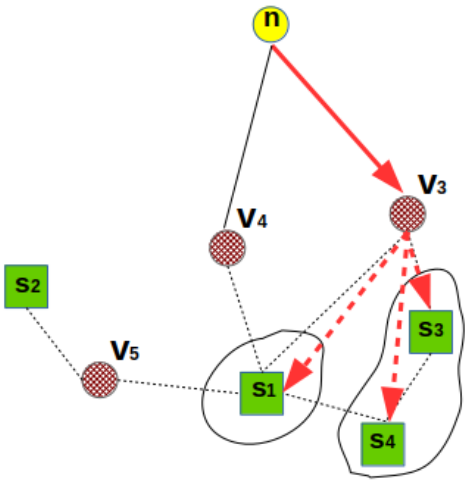


Figure 8.5: Forwarders selection process

8.2.4/ SINK DISTRIBUTION

Since the algorithm follows an anycast communication scheme, the destination sink is not fixed. If the k value is smaller than the number of sinks in the network $k < |S|$, the destination sink may change during the packet forwarding.

In our algorithm, the forwarders selection process makes use of a particular sink as destination to decide on the most suitable forwarder. However, it does not mean that it is forcibly the sink that will receive the packet. The destination sink may change during the packet routing due to an encounter with a void area or a final weighted metric that forces the packet to go in a different way. Nevertheless, the change is only possible if $|S_a| > k_{curr}$.

After the forwarders selection, not all sinks are used as destinations $S_a \backslash S_{used} \neq \emptyset$. The remaining sinks must be distributed over the forwarders in F. This is important in order to allow the packet to change its final destination. The distribution process considers the same principle of the sink list ordering, assigning each of the remaining sinks to the forwarder with the sinks that present the smallest energy cost of the transmission in relation to the remaining sink.

For a remaining sink $s_r \in S_a \backslash S_{used}$, the algorithm calculates the transmission energy cost E_{tx} from s_r to both the forwarder $f_i.neighbor$ ($f_i \in F$) and each of the sinks in $f_i.sinks$. Then, the sink s_r is included in the sink list of the f_i that presents the lowest E_{tx}.

Following the example of Figure 8.6, s_2 is the remaining sink that must be allocated as a possible destination. The process of distributing the remaining sinks considers the

proximity of an already selected sink. In the example, sink s_1 is the closest to s_2, so the sink s_2 is associated to the packet that will be forwarded towards s_1.

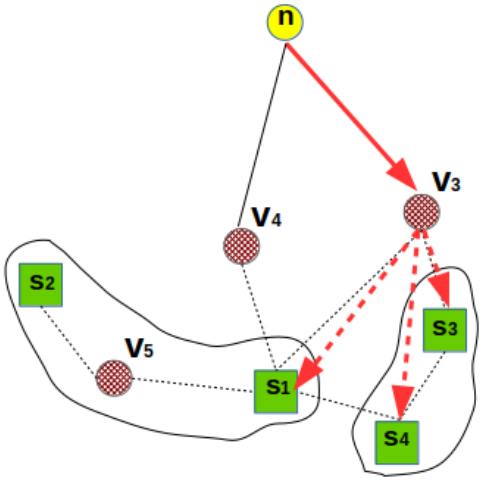
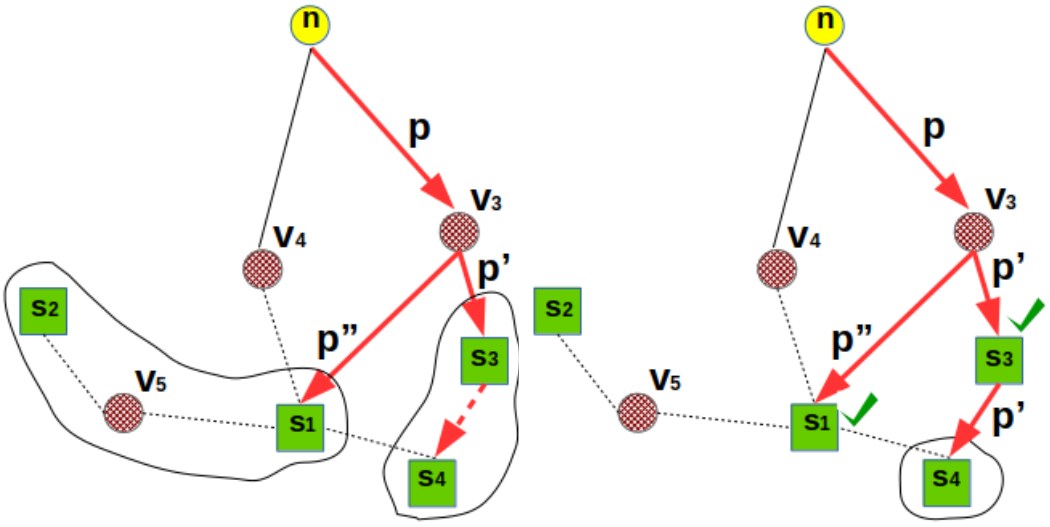


Figure 8.6: Sink distribution process

8.2.5/ ROUTING

The actual routing takes place after the forwarders selection, by the end of Algorithm 7. During the actual routing, the packet is duplicated if $|F| > 1$, and it is updated with the forwarder address, the data of the new list of available sinks and the corresponding k. Also as part of the routing process, and for the case a duplication is necessary, the sending order is decided based on the duty cycle of the selected forwarders. The first packet to be sent is the one of the first forwarder to wake-up. The algorithm makes use of the duty-cycle schedule information from the neighbor nodes to determine the encounter moment in order to define the sending order.



(a) Initial state (b) Filtered neighbors

Figure 8.7: Packet duplication and routing

In Figure 8.7 we have the simulation of the last part of the algorithm. The node n sends the packet p to the selected forwarder v_3 having the sinks (s_3, s_4, s_1, s_2) and $k = 3$. The entire process is re-executed at node v_3, which duplicates the packet p, sending the replica p' towards (s_3, s_4) with $k = 2$ and the replica p'' towards (s_1, s_2) with $k = 1$. Finally, the process is executed in sink s_3, that forwards the packet p' towards s_4 with $k = 1$.

8.3/ SIMULATION AND RESULTS

GeoK protocol was developed in C as an application module for Contiki OS [6] and evaluated through simulations using Cooja [6]. The performance of our solution was compared to an existing approach (KanGuRou) [56], that was adapted to Contiki OS and tested with Cooja under the same configurations. The original version of KanGuRou was implemented and tested with WSNet [93]. The simulation environment and details are outlined in Table 8.1.

Table 8.1: Configuration for the simulations

Simulation Settings		
Deployment density (d)	8 neighbors (on average)	
Network area (variable)	$\frac{\pi \times r^2}{d} \times \lvert V \rvert$ (8.3), ($\lvert V \rvert = \lvert N \rvert + \lvert S \rvert$)	
Communication range (r)	50 m	
Packet generation rate	20% chance at every minute for each sensor	
Radio type	802.15.4	
MAC protocol	CX-MAC, modified version of [106]	
Execution time	120 minutes for each network	
Weight values (α, β, δ)	see Table 8.2	
	Scalability test	
	Proportional	Not proportional
Number of sensors ($\lvert N \rvert$)	50, 100, 150, 200, 250, 300	100, 500, 1000
Number of sinks ($\lvert S \rvert$)	10% of the number of sensors	10 sinks
k	20% to 100% of the sinks	10 sinks
Number of networks	300 with voids, 300 without voids	300 without voids
Packet size	240 bytes	100 bytes

Table 8.2: Weight values

	$pk \leq 0.4$	$0.4 < pk \leq 0.8$	$pk > 0.8$
α	0.3	0.2	0.1
β	0.1	0.1	0.1
δ	0.6	0.7	0.8

By equation 8.3 in Table 8.1, the network deployment density changes with the size of the network when the size of deployment area is kept the same. Since we want to keep a similar deployment density over all variations of $|V|$, we make the network area vary with the number of deployed nodes.

As presented in Algorithm 10, the weight values change in relation to the ratio of k_{curr} and the number of available sinks (S_a). Table 8.2 presents the weight values for the sim-

ulations. We obtained the weight values and the intervals based on experimentation and the analysis of the algorithm behavior in relation to each metric (described in Subsection 8.2.3). As for instance, an elevated ratio $k_{curr}/|S_a|$ means that the packet has a limited possibility of changing the final destination. In this scenario, a duplication is more likely to happen in order to allow the packet to go through disjoint paths. Because of that, we increase the weight for the metric concerning the duplication avoidance (δ).

The solution performance is evaluated by observing the average latency, defined by the average time a packet takes to be routed from the source to each of the destination sinks, and the maximum energy consumption, which gives an indication of the network lifetime. As per explanation, we consider a network to be alive as long as all nodes have some energy. Therefore, network lifetime is considered to be the earliest moment at which a node's battery is completely depleted.

The simulation execution considered two main network topologies: with void areas and without void areas. The simulation outcome is given likewise considering three series of results. The first one fixes the number of sensors and sinks, and varies the values of k, in order to evaluate the performance of the solution when k increases, reaching $k = |S|$. For the second one, the relative k value is fixed and the number of sensors and sinks varies, in order to evaluate the proportional scalability, which states on the behavior of the protocol under a proportional increase of the number of sinks compared to the network size. The last series of results consider the increase of the network size with a fixed number of sinks and $k = |S|$. The objective is check the performance of the solution in terms of scalability without proportion. The number of source nodes is increased, creating more traffic towards a fixed number of sinks.

8.3.1/ Fixed number of sensors and sinks

In this group of results we are interested on analyzing the performance of GeoK in respect to the effects of increasing k in relation to the number of sinks. Networks have 300 sensor nodes and the number of sinks represents 10% of the number of sensors. The values of k varies as a step of 20% of the number of sinks.

8.3.1.1/ Networks without void areas

For the scenario with no void areas and a fixed number of nodes and sinks, we can see in Figure 8.8 that GeoK shows a better latency performance in relation to KanGuRou, with a growing gain with the increase of k. Although for a small k the options of sinks are better, with nodes being able to choose the closest sinks, the performance of the two solutions are relatively close, with a gain of approximately 4% in favor of GeoK. When k becomes larger, the options of sinks become limited, and the packet must be forwarded to sinks that are much farther. In these scenarios, GeoK shows an improvement of over 10% in relation to KanGuRou. It is explained by the fact that for farther sinks our algorithm is able to find better routes, performing fewer hops to reach distant sinks, as in Figure 8.9.

Regarding the maximum energy consumption, we can see in Figure 8.10 that GeoK has also a better performance when compared to KanGuRou. We can notice a maximum gain of almost 30% and a minimum of approximately 13%. That is explained by the nature of our solution that is designed in a way to use different routes during the protocol execution.

It means that the energy consumption is distributed, which reduces the chances of nodes depleting their batteries in early stages.

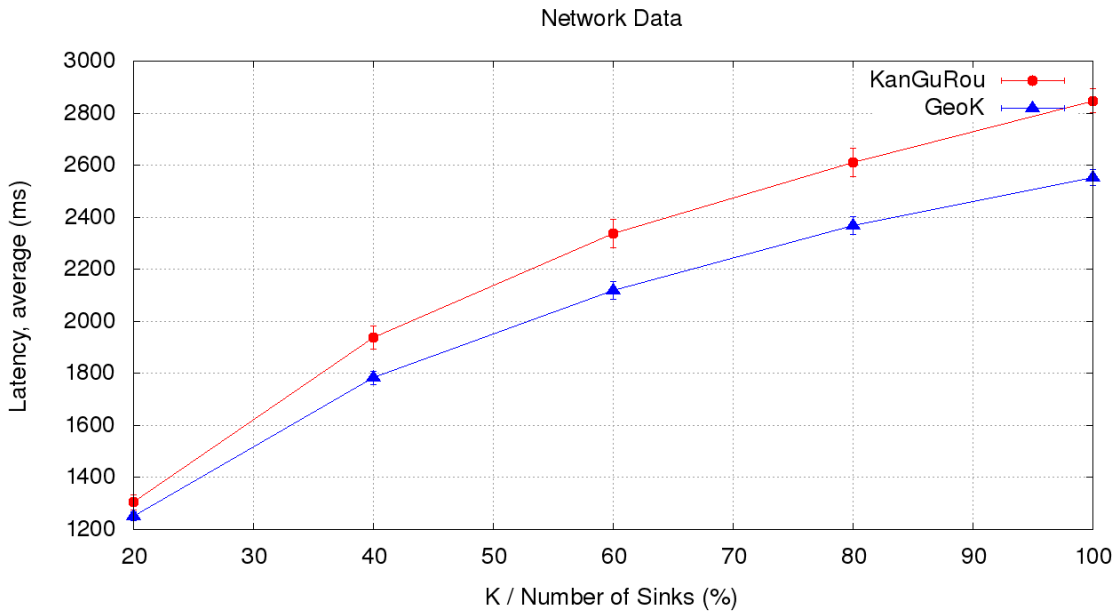


Figure 8.8: Average packet latency - no void areas, 300 sensors, 30 sinks and k varying with a step of 20% of the number of sinks

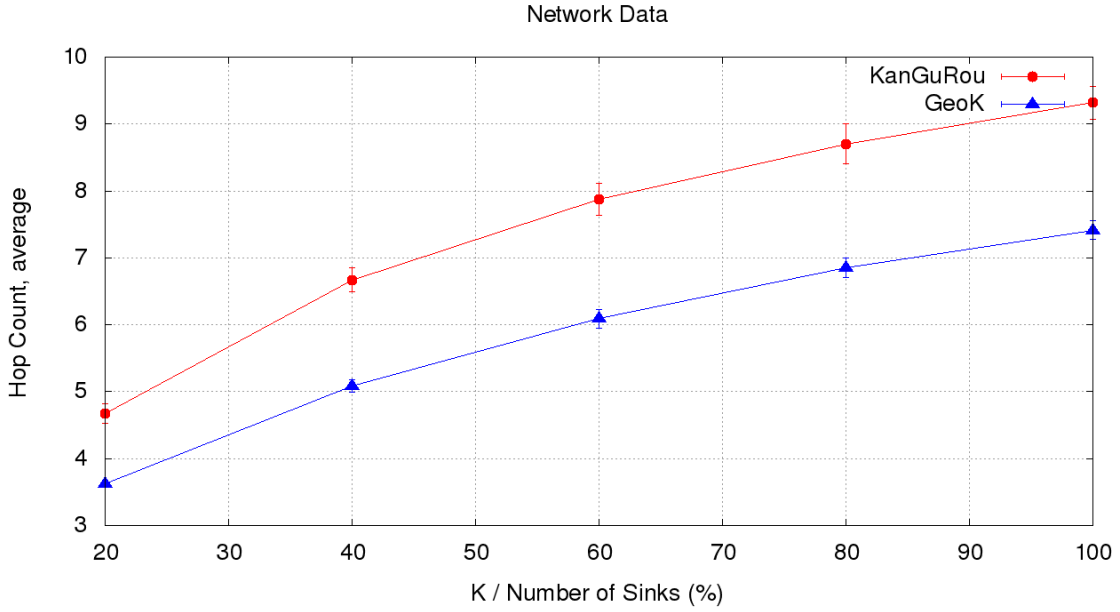


Figure 8.9: Average hop count - no void areas, 300 sensors, 30 sinks and k varying with a step of 20% of the number of sinks

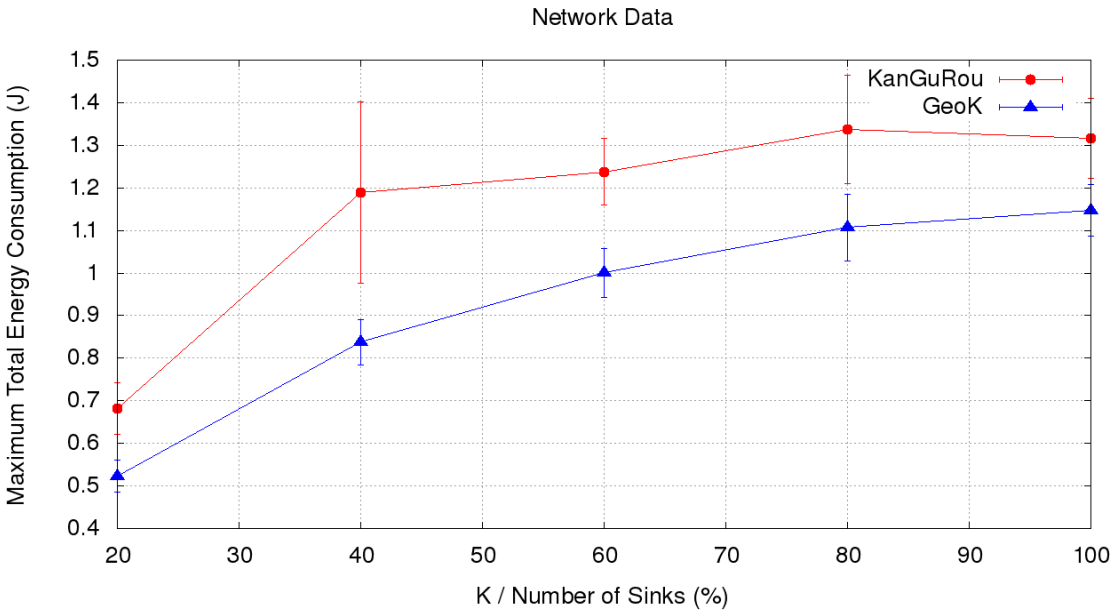


Figure 8.10: Maximum energy consumption - no void areas, 300 sensors, 30 sinks and k varying with a step of 20% of the number of sinks

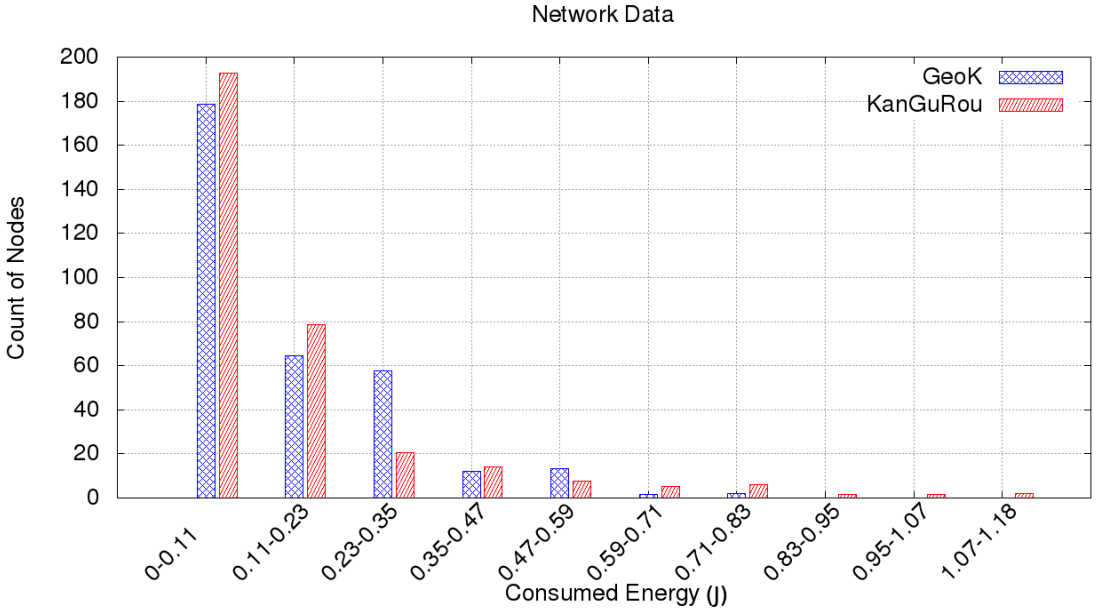


Figure 8.11: Distribution of nodes in terms of consumed energy - no void areas, 300 sensors, 30 sinks and $k = 40\%$ of the number of sinks

We can confirm that observation with Figure 8.11, that shows the distribution of nodes based on the consumed energy at the end of the simulation. We can notice that KanGuRou has some nodes with higher consumption levels. It is a consequence of the fact that KanGuRou always uses the same paths for the packet forwarding, which over time depletes the energy of the same nodes. This characteristic is also observed in Figure

8.12, that shows the energy consumption over time of the node having spent the maximum energy at the end of the simulation.

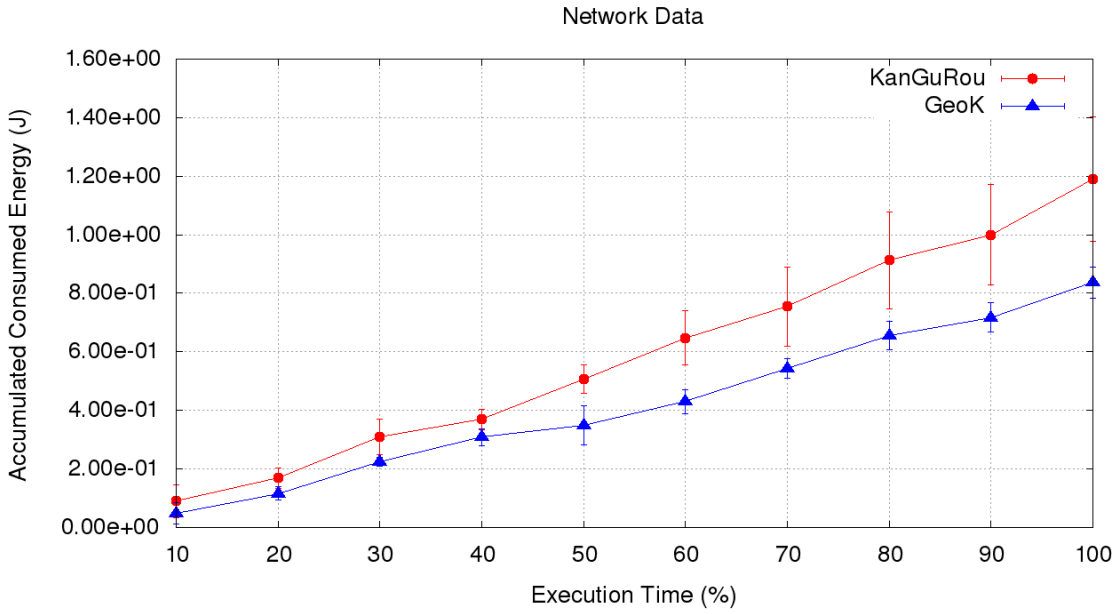


Figure 8.12: Energy consumption over time for the node that consumed the maximum energy at the end of the simulation - no void areas, 300 sensors, 30 sinks and $k = 40\%$ of the number of sinks

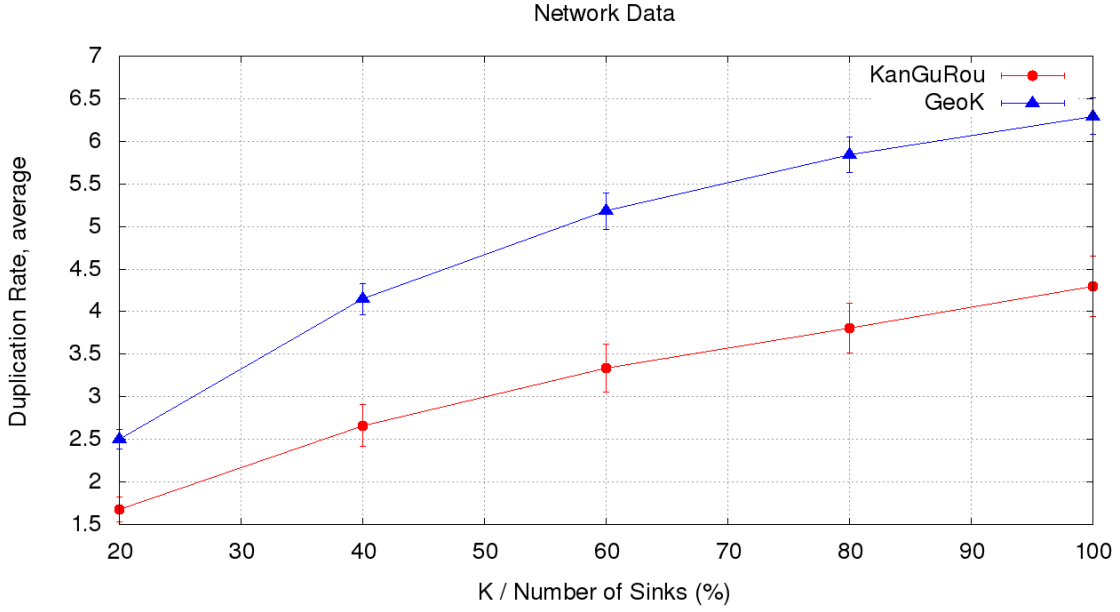


Figure 8.13: Average duplication - no void areas, 300 sensors, 30 sinks and k varying with a step of 20% of the number of sinks

Contrarily to the average latency, the performance gain of the maximum energy consumption eases with the increase of k. It is justified by the increase of packets being

forwarded. When k is larger, it means that at some point the packet is duplicated. Since there are more packets circulating in the network, the energy consumption raises and the performance decreases. In average, packet duplication is slightly higher with GeoK when compared to KanGuRou, as displayed in Figure 8.13. In order to have a faster progress towards a sink, GeoK is constrained to diverge the path in an earlier stage of the forwarding process, so replications take place more frequently.

8.3.1.2/ NETWORKS WITH VOID AREAS

When networks with void areas are considered for the same number of sensors and sinks, we can also notice a good performance for GeoK. For the latency, we can see in Figure 8.14 that GeoK has even better results when compared to KanGuRou. That is explained by the void announcement strategy. Since nodes in void areas advertise on their situation, the neighbor nodes may avoid forwarding packets through the problematic area, resulting in better delivery time. However, the performance gain decreases with the increase of k. That happens because the possibility of entering in a void area grows with the increment of k. Packets must enter in recovery mode more frequently, which increases the hop count. Nevertheless, similarly to the case without void areas, the average hop count for GeoK is smaller than KanGuRou as presented in Figure 8.15, resulting in lower latencies. GeoK presents a maximum gain in latency of almost 13% and a minimum of approximately 8%.

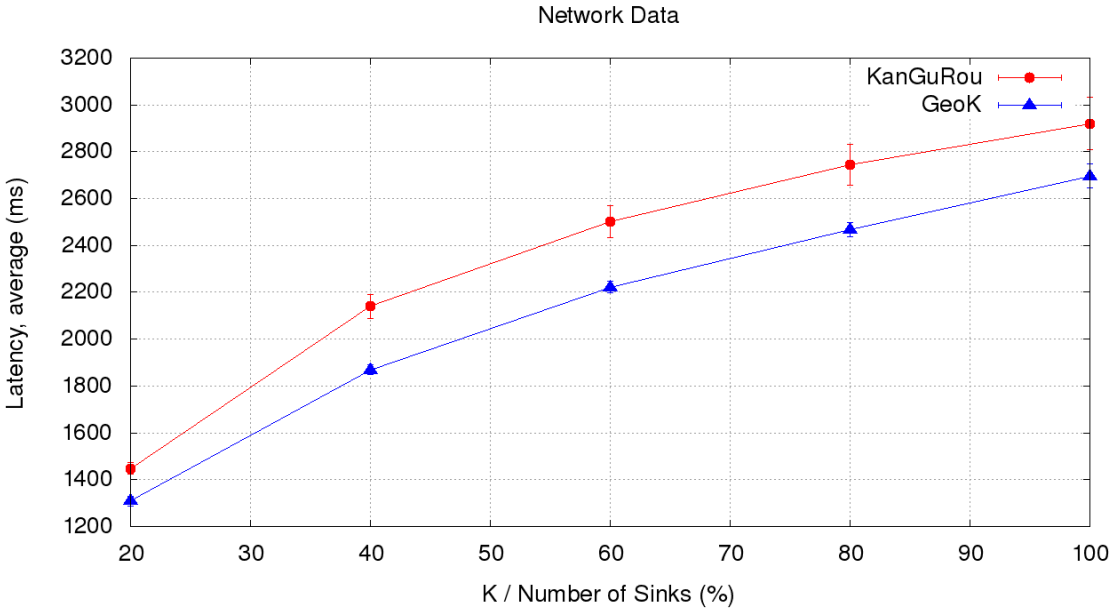


Figure 8.14: Average packet latency - with void areas, 300 sensors, 30 sinks and k varying with a step of 20% of the number of sinks

In terms of maximum energy consumption, GeoK also performed better than KanGuRou, with a maximum gain of almost 26% and a minimum of approximately 7%, as displayed in Figure 8.16. The same tendency observed in the scenario without void areas is noticed when void areas are present. The gain is reduced with the increase of k. Once again, it can be justified by the fact that more replications are necessary, increasing the amount

of packets in the network. When a void area is detected for a specific sink, the node may decide to split the packet through disjoint paths, so the void area is bypassed. We can notice in Figure 8.17 that the duplication rate increases.

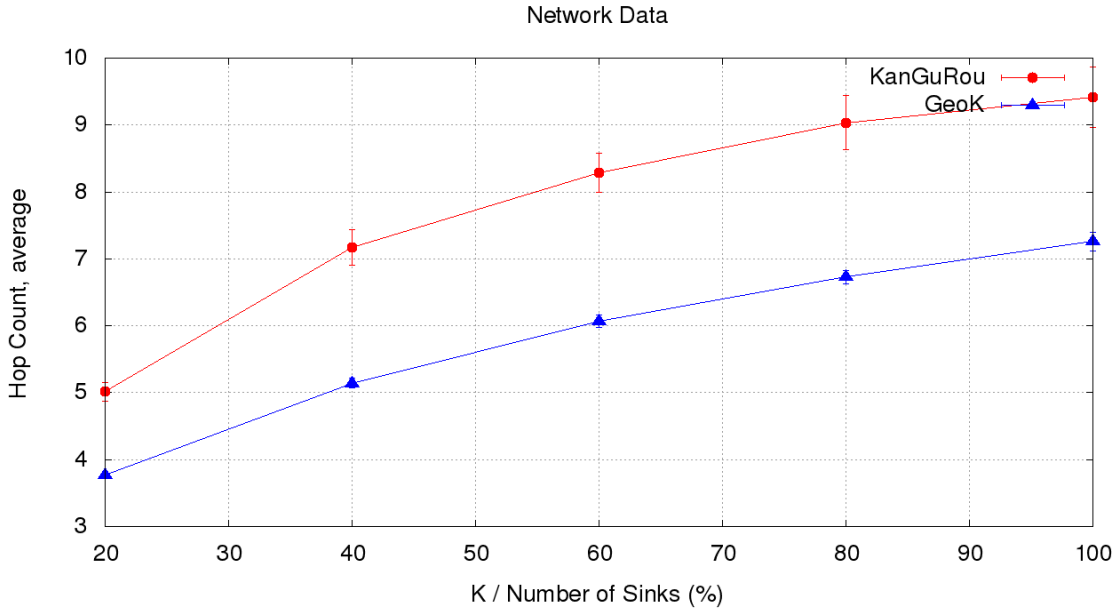


Figure 8.15: Average hop count - with void areas, 300 sensors, 30 sinks and k varying with a step of 20% of the number of sinks

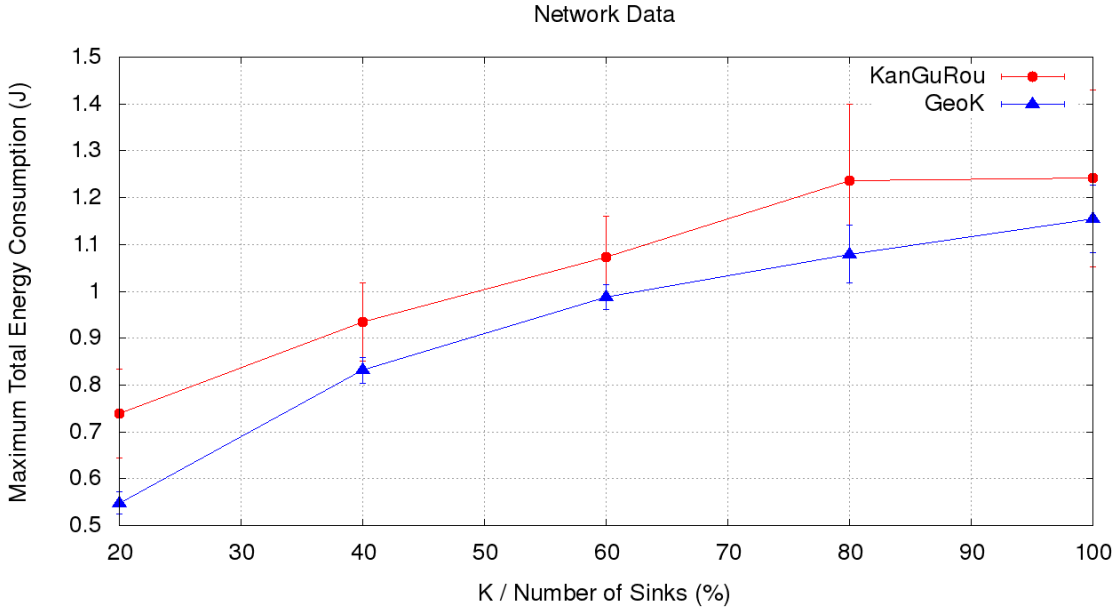


Figure 8.16: Maximum energy consumption - with void areas, 300 sensors, 30 sinks and k varying with a step of 20% of the number of sinks

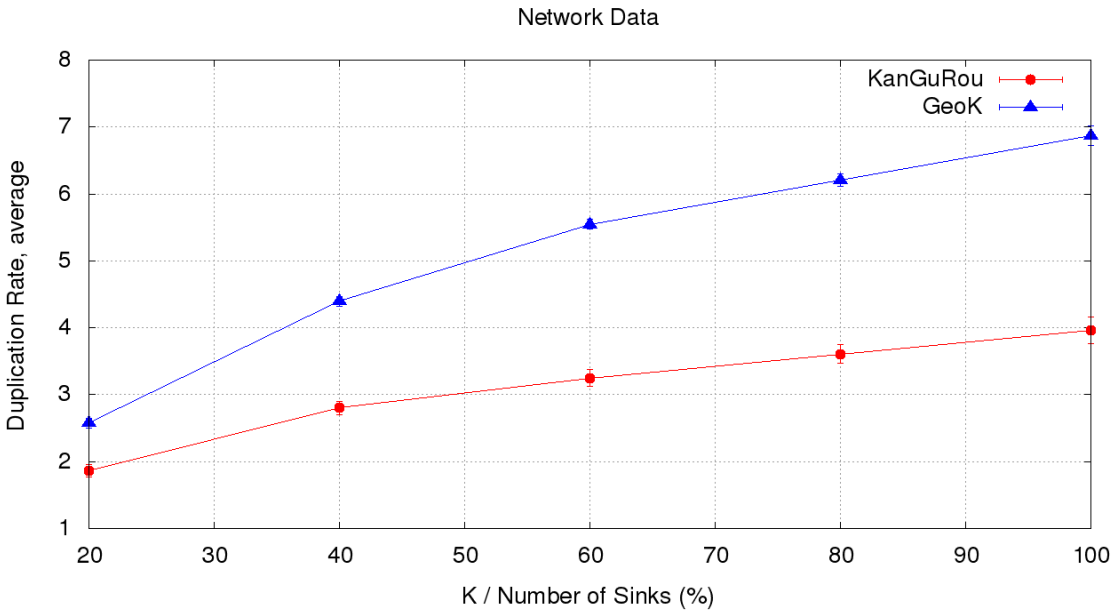


Figure 8.17: Average duplication rate - with void areas, 300 sensors, 30 sinks and k varying with a step of 20% of the number of sinks

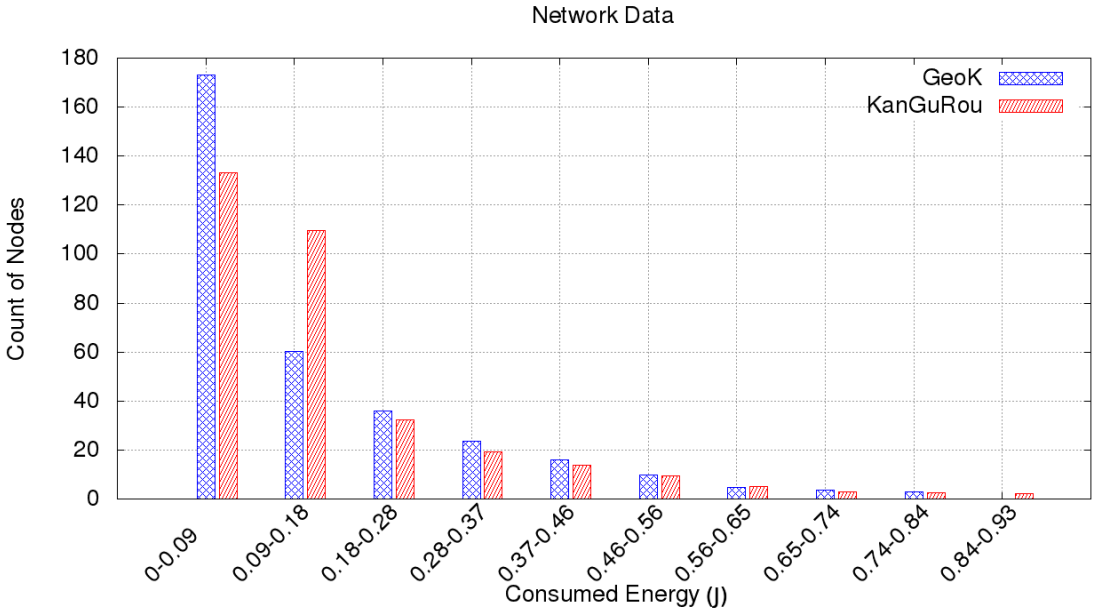


Figure 8.18: Distribution of nodes in terms of consumed energy - with void areas, 300 sensors, 30 sinks and $k = 40\%$ of the number of sinks

The distribution of nodes in terms of energy consumption in Figure 8.18 shows that GeoK is still able to balance the consumption, although in a lower level compared to the networks without void areas. The problem is that when the recovery mode is activated, since it follows the face routing strategy, the paths are the same. The better result obtained by GeoK comes from the void notification strategy. This preventive measure helps nodes avoiding the packet forwarding to void areas.

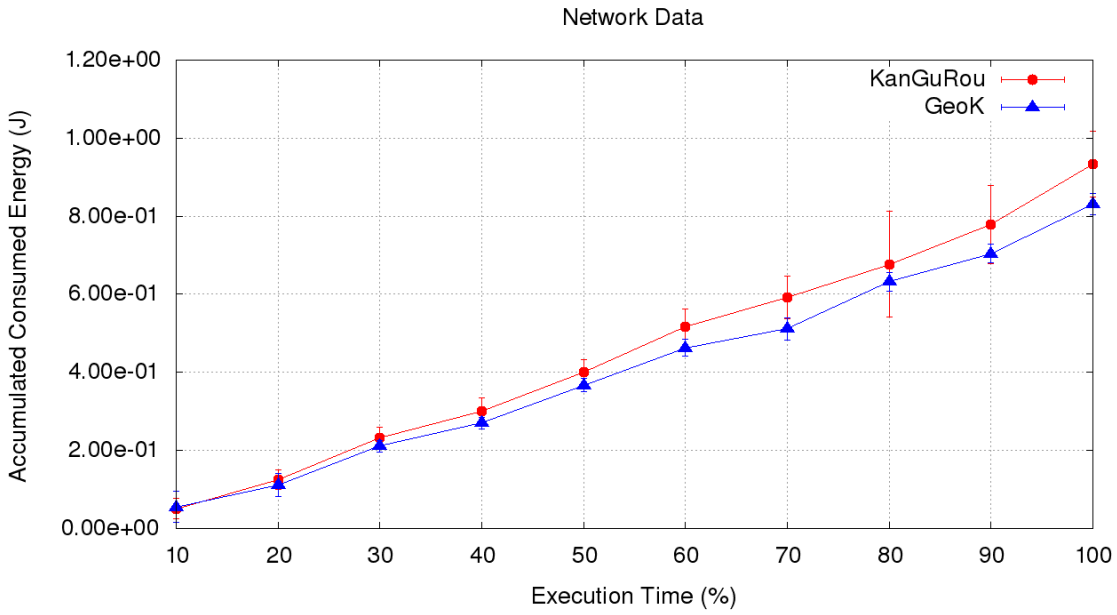


Figure 8.19: Energy consumption over time for the node that consumed the maximum energy at the end of the simulation - with void areas, 300 sensors, 30 sinks and $k = 40\%$ of the number of sinks

8.3.2/ VARIABLE NUMBER OF SENSORS AND PROPORTIONAL NUMBER OF SINKS

With the relative k fixed to 40% of the number of sinks, the number of sensors varying from 50 to 300 nodes and the number of sinks as a ratio of the number of sensors (10%), we intend to analyze the behavior of the solution in terms of proportional scalability. It is related to the behavior of the solution under a proportional increase of the number of sinks in relation to the network size. This scenario results in a controlled increase of the network. The increase in the number of sensor nodes also increases the amount of packets being forwarded. Because of that, increases in both latency and energy consumption are expected.

8.3.2.1/ NETWORKS WITHOUT VOID AREAS

For the scenario without void areas, we can notice that GeoK has better performances for both average latency and maximum energy consumption, as displayed in Figures 8.20 and 8.21 respectively. The increase in network size translates to even better results for GeoK in terms of maximum energy consumption, with a maximum gain of approximately 30%, and a moderate improvement of the Latency of 7% on average with a small variation. By increasing the network size, the possibilities of different paths are also multiplied. The packets are able to alternate through different route paths, leading to better energy balance and consequently a smaller maximum energy consumption. However, the gain for the average latency performance becomes stable, even with the increase of the network size. It is because the ratio k/sinks/sensors is kept the same. The k value is a ratio of the amount of sinks, and the number of sinks is a ratio of the amount of sensors.

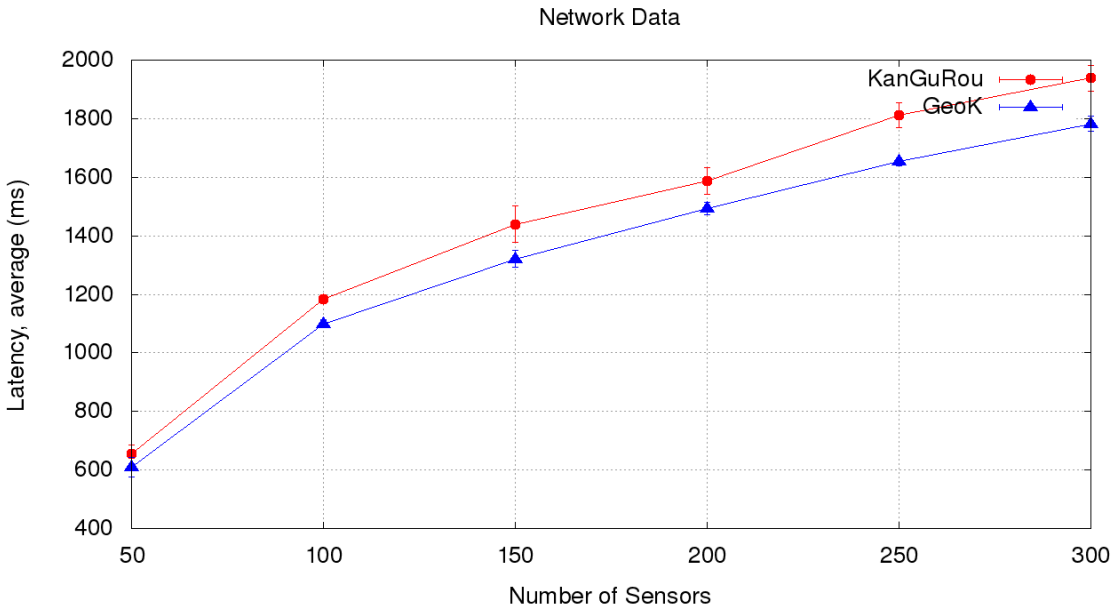


Figure 8.20: Average packet latency - no void areas, sensors varying from 50 to 300, sinks at 10% of the sensors and k fixed to 40% of the number of sinks

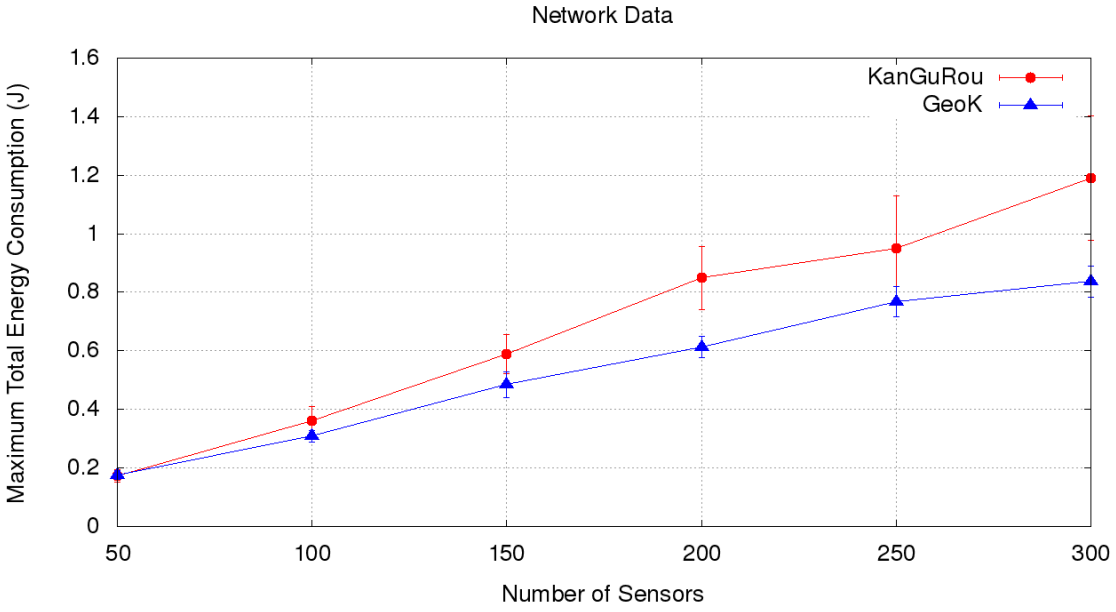


Figure 8.21: Maximum energy consumption - no void areas, sensors varying from 50 to 300, sinks at 10% of the sensors and k fixed to 40% of the number of sinks

8.3.2.2/ NETWORKS WITH VOID AREAS

For the case with void areas, we can also notice a performance gain of approximately 12% for the average latency, as shown in Figure 8.22. The better results for GeoK are again due to the void announcements strategy. And the stability of the gain comes from the

proportionality of the triplet k/sinks/sensors. In terms of maximum energy consumption, we can notice in Figure 8.23 a smaller gain for GeoK, with a maximum of approximately 20%. Because for some cases packets enter in recovery mode, as per the recovery strategy, the used path is always the same. Consequently, the maximum energy consumption increases, since there are fewer variations in the routing paths.

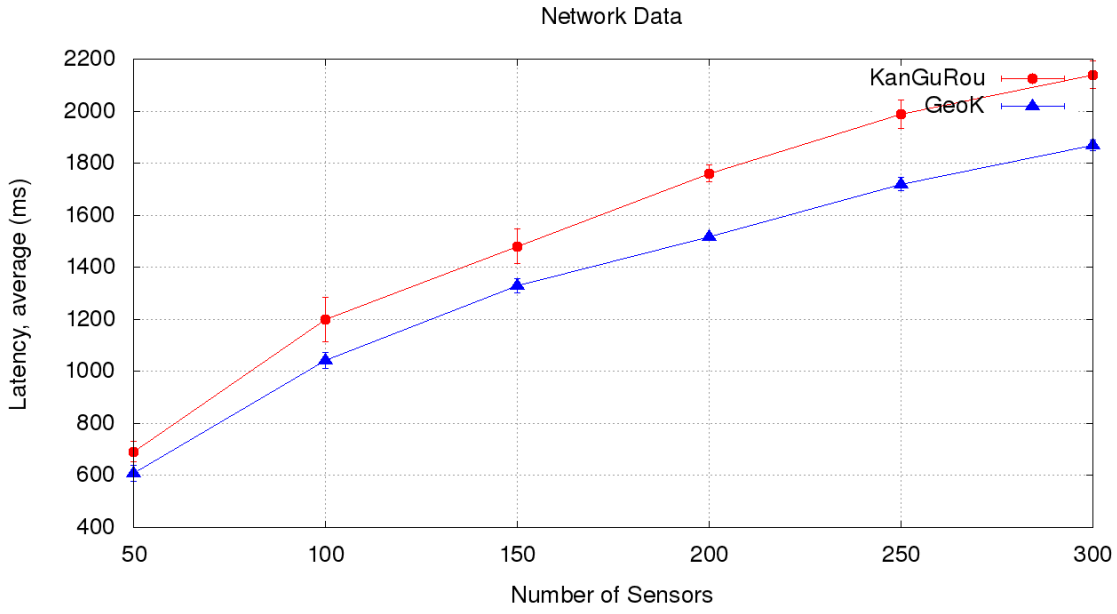


Figure 8.22: Average packet latency - with void areas, sensors varying from 50 to 300, sinks at 10% of the sensors and k fixed to 40% of the number of sinks

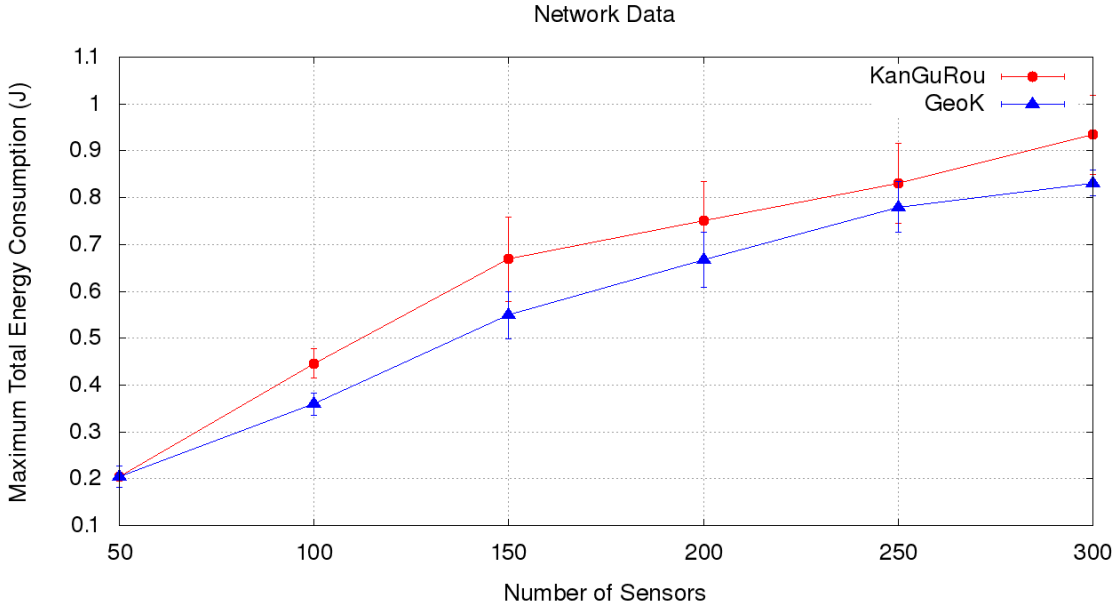


Figure 8.23: Maximum energy consumption - with void areas, sensors varying from 50 to 300, sinks at 10% of the sensors and k fixed to 40% of the number of sinks

8.3.3/ VARIABLE NUMBER OF SENSORS AND FIXED NUMBER OF SINKS

We also performed tests considering the increase of the number of sensor nodes and a fixed number of sinks, in order to verify the performance of our solution in terms of scalability without proportion. We considered networks without void areas, with 100, 500 and 1000 sensor nodes, 10 sinks and $k = |S|$, in order to analyze the worst case scenario, when the k value equals the total number of sinks. The main objective is to verify if our solution is capable of handling a massive increase of the number of sensors, with a small number of sinks to manage the data flows.

As we can see in Figure 8.24, the average latency result is quite similar between GeoK and KanGuRou. Since k is fixed to 100% of the sinks, GeoK has fewer chances to improve the path. However, we can observe in Figure 8.25 that GeoK uses in average 30% fewer hops than KanGuRou. Since the number of sinks is fixed, it is expected that the latency increases with the network growth.

In terms of maximum energy consumption, we can see in Figure 8.26 that our energy balance strategy makes GeoK to have better results, with an average improvement of 50%. Since the networks are much larger, the algorithm is able to distribute the load over different paths and balance the energy consumption.

We can see the evolution of the maximum energy consumption in Figure 8.27 for the network with 1000 sensor nodes. Our solution presents a lower increase of the energy consumption over time, because of the energy balance strategy. We can confirm this with Figure 8.28, that shows the distribution of nodes in terms of consumed energy at the end of the simulation. We can see that nodes in KanGuRou go beyond GeoK in the consumed energy, which represents a much faster progress towards energy depletion.

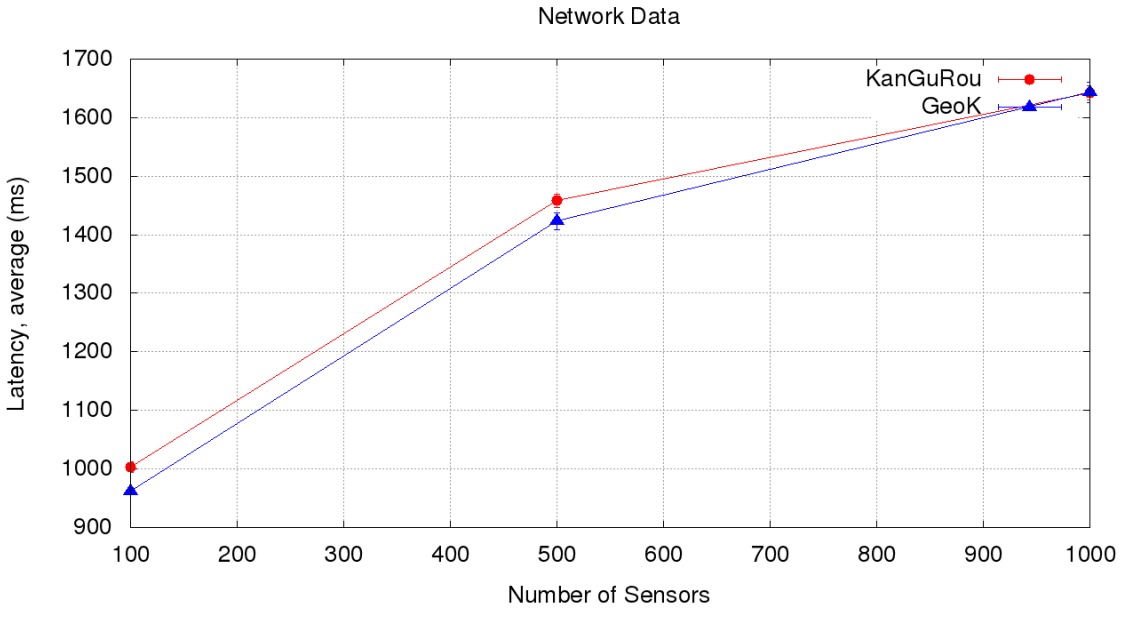


Figure 8.24: Average packet latency - without void areas, networks with 100, 500, 1000 sensor nodes, 10 sinks and k fixed to 100% of the number of sinks

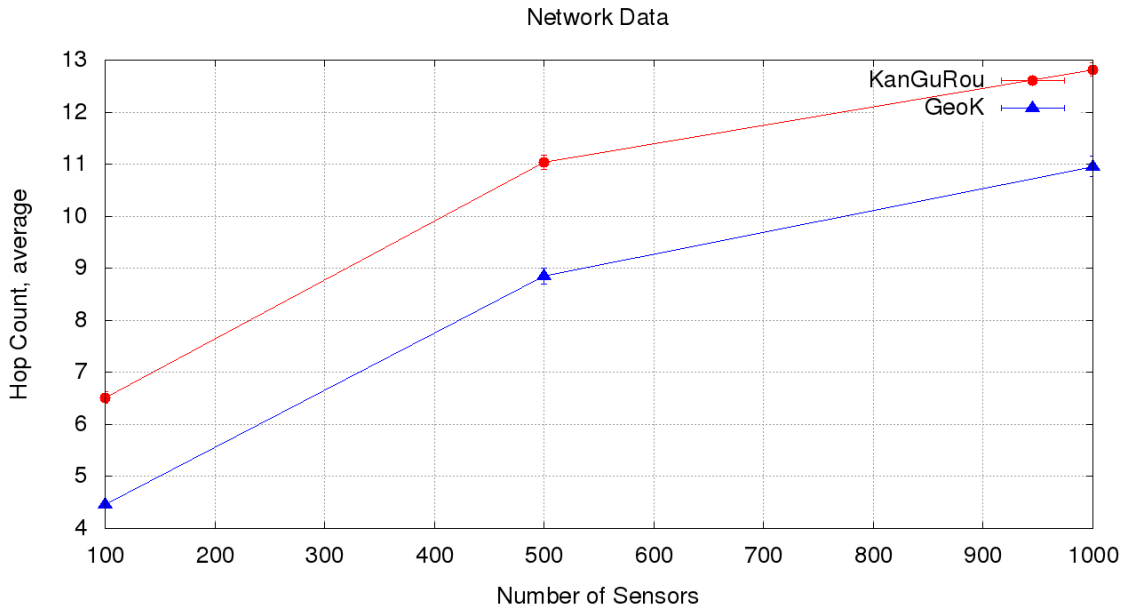


Figure 8.25: Average hop count - without void areas, networks with 100, 500, 1000 sensor nodes, 10 sinks and k fixed to 100% of the number of sinks

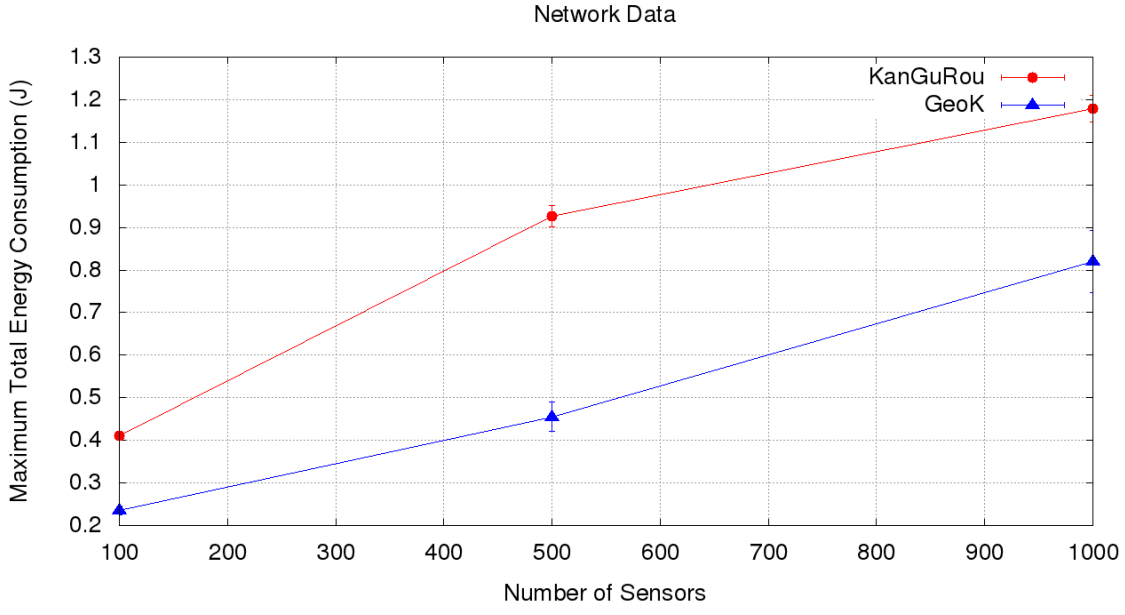


Figure 8.26: Maximum energy consumption - without void areas, networks with 100, 500, 1000 sensor nodes, 10 sinks and k fixed to 100% of the number of sinks

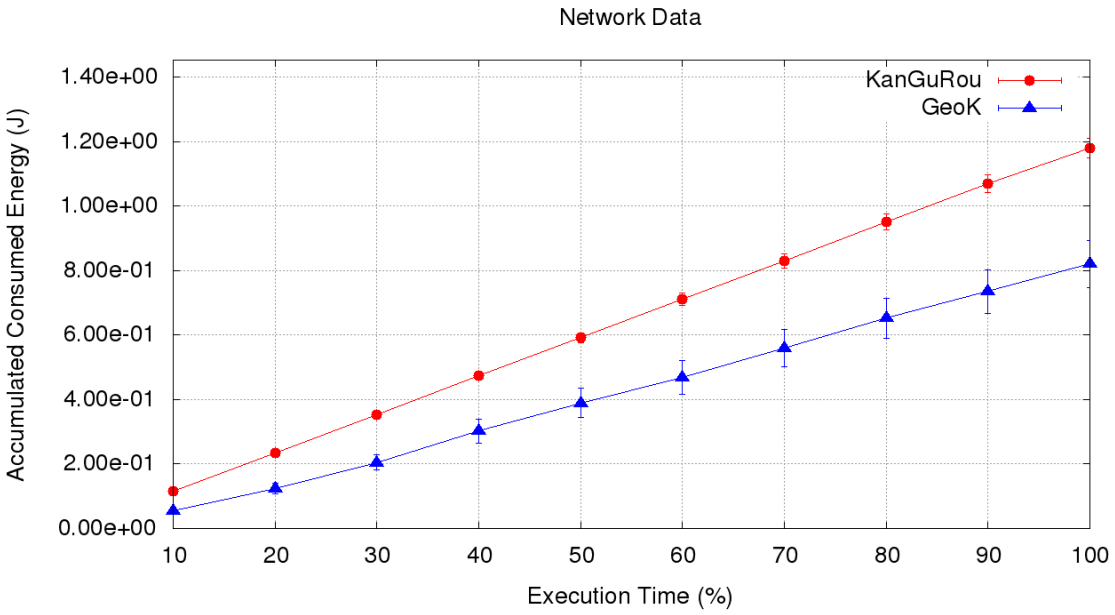


Figure 8.27: Maximum energy consumption over time - without void areas, network with 1000 sensor nodes, 10 sinks and k fixed to 100% of the number of sinks

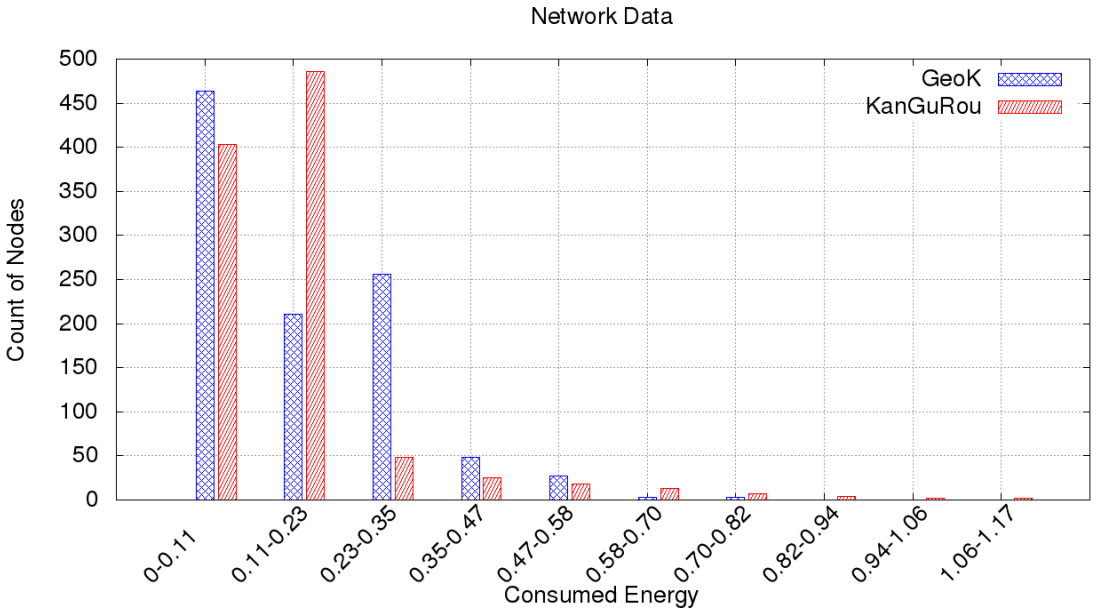


Figure 8.28: Distribution of nodes in terms of consumed energy at the end of the simulation - without void areas, network with 1000 sensor nodes, 10 sinks and k fixed to 100% of the number of sinks

8.4/ CONCLUSION

This chapter presented GeoK, a new k-anycast geographic routing solution for wireless sensor networks with multiple sinks, capable of forwarding packets to exactly k sinks, while minimizing the latency and increasing the network lifetime. Our strategy makes use of variable weighted metrics to establish the list of forwarders, as well as the necessity of packet duplication. The main goal was to find a balance between latency and network lifetime optimizations. We tested our solution through simulations against an existing strategy called KanGuRou. Both protocols were implemented on top of the Contiki OS and the simulations were executed using COOJA simulator. The simulation results indicate that our solution has an overall better performance than the existing protocol, with maximum gains of approximately 13% for latency and 30% for maximum energy consumption. For the scalability test without proportion, we can see that GeoK has a better behavior than KanGuRou, with an average gain of 50% for the maximum energy consumption.

GeoK is a general anycast protocol. It means that it may work as 1-anycast when $k = 1$ and as multicast when $k = |S|$, since its implementation follows a k-anycast communication scheme. However, the gain of generality is paid by an increase in complexity. The path decision must account for the fact that k varies within the interval $[1, |S|]$. When a packet duplication takes place, the distribution of sinks over each packet depends on k and the position of the sinks.

9

MULTICAST

Contents

For some applications, the information availability is an important requirement. The sensed data must be delivered to a number of specific destinations. We can cite a city traffic control system as an application where information availability is fundamental. In city traffic control, the current state of the roads may be distributed over a number of sinks in order to control the timings in traffic lights based on the traffic flows. Another example of application is related to the tracking of buses in a public transportation system. The information of the position of a bus is relevant to a number of predefined bus stations. Sensor nodes are spread in the city tracking the passing buses. The data from each bus is forwarded to a number of predefined sinks, representing each bus station served by the detected bus. In that sense, it is important to find mechanisms that allow the forwarding of the data to a number of predefined destinations. Thus, we are interested in the case of MS-WSN with multicast communication scheme as implementation strategy for the many-to-many case with predefined sinks.

The existing works considering a multicast communication scheme ([57] and [58]) are mainly focused on reducing the energy consumption, leaving the latency aspect unexplored. Because of that, we propose a multicast routing protocol capable of forwarding packets to all predefined sinks, focused on latency and network lifetime optimizations. Our solution searches for common forwarders in order to avoid the duplication of packets in early stages of the forwarding process.

9.1/ MOTIVATION

As presented in Table 2.1, from Chapter 2, the many-to-many problem can be solved with different implementation strategies. Nevertheless, if the sinks are predefined by the application, the number of possibilities is reduced, and some of the implementation strategies have important limitations. A unicast routing protocol can be used to solve the many-to-many problem with predefined sinks if we assume that packets are duplicated at source and forwarded in sequence. However, the energy consumption would increases dramatically, since the number of packets being forwarded individually is increased as a multiple of the number of target sinks. On the other hand, k-anycast routing protocols can be used for the many-to-many case with predefined sinks. The problem with k-anycast solutions in this scenario is that the final destination is decided on the way. Thus, for the case of predefined targets, the k-anycast implementation strategy only works if some conditions are satisfied. Considering S the set of all sinks and S' the set of predefined target sinks, S' is a subset of S ($S' \subseteq S$). The k value must be the number of all predefined sinks ($k = |S'|$), and the packet must be addressed only to the sinks in the subset S'. Moreover, k-anycast solutions are designed to perform the packet routing to any of the sinks, which is the advantage of this strategy, since the destination is decided during the forwarding process. When $k = |S|$, the destinations are predefined, which restricts the next hop decision. If the algorithm is designed to forward the packet to the closest sink, it may decide to duplicate the packet towards disjoint paths, instead of authorizing small deviations, which will increase the energy consumption.

In that sense, the most suitable communication scheme for the many-to-many problem with predefined sinks is a multicast strategy, where the packets are delivered to a number of specific sinks in the network. However, to the best of our knowledge, there are very few works in literature covering the multicast communication scheme. We can cite the works in [57] and [58], already presented in Chapter 3. They describe hop-count based solutions for the packet delivery to all sinks in the network. They rely on heavy network knowledge in order to perform all paths optimizations, merging common routes and later defining splitting points. They are also focused on data aggregation as a method to reduce energy consumption, and not explicitly oriented to latency minimization.

In this chapter, we propose a Geographic Multicast routing solution (GeoM), focused on reducing the packet latency and increasing the network lifetime. In our solution, the next hop decision is based on the combination of network metrics and the identification of common forwarders in order to avoid the early duplication of packets. We define a deviation factor which allows the packet forwarding towards a group of sinks through a common path, postponing the duplication to the point where no common forwarder is found. Moreover, based on the amount of consumed energy, the paths are constantly changed during the lifetime of the network, which balances the energy consumption in the node's neighborhood. We also combine the face routing and the passive participation [103] techniques as void handling strategy.

9.1.1/ SYSTEM MODEL AND ASSUMPTIONS

The system model and assumptions for this chapter are the same as the ones presented in Subsection 8.1.1. However, the notion of k sinks is no longer valid, and the information is forwarded towards all sinks defined at the source node. We assume that the set of

predefined sinks is given, and it represents all sinks in the network.

9.1.2/ k-ANYCAST COMPARISON

Due to the differences in terms of assumptions, the existing works in multicast communication scheme [57] and [58] are not considered for the comparison purposes. The two approaches are hop-count based solutions, meaning that the routing paths are based on the hop-distance from the sensor nodes to the sinks. Our solution considers a geographic routing strategy. Because of that, we use the existing k-anycast solution KanGuRou [56] as the reference for the validation process of our solution. The details of the KanGuRou algorithm were already presented in Subsection 8.1.2. As previously explained, a k-anycast routing protocol, such as KanGuRou, is capable of solving the many-to-many problem.

Additionally, a comparison between a k-anycast solution against a multicast solution puts in evidence the differences from the both strategies in terms of performance. It is possible to show that the network performance can be further improved with a forwarding process that is optimized to the scenario of many-to-many communication with predefined sinks.

9.2/ GEOGRAPHIC MULTICAST ROUTING (GEOM)

GeoM is a geographic routing protocol for Multi-Sink Wireless Sensor Networks that is capable of forwarding a generated packet to all sink in the network or to a set of predefined sinks using a multicast communication scheme. It is focused on finding the trade-off between latency and the network lifetime. The next hop is selected by the current node based on the calculation of weighted metrics, and the best intersection among forwarder candidates towards the sinks. The best intersection is the candidate forwarder that minimizes the final weighted metric towards the sinks. The algorithm can be divided into two main processes:

- **filtering process**, which is responsible for filtering the neighbor nodes in order to create a list of candidate forwarders. The filtering takes place by eliminating the neighbor nodes with negative progress and neighbors in void areas. If no neighbor is found, the recovery mode is triggered and a neighbor is selected using a void handling technique. At the same time, a broadcast message is sent to inform the neighbor nodes that the current node is in a void area.

- **selection process**, which is dedicated to effectively selecting the forwarders. The candidates are evaluated based on weighted metrics, and a forwarder is selected based on the largest intersection of sinks. We also consider a modifier called deviation factor, which increases the possibility of path intersections by relaxing the constraints of the weighted metrics.

Although GeoM does not require massive network knowledge, broadcast messages are used to periodically advertise the existence of neighbor nodes, their void area status and the amount of consumed energy. Moreover, relevant information is included in the packet in order to help the next hop decision. The information inserted in the header area of the

data packet includes the destination sinks and the current progress of the packet towards the sinks.

9.2.1/ FILTERING PROCESS

Algorithm 11 $GeoM(n, V_n, H, packet)$ - Run at the current node.

Input: n: current node, $packet$: packet to be forwarded, V_n: set of neighbors of n, H_s^n: set of neighbor nodes of n in a void area for the sink s

1: $S_r \leftarrow GetSinks(packet)$ /* Set of target sinks */
2: $p \leftarrow GetProgress(packet)$ /* The progress offered by the previous hop */
3: **if** $n \in S_r$ **then**
4: $S_r \leftarrow S_r \backslash \{n\}$
5: **end if**
6: **if** $S_r = \emptyset$ **then**
7: **return**
8: **end if**
9: $L \leftarrow \emptyset$ /* Set of pairs [candidate, sink] */
10: /* Check if the current node offers a positive progress towards the sinks */
11: /* A negative value means that the packet is in recovery mode */
12: $p_{new} \leftarrow ProgressTowards(n, S_r)$
13: **if** $p_{new} < p$ **then**
14: /* Select a candidate with the recovery mode using face routing */
15: $L = Recovery(n, V_n, S_r)$ /* In this case L has a single candidate */
16: $ForwardersIntersec(n, L, S_r, packet)$
17: **return**
18: **end if**
19: /* Look for the neighbors closer to the sinks than the current node */
20: $S' \leftarrow \emptyset$ /* Set of found sinks */
21: **for all** $v_j \in V_n$ **do**
22: **for all** $s_i \in S_r$ **do**
23: **if** $dist(n, s_i) > dist(v_j, s_i)$ **and** $v_j \notin H_{s_i}^n$ **then**
24: $L \leftarrow L \cup \{[v_j, s_i]\}$
25: **if** $s_i \notin S'$ **then**
26: $S' \leftarrow S' \cup \{s_i\}$
27: **end if**
28: **end if**
29: **end for**
30: **end for**
31: **if** $0 < |S'| \leq |S_r|$ **then**
32: $ForwardersIntersec(n, L, S', packet)$
33: **end if**
34: /* Check if a forwarder was found to all sinks */
35: $S'' \leftarrow S_r \backslash S'$
36: **if** $|S''| > 0$ **then**
37: /* Notify neighbors about the existence of a void area */
38: $SendVoidNotification(n, V_n, S'')$
39: $L = Recovery(n, V_n, S'')$ /* In this case L has a single candidate */
40: $ForwardersIntersec(n, L, S'', packet)$
41: **end if**
42: **return**

The filtering process starts at the reception of a packet and is described in Algorithm 11. The first step is to check if the current node n is a target sink itself. In this case, the node

must be removed from the set of destination sinks (S_r), since the packet has reached a target sink (Algorithm 11, line 4). The set of target sinks must be composed of a set of unique and identifiable entities. This is important in order to assure that all packets are delivered to different sinks. If, instead, we had only a number of target sinks to reach, we would risk delivering all the packets to the same sink, because of the distributed aspect of our solution.

The second step is to check the progress of the current node in relation to the previous hop. If the current node (n) progress towards the set of remaining sinks (S_r) represents a negative value compared to the value from the previous hop (line 12), it means that the packet was in recovery mode and no positive progress has yet been found. In this case, the packet must keep the recovery mode status and the forwarder candidate is selected using face routing [103]. In this case, only one node is selected and the packet is directly forwarded to this neighbor.

If the current node presents a positive progress towards the target sinks, the final step of the filtering process starts. Based on the neighbors (V_n) of the current node, the algorithm creates a new set of candidate forwarders (L). The solution selects the neighbor nodes with a positive progress to at least one sink, and excludes the neighbors that have already announced being in a void area (line 23), as exemplified in Figure 9.1.

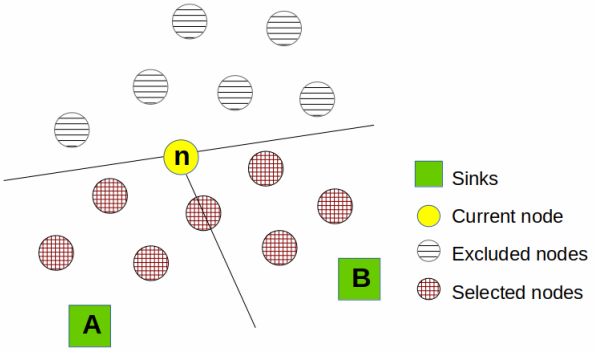


Figure 9.1: Filtering process

The $H^n_{s_i}$ set contains the neighbor nodes of n in a void area for the sink s_i. At the same time, a second sink set (S') is created in order to store the found sinks (line 26). If the number of found sinks is smaller than the amount of target sinks, it means that the algorithm could not find a suitable candidate to all sinks (line 35). In this case, the recovery mode is again activated in order to complete the list of candidates. A broadcast message (void announcement) is sent in order to inform the neighbors nodes that the current sink is in a void area (line 38). This strategy is important in order to allow future packets to avoid going through the void area.

9.2.2/ SELECTION PROCESS

The selection process is called when a forwarder candidate has to be selected. It is responsible for evaluating the list of candidates based on the weighted metrics and searching for the best intersection of candidates and sinks in order to avoid replications.

Aggregated decision metric calculation. The first step is to calculate the weighted

metrics to all candidates in relation to the sinks. The aggregated decision metric $W[s_j, v_i]$ represents the value of the calculated weighted metrics to the candidate node v_i in relation to the target sink s_j. The calculation process considers three network metrics and it is described in Algorithm 12. The first metric $D[v_i, s_j]$ is the relative distance from the candidate node v_i to the target sink s_j (line 14). The objective is to maximize the progress towards the sink (minimum distance between the neighbor node and the sink). The second metric $E[v_i]$ is the energy cost of the transmission of a packet from the current node to the neighbor node v_i (line 15). The objective is to select the neighbor node that minimizes the energy cost of the transmission. The combination of these two metrics provides a balance between the progress towards the sink and the energy cost of that progress. Finally, the third metric $C[v_i]$ is the total energy consumption of the neighbor node v_i (line 16). The objective is to select the node with the least energy consumption in order to balance the energy consumption in the neighborhood of the current node, avoiding the early depletion of the battery. The total consumed energy of the neighbor is obtained from the broadcast messages. The weights α, β and δ are used as configuration parameters, changing the impact of each metric in the final aggregated decision metric.

Algorithm 12 $W = CalculateMetric(n, S_r, L)$

Input: n: current node, S_r: set of destination sinks, L: set of pairs [candidate, sink] with positive progress
Output: W: a matrix $[|S_r|, |V|]$ that stores the calculated weighted metrics

1: /* α: weight for the distance metric */
2: /* β: weight for the energy cost metric */
3: /* δ: weight for the consumed energy metric */
4: **for all** $v_i \in \{v_k \mid [v_k, s] \in L\}$ **do**
5: **for all** $s_j \in S_r$ **do**
6: $D[v_i, s_j] \leftarrow dist(v_i, s_j)$ /* Returns the euclidean distance from node v_i to sink s_j */
7: **end for**
8: $E[v_i] \leftarrow EnergyCost(n, v_i)$ /* Returns the energy cost of the transmission from n to v_i */
9: $C[v_i] \leftarrow ConsumedEnergy(v_i)$ /* Returns the total consumed energy of the node v_i */
10: **end for**
11: $W \leftarrow \emptyset$
12: **for all** $s_j \in S_r$ **do**
13: **for all** $v_i \in \{v_i \mid [v_i, s_j] \in L\}$ **do**
14: $x \leftarrow \alpha \times \frac{D[v_i,s_j]-Min(D[*,s_j])}{Max(D[*,s_j])-Min(D[*,s_j])}$ /* for the distance between v_i and s_j */
15: $y \leftarrow \beta \times \frac{E[v_i]-Min(E)}{Max(E)-Min(E)}$ /* for the energy cost from n to v_i */
16: $z \leftarrow \delta \times \frac{C[v_i]-Min(C)}{Max(C)-Min(C)}$ /* for the energy consumption of neighbor v_i */
17: $W[s_j, v_i] = x + y + z$
18: **end for**
19: **end for**
20: **return** W

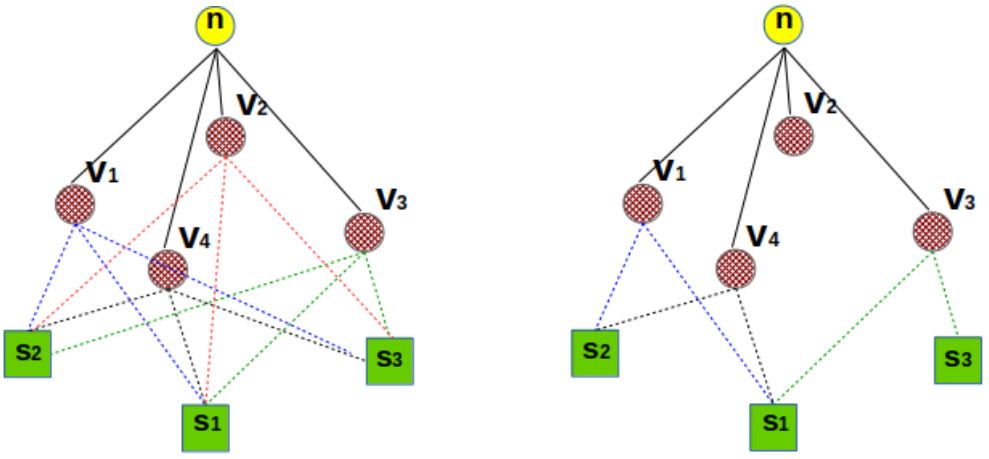
Candidates pre-selection. The second step is related to the pre-selection of candidates based on their aggregated decision metric $W[s_j, v_i]$ (Algorithm 13). The algorithm fixes a sink and calculates the mean ($\overline{w}_{s_j}$) and the standard deviation of the aggregated decision metric for all candidates having that sink in their list. Later, a new set (C) is created having all the candidates that passed the pre-selection step for each sink. The pre-selection eliminates the candidates that have their aggregated decision metric with a value higher than the mean plus a factor (γ) of the standard deviation (line 7). By using a factor for the standard deviation the algorithm increases the forwarding angle, allowing non optimal nodes to be selected. This strategy is important to increase the possibility of intersections

among candidates in relation to each sink. As a result of this process, the set of selected nodes is reduced.

Let us consider the network in Figure 9.2 as an example. Table 9.1 illustrates the selection process. The first line of the table represents all sensors, the first column represents all sinks, and the last column represents the mean of the aggregated decision metric in relation to a specific sink ($\overline{w}_{s_j}$). Each w_{s_j,v_i} element in the table represents the calculated decision metric $W[s_j, v_i]$. During the selection process, the aggregated decision metrics which are bigger than the mean in relation to sink plus a factor of the standard deviation ($w_{s_j,v_i} > \overline{w}_{s_j} + \gamma \times \sigma_{\overline{w}_{s_j}}$) are eliminated. The example in Table 9.1 shows the elimination of w_{s_3,v_1}, w_{*,v_2}, w_{s_2,v_3} and w_{s_3,v_4}. The network in Figure 9.2 shows the final result of the selection process.

Table 9.1: Calculated w for each pair $[s_j, v_i]$

	v_1	v_2	v_3	v_4	
s_1	w_{s_1,v_1}	~~w_{s_1,v_2}~~	w_{s_1,v_3}	w_{s_1,v_4}	$\overline{w}_{s_1}$
s_2	w_{s_2,v_1}	~~w_{s_2,v_2}~~	~~w_{s_2,v_3}~~	w_{s_2,v_4}	$\overline{w}_{s_2}$
s_3	~~w_{s_3,v_1}~~	~~w_{s_3,v_2}~~	w_{s_3,v_3}	~~w_{s_3,v_4}~~	$\overline{w}_{s_3}$



(a) Before the selection process

(b) After the selection process

Figure 9.2: Selection process (solid lines represent neighbor connections and dotted lines represent paths)

Intersection search. The intersection search strategy tries to avoid the early duplication of packets, which translates to fewer packets being individually forwarded. It is responsible for calculating the intersection among the pre-selected candidates. The algorithm creates the intersection set P with the structure that holds the set of candidate forwarders ($P[i].nodes$) to a common set of sinks ($P[i].sinks$). In order to start the process, the closest sink (s) to the current node is selected (Algorithm 13, line 14) and all its candidates are inserted in the structure ($P[p_k].nodes$). The sink s is removed from the set of destination sinks (S_r) and included in the set of the already selected sinks (S_{used}). The algorithm searches for the next sink s by selecting the closest sink to an already selected sink (line 33). Later, the algorithm checks the list of candidates of the new s against the set of the

already selected sinks (line 25). If an intersection is detected, the sink s is added in the P set corresponding to the intersection. Only the intersected candidates are kept in the $P[p_k].nodes$ structure. However, if no intersection is detected, a new element is created in the P set with the sink s and all its candidates. A new element in P represents a duplication of the packet due to the impossibility of finding an intersection. The entire process is repeated iteratively for all sinks.

Algorithm 13 $ForwardersIntersec(n, L, S_r, packet)$ - Run at the current node.

Input: n: current node, L: set of pairs [candidate, sink] with positive progress, S_r: set of remaining sinks, $packet$: packet to be forwarded

1: $W \leftarrow CalculateMetric(n, S_r, L)$ /* The matrix that stores the weighted metrics */
2: $C \leftarrow \emptyset$ /* The list that holds the structure having the candidate nodes */
3: **for all** $s_j \in S_r$ **do**
4: **for all** $v_i \in \{v_k | [v_k, s_j] \in L\}$ **do**
5: /* Check if the calculated metric of the node v_i for the sink s_j is smaller than the */
6: /* mean ($\overline{w}_{s_j}$) of all other candidates plus a factor (γ) of the standard deviation $\sigma_{\overline{w}_{s_j}}$ */
7: **if** $W[s_j, v_i] \leq \overline{w}_{s_j} + \gamma \times \sigma_{\overline{w}_{s_j}}$ **then**
8: $C[s_j].nodes \leftarrow C[s_j].nodes \cup \{v_i\}$
9: **end if**
10: **end for**
11: **end for**
12: $P \leftarrow \emptyset$ /* P is the list that holds the intersection structure */
13: $S_{used} \leftarrow \emptyset$
14: $s \leftarrow ClosestSink(n, S_r)$ /* Get the closest sink to n out of those in S_r */
15: **repeat**
16: /* Go through P and search for intersection */
17: **for all** $p_k \in P$ **do**
18: **if** $P[p_k].nodes \cap C[s_j].nodes \neq \emptyset$ **then**
19: $P[p_k].nodes \leftarrow P[p_k].nodes \cap C[s].nodes$
20: $P[p_k].sinks \leftarrow P[p_k].sinks \cup \{s\}$
21: $C[s].nodes \leftarrow \emptyset$
22: **end if**
23: **end for**
24: /* If no intersection is found, a new entry is created (packet duplication) */
25: **if** $C[s].nodes \neq \emptyset$ **then**
26: $p \leftarrow |P| + 1$
27: $P[p].nodes \leftarrow C[s].nodes$
28: $P[p].sinks \leftarrow \{s\}$
29: $C[s].nodes \leftarrow \emptyset$
30: **end if**
31: $S_{used} \leftarrow S_{used} \cup \{s\}$
32: $S_r \leftarrow S_r \backslash \{s\}$
33: $s \leftarrow ClosestSink(S_{used}, S)$ /* Get the closest sink in S to any sink in S_{used} */
34: **until** $S_r \neq \emptyset$
35: **for all** $p_k \in P$ **do**
36: $v \leftarrow v_i \in P[p_k].nodes$ which minimizes the mean for all $W[s_h, v_i]$, $s_h \in P[p_k].sinks$
37: $SendPacket(P[p_k].sinks, v, packet)$
38: **end for**
39: **return**

Considering the example in Table 9.2, sinks $[s_1, s_2]$ share two common candidate forwarders $[v_1, v_4]$, sinks $[s_1, s_3]$ share one common candidate $[v_3]$, and sinks $[s_2, s_3]$ have no common candidates. For the sake of the example, let us assume that s_2 is the closest sink

to the current node. Because of that, s_2 and all its candidate forwarders (v_1 and v_4) are included in the P structure. The P structure contains $[s_2], [v_1, v_4]$. Then, the closest sink to s_2 is selected. Assuming that s_1 is the selected sink, the candidate forwarders of the sink s_1 are compared to the existing forwarders in the P structure. An intersection is found with the candidate forwarders $[v_1, v_4]$, so s_1 is included in the P structure in the same set of the sink s_2. The P structure contains now $[s_1, s_2], [v_1, v_4]$. The candidate forwarder v_3 for the sink s_1 is eliminated because it is not a valid candidate forwarder for the sink s_2. Finally, the sink s_3 is selected. The candidate forwarders of the sink s_3 are compared to the existing forwarders in the P structure. At this time, no intersection is found. In this case, a new entry is created in P. The final P structure contains the elements $\{[s_1, s_2], [v_1, v_4]\}$ and $\{[s_3], [v_3]\}$.

Table 9.2: Intersection search

Sinks	Candidate Forwarders			
s_1	v_1	-	v_3	v_4
s_2	v_1	-	-	v_4
s_3	-	-	v_3	-

Forwarder selection. The final step considers that multiple candidates may still exist within an entry p_k of the P list, meaning that a set of sinks $P[p_k].sinks$ may have more than one candidate forwarder $|P[p_k].nodes| > 1$. For that matter, the algorithm searches for the candidate forwarder that minimizes the mean of the aggregated decision metric considering the set of sinks $P[p_k].sinks$. This is the opposite view of the pre-selection step, where the mean was calculated for all candidates for a single sink. At this time, we calculate the mean for all sinks in $P[p_k].sinks$ for a given candidate $v_i \in P[p_k].nodes$. Once the candidate v_i is defined, the packet is sent.

Table 9.3: Forwarder Selection

	v_1	v_2	v_3	v_4	
s_1	w_{s_1,v_1}	-	-	w_{s_1,v_4}	$\overline{w}_{s_1}$
s_2	w_{s_2,v_1}	-	-	w_{s_2,v_4}	$\overline{w}_{s_2}$
s_3	-	-	w_{s_3,v_3}	-	$\overline{w}_{s_3}$
	$\overline{w}_{v_1}$			$\overline{w}_{v_4}$	

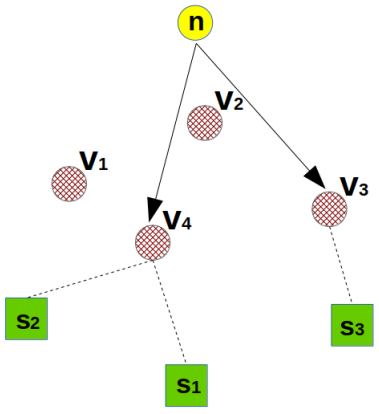


Figure 9.3: Packet duplication

Considering the example in Table 9.3, $\overline{w}_{v_1}$ is the mean of the weighted metric in relation to v_1, and the sink/neighbors intersection structure is composed of $\{[s_1, s_2], [v_1, v_4]\}$ and $\{[s_3], [v_3]\}$. For the entry $\{[s_1, s_2], [v_1, v_4]\}$, the forwarder neighbor is the node having $Min(\overline{w}_{v_1}, \overline{w}_{v_4})$, and for the entry $\{[s_3], [v_3]\}$, the forwarder neighbor is v_3.

If we assume that $\overline{w}_{v_4}$ minimizes the mean of the weighted metrics towards s_1 and s_2, then Figure 9.3 represents the final selection of forwarders to reach the sinks s_1, s_2 and s_3.

9.3/ SIMULATION AND RESULTS

GeoM protocol was developed in C as an application module for Contiki OS and evaluated using Cooja. The performance of our solution was compared to KanGuRou, that was also adapted to Contiki OS and tested with Cooja under the same configurations. KanGuRou is a k-anycast solution, however it is capable of performing multicast routing, if we consider the conditions described in Section 9.1. The decision to use KanGuRou as a benchmark is related to the compatibility of assumptions. Both GeoM and KanGuRou are designed for geographic routing, which implies a small need of network knowledge, no routing table maintenance and no set up phase.

Table 9.4: Configuration for the simulations

	Simulation Settings	
Deployment density (d)	8 neighbors (on average)	
Network area (variable)	$\frac{\pi \times r^2}{d} \times \lvert V \rvert$ (9.1), ($\lvert V \rvert = \lvert N \rvert + \lvert S \rvert$)	
Communication range (r)	50 m	
Packet size	100 bytes	
Packet generation rate	20% chance at every minute for each sensor	
Radio type	802.15.4	
MAC protocol	CX-MAC, modified version of [106]	
Execution time	120 minutes for each network	
Weights and deviation factor	$\alpha = 0.7$, $\beta = 0.1$, $\delta = 0.2$ and $\gamma = 0.5$	
	Scalability test	
	Proportional	Not proportional
Number of sensors ($\lvert N \rvert$)	50, 100, 150, 200, 250, 300	100, 500, 1000
Number of sinks ($\lvert S \rvert$)	10% of the number of sensors	10 sinks
Number of networks	60 with voids, 60 without voids	300 without voids

The simulation environment and details are outlined in Table 9.4. In order to keep a similar deployment density over all variations of $|V|$, we make the network area vary with the number of deployed nodes, as given by equation 9.1 in Table 9.4. The weight values and the deviation factor were obtained with experimentation and the analysis of the influence of each metric in the behavior of the solution. We tested the solutions under different network topologies, varying the number of sensor nodes and the existence of void areas.

The performance is evaluated by observing the average of all metrics for all simulations, and the results are presented with a confidence interval of 95%. The considered metrics include the packet latency, defined by the average time a packet takes to be routed from the source to all sinks. It is calculated to each network considering the sum of the latencies of the packets that were successfully received by all sinks divided by the number of

packets. We also analyze the maximum energy consumption, which regards the node that consumed the largest amount of energy in the network at the end of the simulation. It gives an indication of the network lifetime, since it shows how far the node is from the entire depletion of its battery. We consider a network to be alive as long as all nodes have some energy. Therefore, network lifetime is considered to be the earliest moment at which a node's battery is completely depleted.

Two main testing scenarios were considered: proportional scalability and scabality without proportion. The results are presented likewise. For the proportional scalability analysis, the amount of sensor nodes and sinks increases at the same rate in order to evaluate the behavior of the solution when the network has a proportional grow. For the scalability without proportion, the number of sensor nodes increases and the number of sinks remains constant.

9.3.1/ VARIABLE NUMBER OF SENSORS AND PROPORTIONAL NUMBER OF SINKS

With a variable number of sensor nodes and a proportional number of sinks, we are interested on analyzing the performance of GeoM under an organized growth. The objective is to evaluate the capacity of the algorithm to handle the increase of the number of destinations. The results are presented in two groups: networks without void areas and networks with void areas.

9.3.1.1/ NETWORKS WITHOUT VOID AREAS

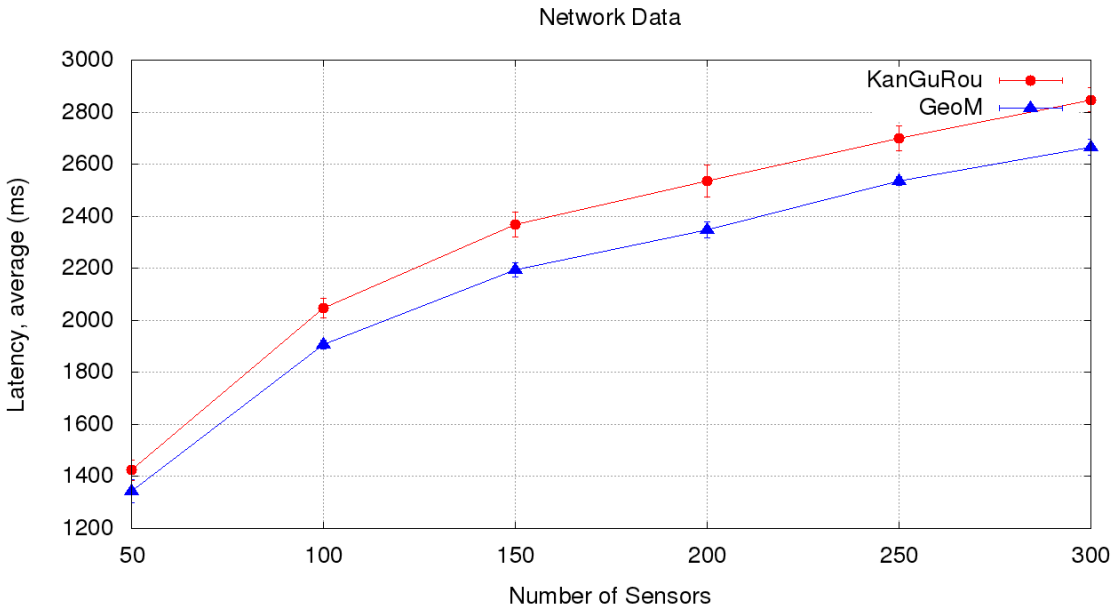


Figure 9.4: Average packet latency results with sensor varying from 50 to 300 nodes and sinks at 10% of the sensor nodes in a network without void areas

The latency results are presented in Figure 9.4. We can notice that GeoM presents a performance gain of around 7%. This as a positive indication, since the energy balance

and the deviation factor strategies in GeoM generally increase the path length. The performance gain can be explained by the fact that GeoM duplicates the packets at better diverging points, not only when the paths become disjoint, which speeds up the delivery. This is part of the trade-off between better energy consumption and latency.

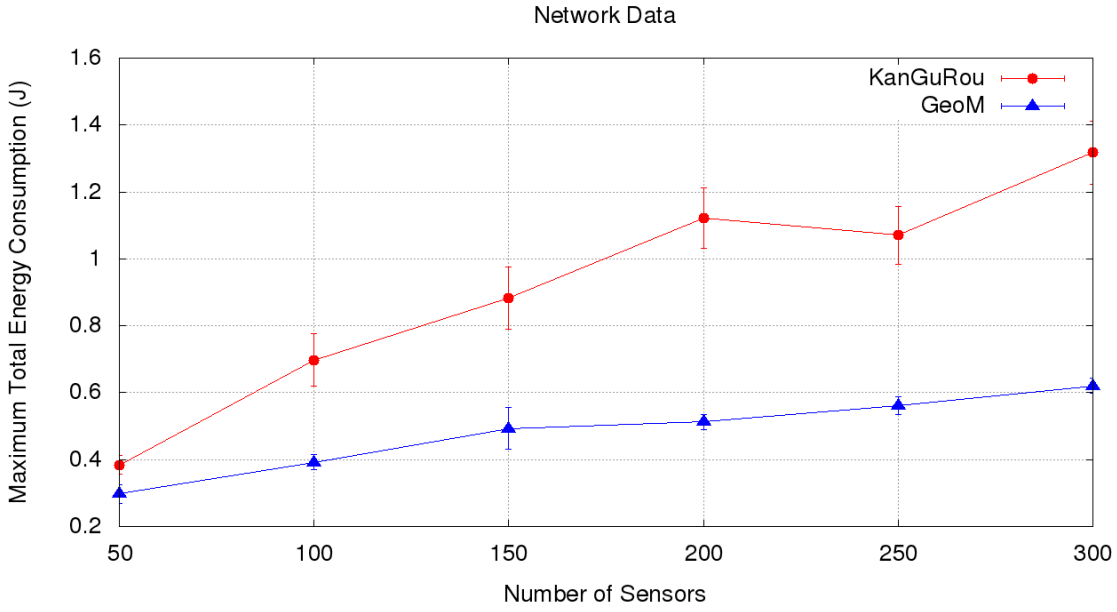


Figure 9.5: Maximum energy consumption results with the network size varying from 50 to 300 nodes and sinks at 10% of the sensor nodes - networks without void areas.

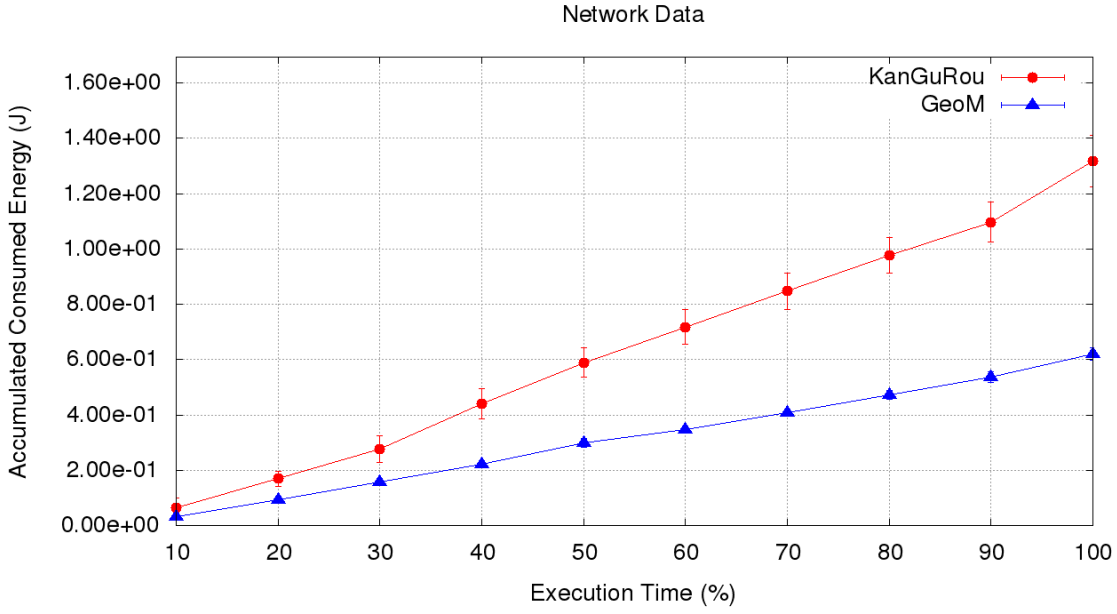


Figure 9.6: Evolution of the maximum consumed energy - results for 300 sensors and 30 sinks in a network without void areas

As we can notice in Figure 9.5, GeoM presents a better performance in terms of maximum

energy consumption, with gains varying from 23% to 53%. The values for the maximum energy consumption increase when the network grows for both solutions. However, the increase in GeoM is much smoother compared to KanGuRou. This is explained by our energy balance strategy and the deviation factor that distributes the load among other neighbors instead of using the same path for every forwarding. We can confirm this observation with Figure 9.6, that shows the evolution of the maximum energy consumption during the simulation for the scenario with 300 sensor nodes. We can see that GeoM has a much smoother increase compared to KanGuRou. At the end of the simulation, GeoM presents half of the value for the maximum energy consumption. In terms of longevity, the maximum energy consumption for KanGuRou translates into a faster depletion of the node's battery.

In Figure 9.7 we present the distribution of nodes in terms of energy consumption at the end of the simulation for the scenario with 300 sensor nodes. For KanGuRou, we can see a huge concentration of nodes with a small energy consumption, and a small group of nodes with a high level of energy consumption. For GeoM, we can see a different configuration, with a concentration of nodes at the first half of the energy intervals, which shows a better distribution of the energy consumption.

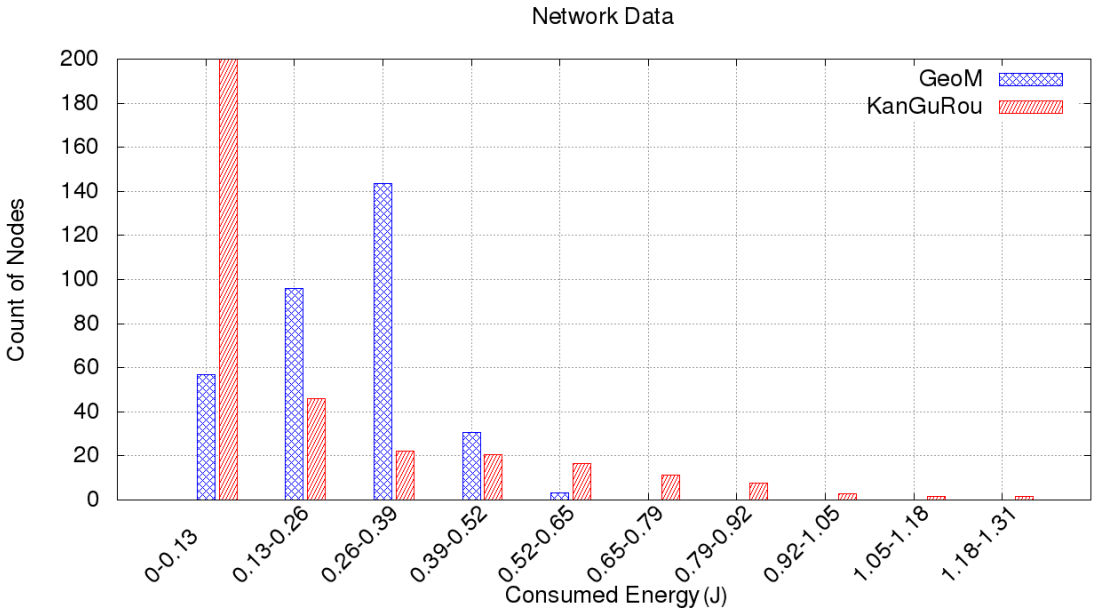


Figure 9.7: Distribution of nodes in terms of consumed energy by the end of the simulation - results for 300 sensors and 30 sinks in a network without void areas

The network may face many issues when a node's battery is completely depleted. The first one is the risk of disconnection, when part of the network is isolated and unable to forward packets to a sink. Even if a disconnection does not take place at a first moment, void areas may be created, since the depleted nodes are generally the ones within the sink neighborhood. The consequence of that is an increase of the latency and energy consumption, since a longer path is expected in order to bypass the void area. Another issue is related to the application, that faces a lack of coverage in the sensed field.

9.3.1.2/ NETWORKS WITH VOID AREAS

The existence of a void area in the network is a common problem of the geographic routing. Strategies must be applied in order to handle the packets entering in a void area. GeoM deals with void areas using two existing techniques, the face routing and the passive participation (void announcements) [103]. When a packet is in recovery mode, after being forwarded to a void area, the next hop decision does not consider the energy and latency optimizations. Normally, the path to exit the void area is always the same, regardless the amount of time it is used. In order to avoid this scenario, GeoM makes use of the void announcements as a way to prevent packets from entering in a void area, whenever it is possible. The simulation results show that GeoM is able to handle void areas and provide performance gains for both the energy and latency.

In terms of latency, GeoM presents a performance gain of around 10% in relation to KanGuRou, as displayed in Figure 9.8. This can be explained by the void announcements strategy. When packets enter in void areas, the paths become longer as a consequence of the void handling technique to exit the void area. However, with the nodes announcing their void situation, neighbors try to forward packets to other directions, avoiding the problematic area.

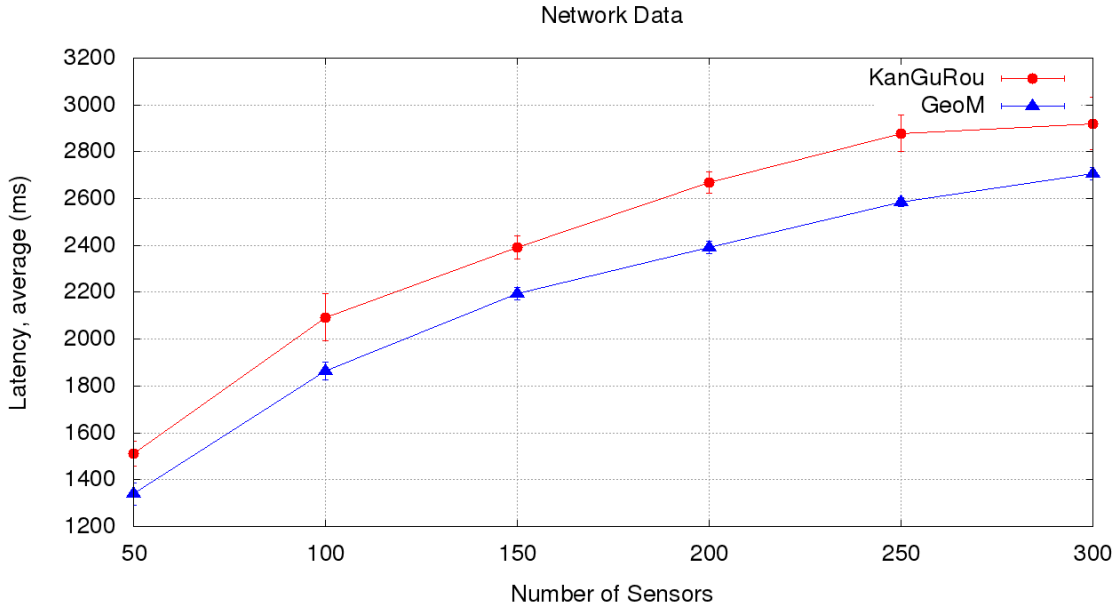


Figure 9.8: Average packet latency results with sensor varying from 50 to 300 nodes and sinks at 10% of the sensor nodes (network with void areas)

In Figure 9.9 we can notice a performance gain in the maximum energy consumption for GeoM varying from 5% to 26%. Even with the existence of void areas, GeoM is capable of distributing the loads and consequently reducing the maximum energy consumption. The network grow affects GeoM positively, with an increase in the performance gain in relation to KanGuRou. With the increase of the network size, the nodes in void areas may be avoided more easily, since both sink options and the possibility of candidate forwarder intersections are also increased.

We can see in Figure 9.10, for the scenario with 300 sensor nodes, that GeoM and Kan-

GuRou have a similar behavior in terms of energy consumption over time. However, GeoM has a slightly better curve compared to KanGuRou, reaching the end of the simulation with a level of energy consumption that represents less than 80% of the energy consumed by KanGuRou.

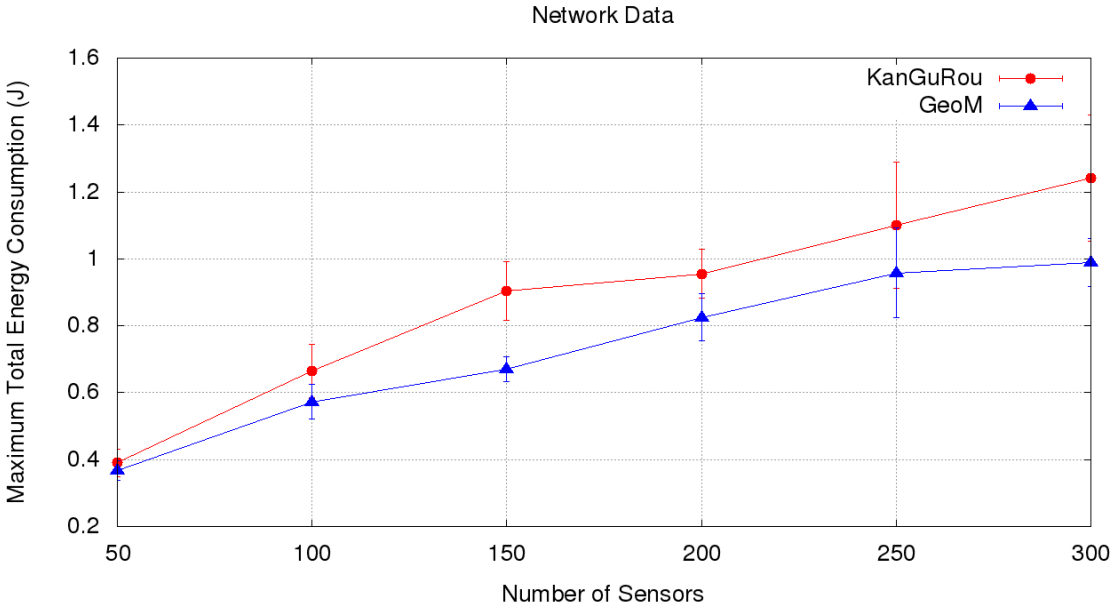


Figure 9.9: Maximum energy consumption results with sensor varying from 50 to 300 nodes and sinks at 10% of the sensor nodes in networks with void areas

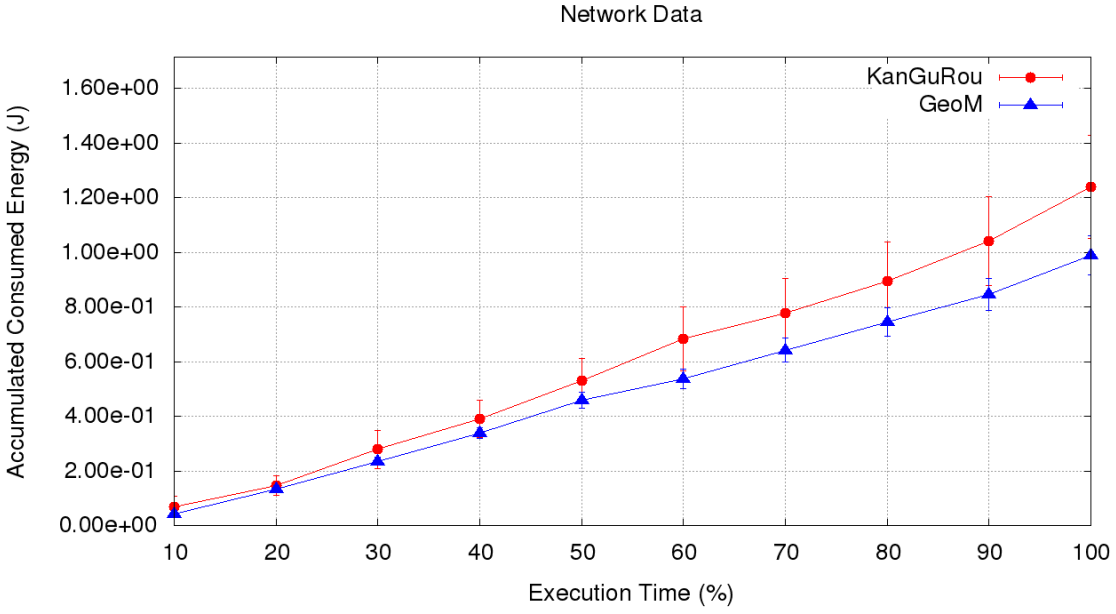


Figure 9.10: Evolution of the maximum consumed energy - results for 300 sensors and 30 sinks (network with void areas)

The distribution of the nodes in terms of energy consumption at the end of the simulation

presented in Figure 9.11 shows that GeoM manages to distribute the energy consumption, even for the networks with void areas. This may represent an increase of the network lifetime in relation to KanGuRou. If the initial battery level of the nodes was set to $1J$, at the end of the execution, KanGuRou would already have nodes with depleted batteries, while all nodes for GeoM would still be alive.

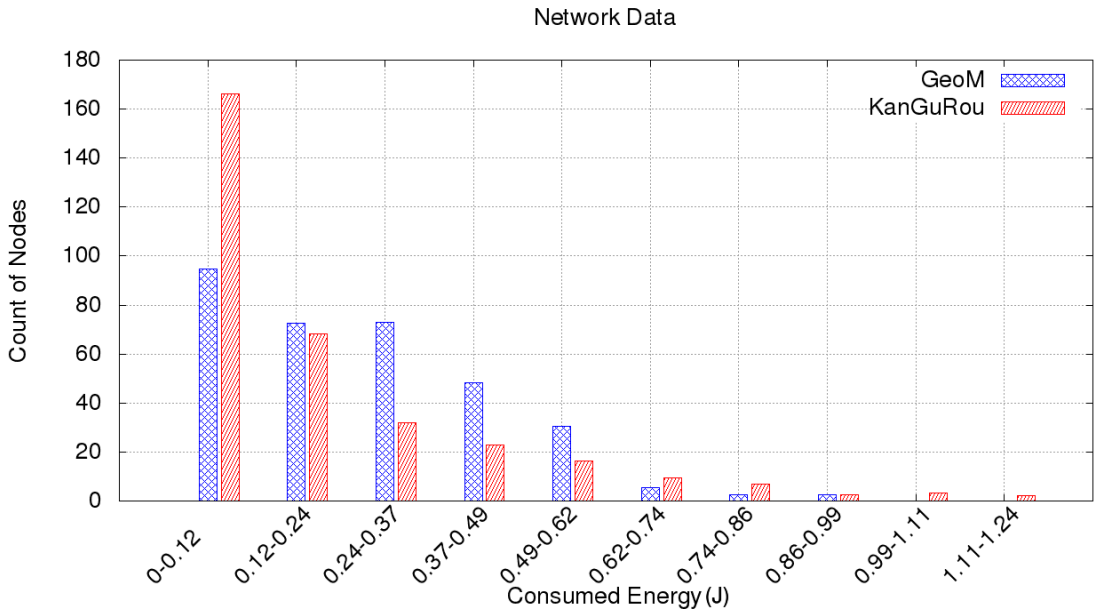


Figure 9.11: Distribution of nodes in terms of consumed energy by the end of the simulation - results for 300 sensors and 30 sinks in networks with void areas

9.3.2/ VARIABLE NUMBER OF SENSORS AND FIXED NUMBER OF SINKS

For the scalability test without proportion our objective is to evaluate the behavior of the solution when the number of sensor nodes is increased, but the number of sinks is kept the same. The objective is to identify if GeoM is capable of handling the network size growth, with an increase in the number of packets. We performed tests considering networks without void areas, with 100, 500 and 1000 sensor nodes, and a fixed number of 10 sinks.

In terms of latency can see in Figure 9.12 a constant performance gain of approximately 10%, which is explained by the fact that GeoM is optimized to deliver the packet to all sinks, searching intersections, but performing a duplication of the packet at better diverging points.

The maximum energy consumption result shows that GeoM has a much better behavior than KanGuRou, with a performance gain that increases with the network growth, as in Figure 9.13. Our algorithm is capable of distributing the energy consumption within the neighborhood of the forwarder node, due to our strategy of searching for intersections among forwarders and changing paths over time.

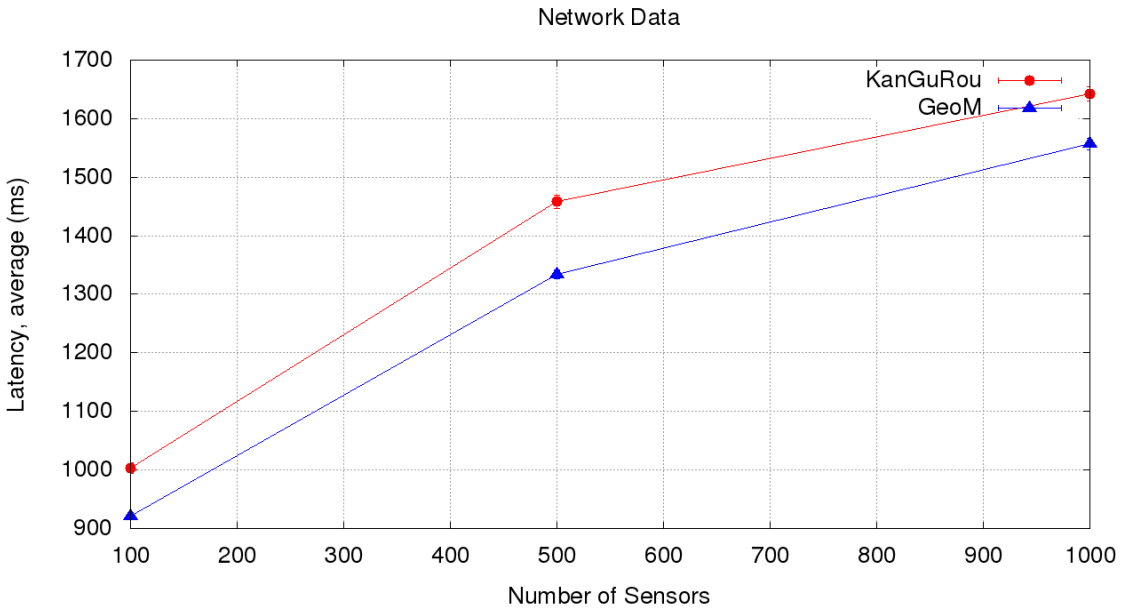


Figure 9.12: Average packet latency - results for networks composed of 100, 500, 1000 sensor nodes and 10 sinks (without void areas)

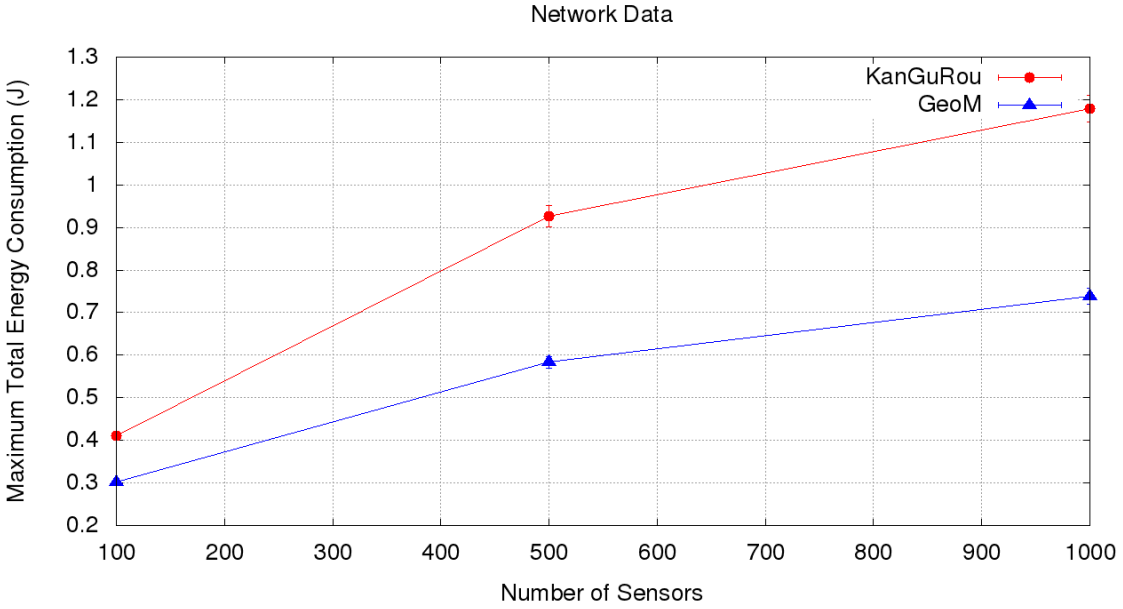


Figure 9.13: Maximum energy consumption - results for networks composed of 100, 500, 1000 sensor nodes and 10 sinks (without void areas)

The evolution of the maximum energy consumption in Figure 9.14 for the network with 1000 sensor nodes presents a lower increase of the energy consumption over time compared to KanGuRou, and the distribution of nodes in terms of consumed energy in Figure 9.15 shows that some nodes in KanGuRou had a much higher energy consumption than GeoM.

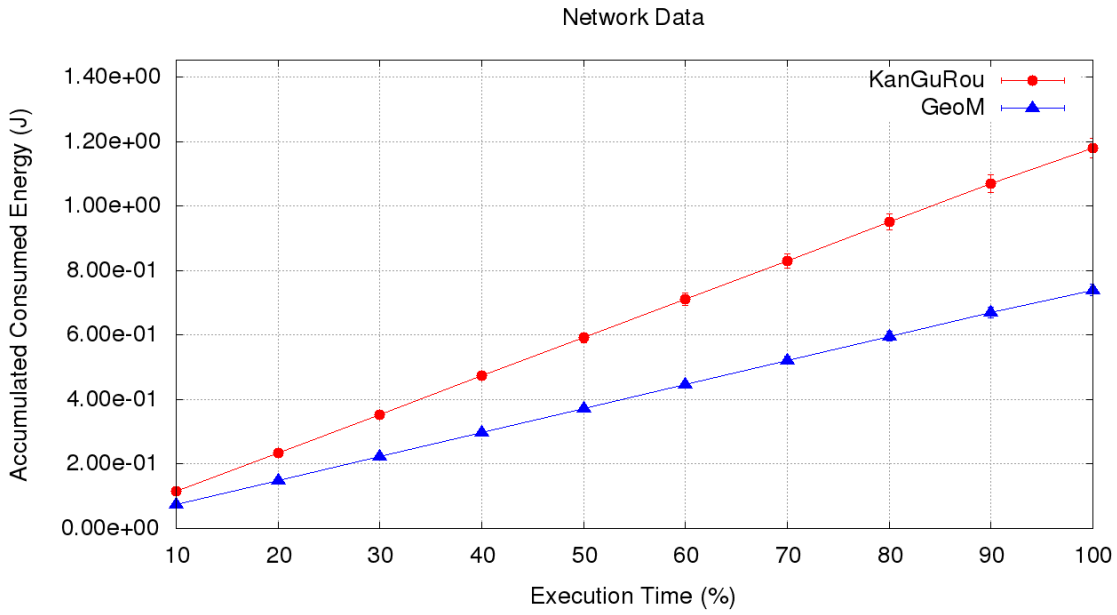


Figure 9.14: Evolution of the maximum consumed energy - results for networks composed of 1000 sensor nodes and 10 sinks (without void areas)

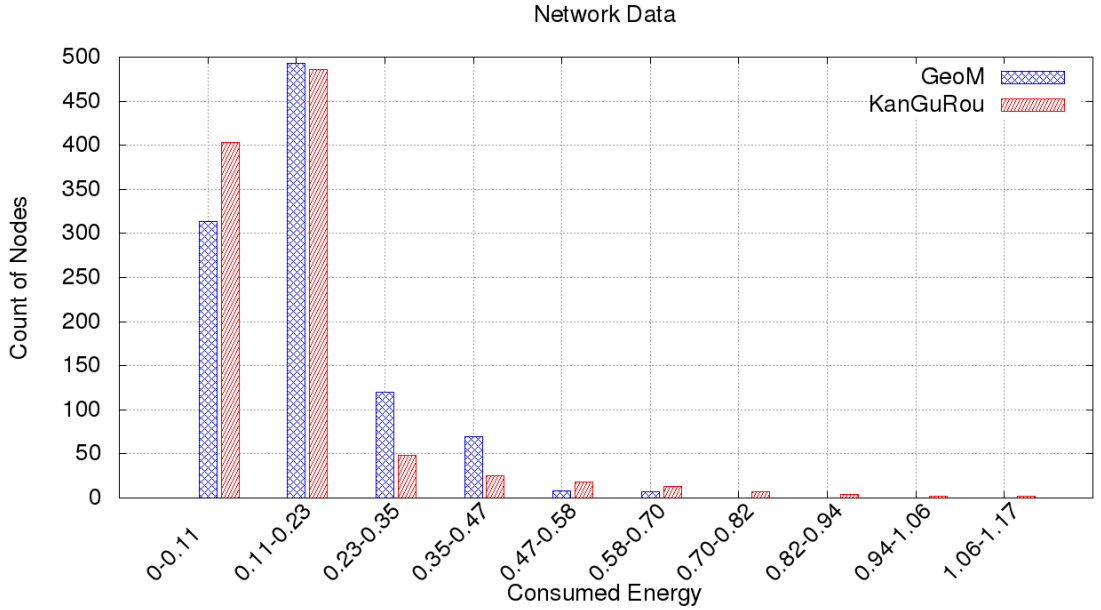


Figure 9.15: Distribution of nodes in terms of consumed energy at the end of the simulation - results for networks composed of 1000 sensor nodes and 10 sinks (without void areas)

9.3.3/ COMPARISON OF GEOM WITH GEOK

GeoM and GeoK solutions share a number of common objectives. Both solutions focus on reducing the latency and the energy consumption in order to increase the network

lifetime. However, the forwarding objectives are different. With GeoK, the objective is to forward a packet to any k sinks in the network. They are not predefined and the target sinks may vary from one packet to another. For GeoM, the objective is to forward packets to a number of predefined sinks. The routing mechanisms are also different. GeoK searches for the best candidate forwarders and tries to avoid the duplication of packets by prioritizing an already selected forwarder. Since any of the sinks may receive the packet, it is enough to search for one best candidate to any of the sinks. GeoM searches for intersections of common forwarders to the maximum number of sinks. Since all predefined sinks must be reached, the objective is to identify a common path that represents a compromise between latency and energy consumption to a maximum number of sinks.

The comparison of both solutions is possible if we consider that all sinks in the network must be reached. For the GeoK algorithm, we set k as the number of all sinks. This way it is possible to analyze both solutions under the same conditions and objectives. We can see in Figure 9.16 that GeoM performs better than GeoK for the average latency. This behavior is expected, since GeoK is not optimized to send packets to all sinks in the network. Since it prioritizes the forwarding to the closest sink, the subsequent sinks are penalized by this choice with a latency increase.

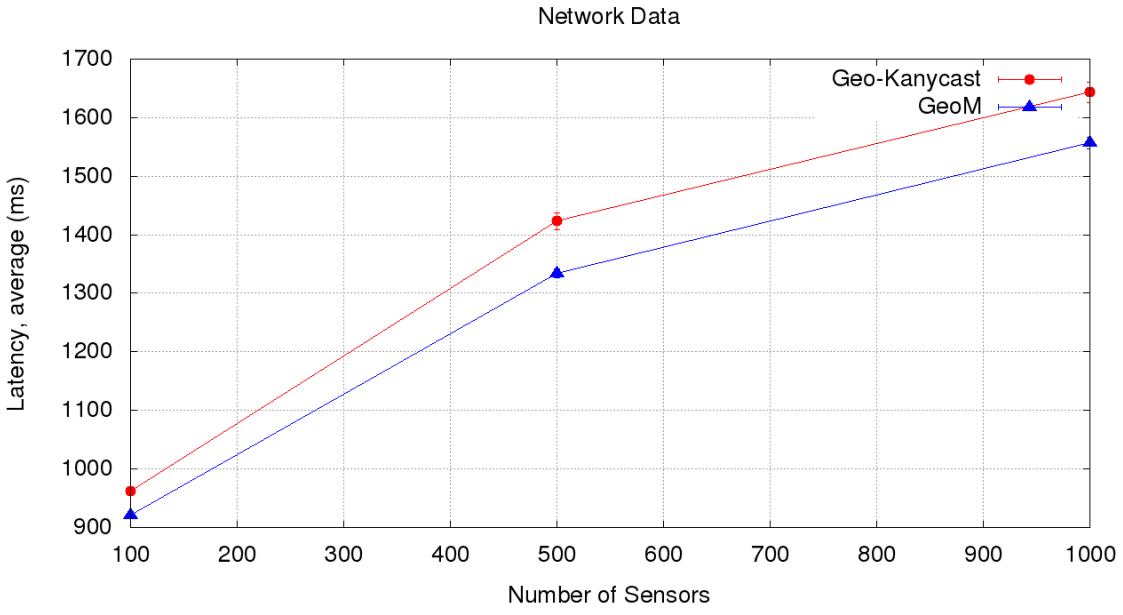


Figure 9.16: Average packet latency - results for networks composed of 100, 500, 1000 sensor nodes and 10 sinks (without void areas)

For the maximum energy consumption, Figure 9.17 shows that GeoM becomes better than GeoK with the network increase. The number of sinks is not proportional to the number of nodes. Because of that, with the increase of the network, we also increase the possibilities of different paths towards the sinks. Since GeoM searches for common paths instead of sending packets to the closest sink, when we have much more path options, GeoM is capable of switching paths more frequently. We can see in Figure 9.18 the evolution of the energy consumption for the node that consumed the largest amount of energy. In Figure 9.19 we present the distribution of the energy consumption at the end of the simulation.

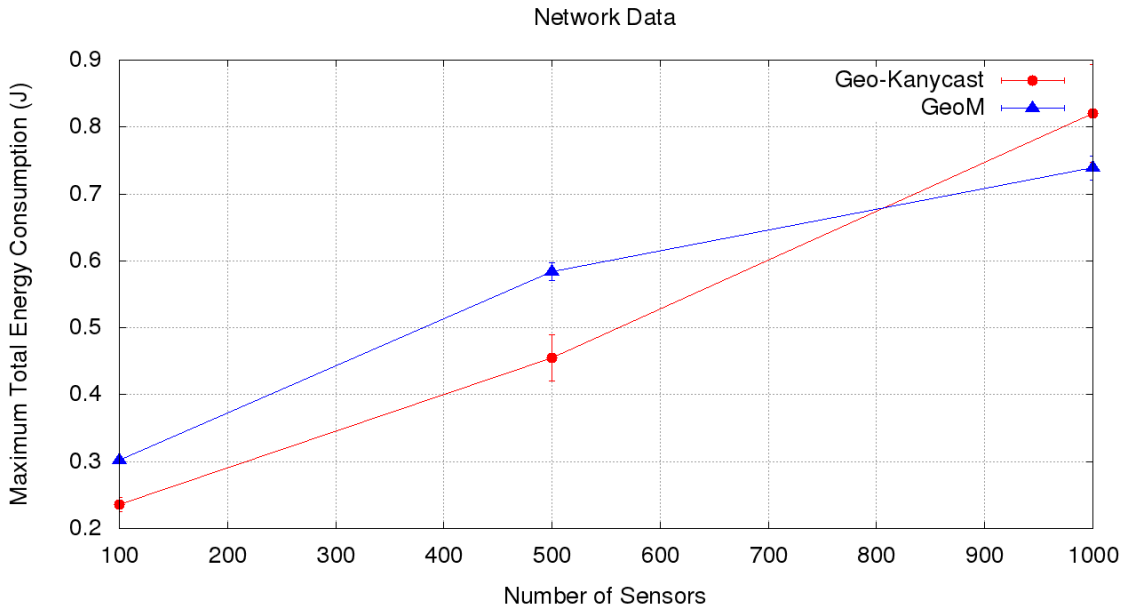


Figure 9.17: Maximum energy consumption - results for networks composed of 100, 500, 1000 sensor nodes and 10 sinks (without void areas)

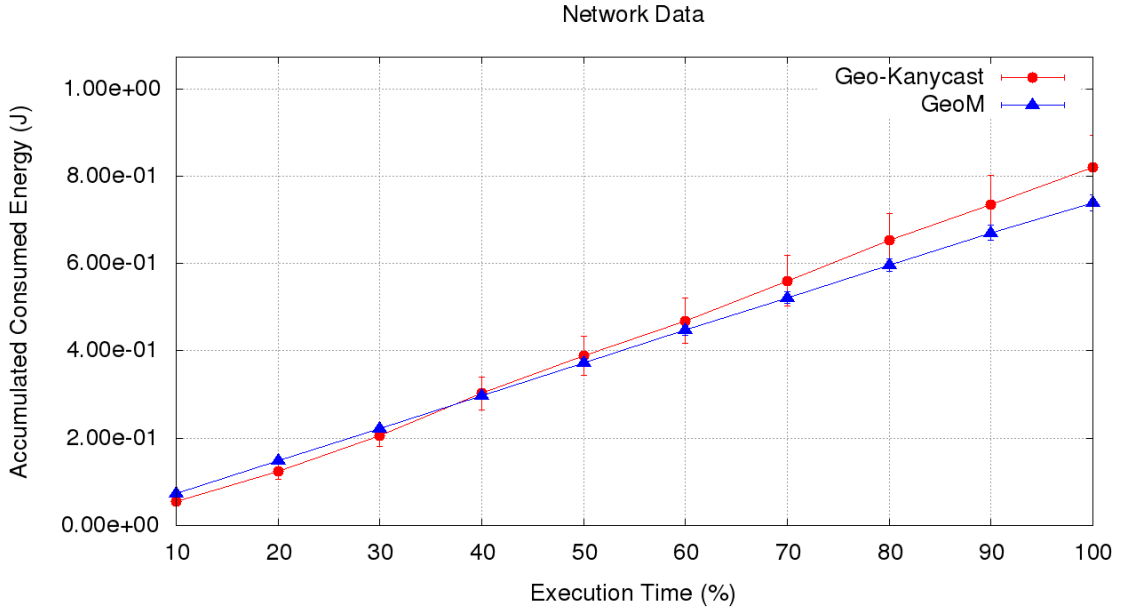


Figure 9.18: Evolution of the maximum consumed energy - results for networks composed of 1000 sensor nodes and 10 sinks (without void areas)

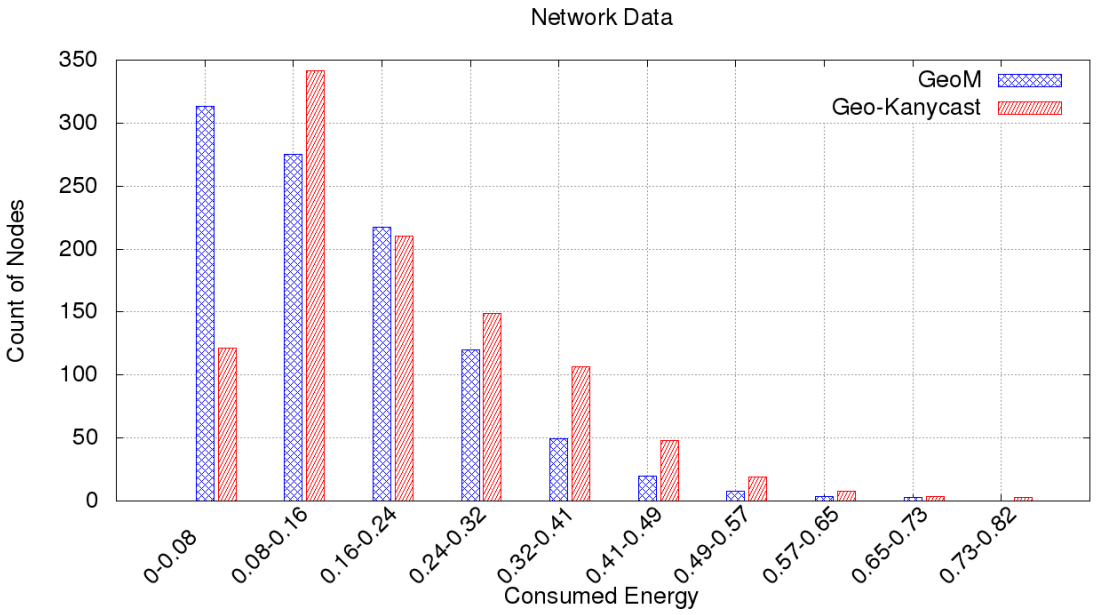


Figure 9.19: Distribution of nodes in terms of consumed energy at the end of the simulation - results for networks composed of 1000 sensor nodes and 10 sinks (without void areas)

9.4/ CONCLUSION

This chapter presented a new Geographic Multicast (GeoM) routing solution for wireless sensor networks with multiple sinks. The main goal was to find a balance between latency and network lifetime optimizations. Our strategy makes use of weighted metrics to establish the list of forwarder candidates, and search for intersections among the sinks and candidates in order to reduce the number of duplications. We tested our solutions through simulations using COOJA simulator. The simulation results from GeoM were compared to another geographic routing strategy capable of forwarding packets to multiple sinks. The obtained results indicate that our solution has an overall better performance than the existing protocol, with maximum gains of approximately 11% for latency and 54% for the maximum energy consumption.

10

REAL-LIFE EXPERIMENT

Contents

10.1/ MOTIVATION

The execution of real-life experiments provides meaningful insights on the performance of wireless sensor network solutions. Along with simulations, it is an important method in the process of validating a new protocol. From one side, the simulation provides the total control of the testing environment and the freedom to extrapolate the testing scenarios, necessary to understand the behavior of the solution and evaluate its performance. On the other side, the real-life experiment brings forward the behavior of the solution in real conditions, with all the constraints and problems related to the wireless communication, the device electronics and the processing and memory capacity of the mote.

Real-life experiments in WSN have been growing in importance and presence as evaluation method in the past years. A study performed on published papers of four important conferences in ad hoc and wireless sensor networks from 2008 to 2013 shows that the number of works relaying on real-world experiments as evaluation method is exceeding the ones that used simulations exclusively [107]. Nevertheless, the scope of the study considered all sorts of Ad Hoc and WSN solution.

Based on our investigation in the state of the art (Part II of this document), after narrowing the filter to identify MS-WSN solutions with real-life results, the scenario is much different. Only five solutions dealing with MS-WSN presented real-life experiments, and none of them considered multicast or k-anycast communication schemes, as shown in Table 5.3 in Chapter 5.

Our focus in this chapter is to investigate the feasibility of our solution in a real-life environment, using the FIT IoT-Lab platform [7]. We execute real-life experiments with our

approach GeoM, a multicast geographic routing protocol, focused on latency and network lifetime optimizations. We analyze its performance and confront the real-life results against simulations. This comparison is part of the feasibility study and a validation of the simulation results presented in Chapter 9. Additionally, we compare the real-life results from GeoM with another existing protocol. We use KanGuRou, a k-anycast geographic routing protocol, as comparison reference. Both, GeoM and KanGuRou, were developed using Contiki OS and adapted to the WSN430 mote, the sensor node used for the execution of the real-life experiments.

10.2/ Code porting and experimental platform description

Our objective is to evaluate the performance of our solution for multicast communication scheme (GeoM) through real-life experiments. For that, we make use of the FIT IoT-LAB testbed [7]. The FIT IoT-LAB is a large scale infrastructure, composed of testbeds varying in size and mote types in several locations in France. The platform comprises a total of 2071 wireless sensor nodes of five different types, distributed over six locations. For the real-life experiments, we used the WSN430 Texas Instrument C1101 mote available in the FIT IoT-LAB testbed of Strasbourg, with a total deployment of 256 nodes, placed in a 3D-grid. The WSN430 has an MCU of 16-bit, with a ROM of 48KB and a RAM of 10KB [7]. The performance evaluation comprises a comparison test between GeoM and KanGuRou [56], both adapted to the WSN430 mote.

10.2.1/ Code adaptation

As presented in Chapter 9, we developed GeoM in C as an application module for Contiki OS, as shown in Figure 10.1. One of the advantages of the Contiki OS is its layered design, enabling the porting of a solution to different hardware platforms.

```
contiki - {root}
├── apps - {where the applications are implemented}
│   ├── geo-kanycast - {GeoK protocol}
│   ├── geo-multicast - {GeoM protocol}
│   └── kangurou - {KanGouRou protocol}
├── core - {Contiki core source}
│   └── net - {all the network related files}
│       ├── mac - {all MAC implementations}
│       └── rime - {Rime stack implementation}
├── cpu - {specific adaptation code for different CPUs}
├── examples - {solutions using the protocols and applications}
│   ├── geo-kanycast_app - {GeoK application example}
│   ├── geo-multicast_app - {GeoM application example}
│   └── kangurou_app - {KanGouRou application example}
├── tools - {environment, cpu-specific, testing tools}
│   └── cooja - {simulator (in JAVA)}
│       ├── build - {log information when running in GUI mode}
│       └── dist - {log information when running in no-GUI mode}
└── platform - {specific adaptation code for different platforms}
```

Figure 10.1: Contiki OS, file structure

The initial implementation of GeoM considered the COOJA mote, which is the native mote for the COOJA simulator. The COOJA mote simulates a sensor node, but it does not have the same restrictions of a real mote in terms of memory and processing capacity. Few adaptations were necessary in order to use GeoM and KanGuRou with the WSN430 mote.

Because of the memory constraints of the WSN430 mote (RAM size of 10 KB and ROM of 48 KB), it was necessary to redesign the protocols structures in order to cope with the limitations. Instead of using direct access matrices and arrays, we implemented reusable sparse matrices. The problem of dynamic memory allocation in limited devices is related to the fragmentation of the memory and the risk of causing segmentation faults when freeing the allocated memory. Because of that, it was necessary to reserve parts of the memory to allow the size increase of the variables according to the execution need.

Another issue that was addressed is related to the float points, which are not well handled by the mote. The monitored metrics had to be re-scaled to avoid printing float numbers into log files. The timer precision based on the mote's clock was also adapted, since it was much less precise than the cooja-mote (native mote type of the COOJA simulator). The clock precision is important for the latency calculation.

Our solution assumes bidirectional communications, which is not guaranteed in the FIT IoT testbed. For that reason, it was necessary to implement a process of only accepting neighbors based on the bidirectional communication.

10.2.2/ EXPERIMENT LAUNCHING

The FIT IoT-Lab platform covers several locations in France. In order to use the testbed, two methods are available: the web interface and the remote access. Experiments may be launched and controlled remotely.

Web interface. Launching an experiment using the Web interface requires two different environments, the local machine, at the user side and the remote server, at the FIT IoT-Lab side. The basic flow is described in Figure 10.2. Mostly, the user interacts with the web interface, with some minor interactions with the shell of the remote server. The first step is related to the download of the source code of the application to be tested at the FIT IoT-Lab server. The remote machine has the correct configuration for the compiler environment, and the Contiki porting for the WSN430 mote. Once the binary is ready, it is necessary to download it to the local machine and upload it again to the server through the web interface. At this point, the experiment can be launched. If additional configuration for the test execution is necessary, it is possible to change the source code and update the firmware of the motes. The interface allows the user to create new experiments, launch and pause existing experiments, change the firmware of the motes and stop the execution. It is also possible to select specific motes to have a different firmware of the other motes in order to execute special tasks or to assume different roles in the network.

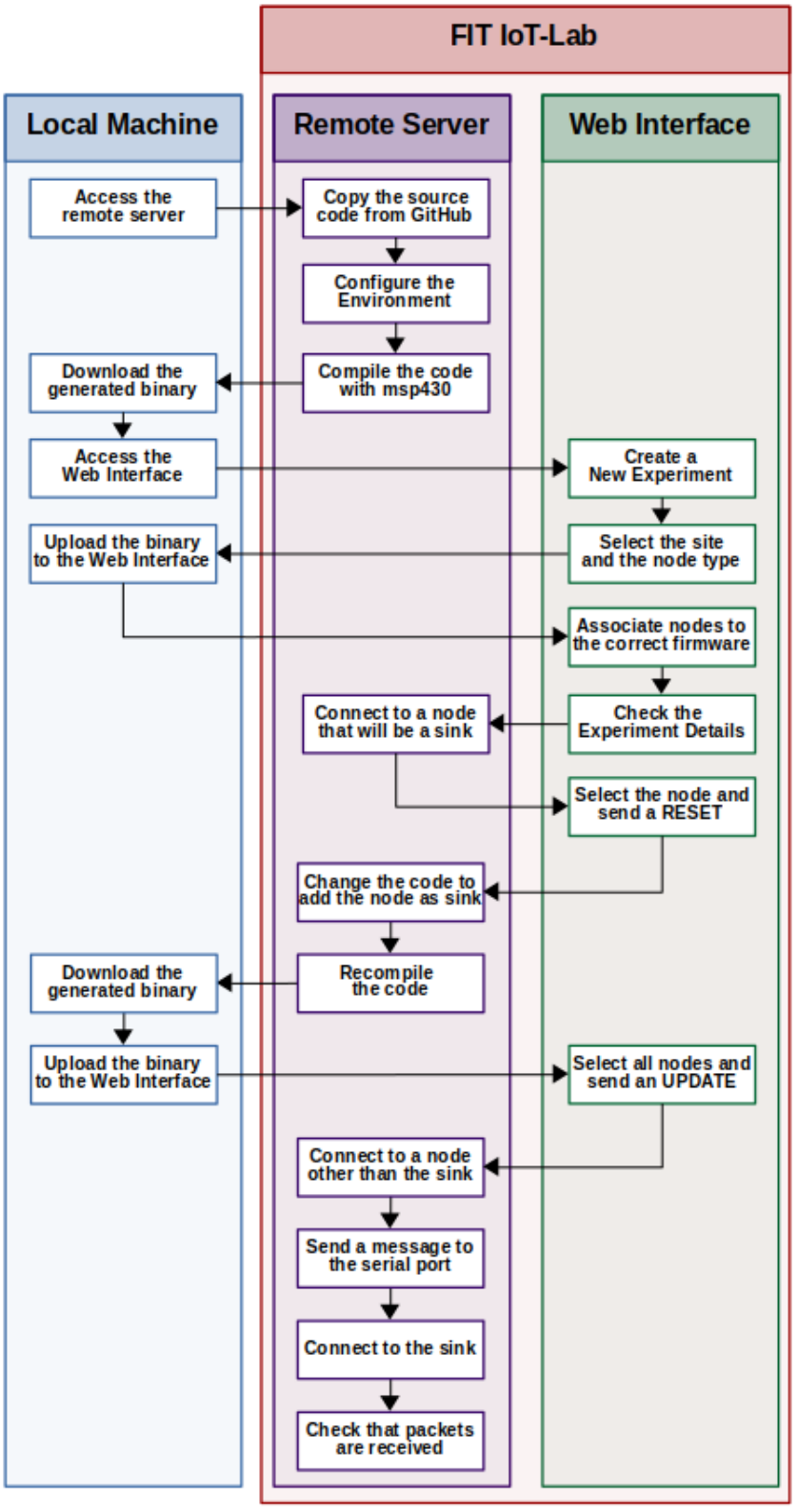


Figure 10.2: FIT IoT-Lab experiment launching with the web interface

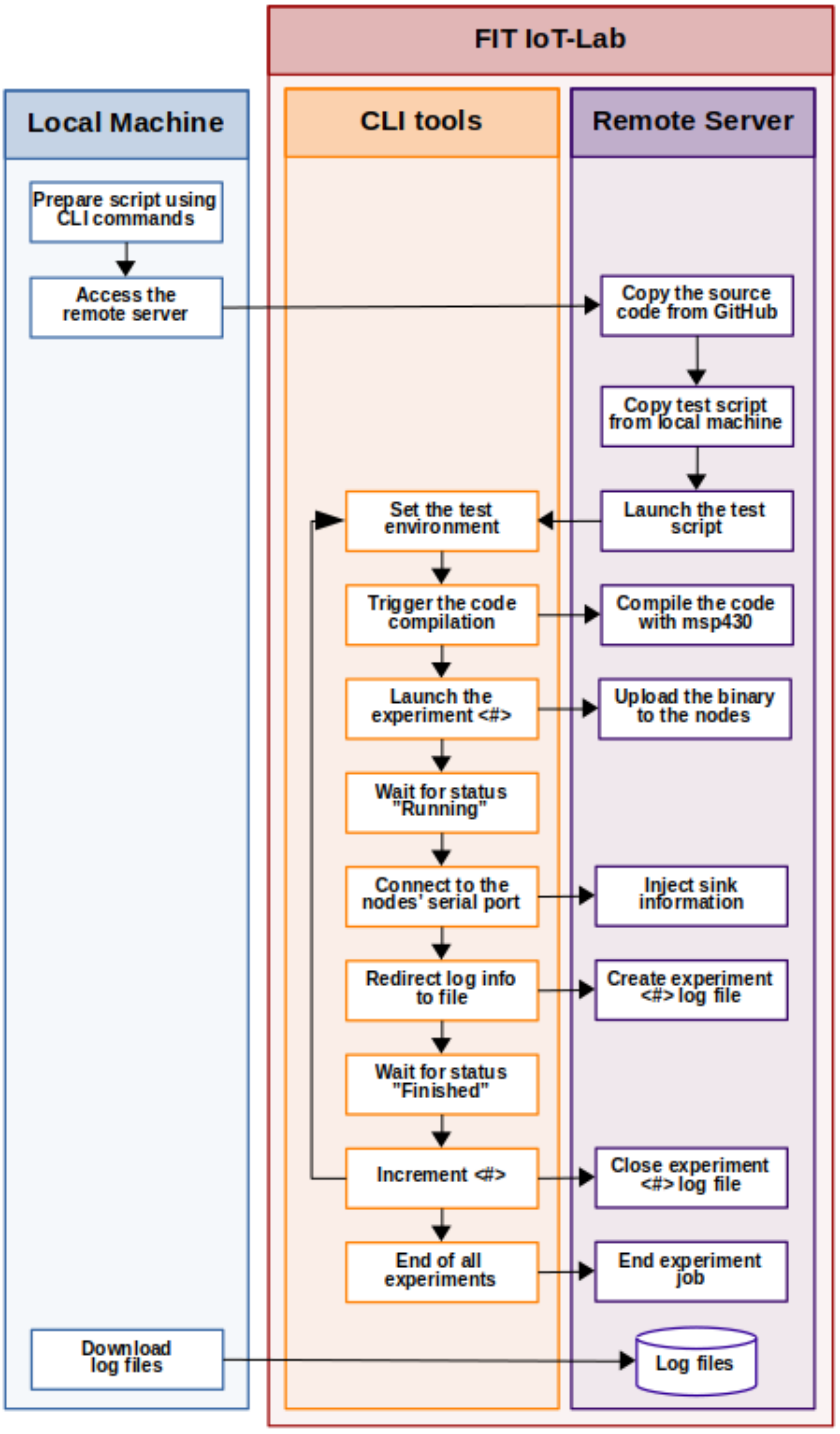


Figure 10.3: FIT IoT-Lab experiment launching with the CLI tools

Remote shell. FIT IoT-Lab proposes the CLI tools to connect with the remote server and launch experiments directly from the terminal using scripts. All the steps executed with the help of the web interface are possible with the CLI tools, as shown in Figure 10.3. With the scripts it is possible to launch different experiments with a particular scenario for each one. The script monitors the execution and launches a new experiment at the end of the current execution. If a problem is detected during the execution, it is possible to create check points that allows the relaunch of the same experiment. A check point may consist

of parsing the motes' output, in order to detect if they were correctly initialized. It is also possible to redirect the log information to a file. The log information is composed of all the data extracted from the serial port of each mote. Because of that, it is possible to generate log outputs for the routing protocol in order to debug and evaluate the performance of the solution.

10.3/ EXECUTION

Both GeoM and KanGuRou were tested under the same conditions and configurations. We analyze not only the real-life results, but also the results from the simulation with the same configurations of the real-life experiments. The objective of executing new simulations is related to the possibility of comparing the obtained results and verify if the trends identified in the real-life experiments are also observed in the simulation. The information in Table 10.1 summarizes the simulation and experimentation settings.

The evaluation considers the average of all metrics for all executions, and the results are presented with a confidence interval of 95%. We analyze the latency, defined by the average time a packet takes to be routed from the source to all sinks. Every sensor node is considered a source node and generates packets. We also look at the hop count, which is the average number of hops from the source node to all sinks. We analyze the maximum energy consumption, which regards the node that consumed the largest amount of energy in the network at the end of the execution. It gives an indication of the network lifetime, since it represents the node's battery status. The network lifetime is considered to be the earliest moment at which a node's battery is completely depleted.

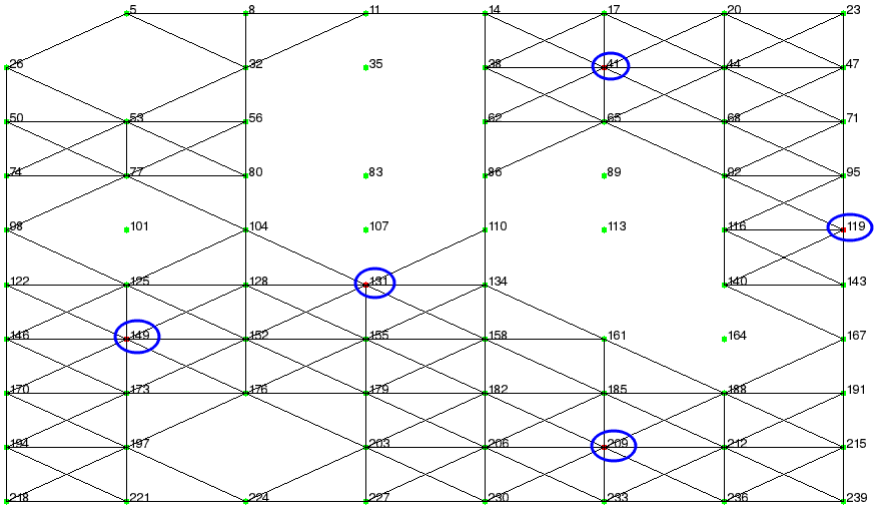
Table 10.1: Simulation and Experimentation settings

	Cooja Simulator	FIT IoT-LAB Testbed
Network Area	$270m \times 270m$	$10m \times 8m$
Deployment	Random Grid	Grid
Executed Networks	100	50 and 20
Mote Type	Cooja Mote	WSN430 TI CC1101
Radio Range/Power	$50m$	$-10dBm$
Execution Time	30 min	30 min and 6 hours
# of Sensors	63 nodes	
# of Sinks	5 nodes	
# of Neighbors	8 nodes (maximum)	
Packet Size	100 bytes	
Packet Generation	20% chance at every minute for each sensor	
Radio Type	802.15.4	
MAC Protocol	CX-MAC, modified version of [106]	

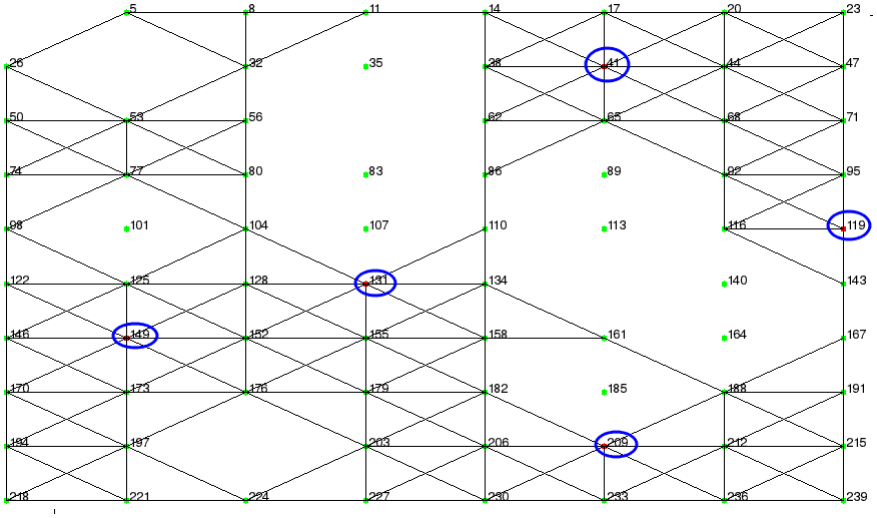
10.3.1/ REAL-LIFE RESULTS

As presented in Table 10.1, we performed 50 executions of about 30 minutes for each solution, which represent 50 hours of test in total for the short run, and 20 executions of about 6 hours, for the long run, which represent 240 hours of testing. However, the

number of exploitable results is reduced. For the short run, only 30 executions out of the 50 for each solution could be used. And for the long run, only 10 executions out of the 20 experiments could be exploited. Despite the fact that FIT IoT-Lab has a controlled testing environment, it is not possible to assure that all links have at all moments the desired quality to create the expected logical network topology. In many cases, the links are not bidirectional, which impacts the execution of the solutions. The bidirectional communication is an important hypothesis for both GeoM and KanGuRou solutions.



(a) Execution number 1



(b) Execution number 2

Figure 10.4: Physical deployment and logical topologies

For the execution of the experiments, we considered only the nodes in one of the planar deployments of the FIT IoT-Lab testbed in Strasbourg. The node deployment in the mentioned testbed forms a grid based physical topology that produces a completely connected network, with a node being able to communicate with all other nodes. It happens because the deployment area is considerably small, with the nodes organized in an surface of $10m \times 8m$. The transmission range of all nodes cover the entire area. Because we needed longer paths to validate the routing strategy, we defined a different logical

topology, by limiting the neighborhood of a node to its immediate physical neighbors. The new logical topology created enough hop-distance between sensors and sinks to test the routing protocol.

Despite the fact that the nodes are fixed and the physical topology is the same, at runtime we may have different networks. The links among nodes vary from one execution to another due to interference and faulty nodes. This phenomenon results in slightly different networks, due to changes in the neighborhood. As for instance, Figures 10.4a and 10.4b show the links for two different executions of the same predefined physical topology. We can see the grid-deployment of nodes with the sinks highlighted with circles.

10.3.1.1/ SHORT RUN

As we can see in Figure 10.5, GeoM presents a higher maximum energy consumption compared to KanGuRou. However, this is the result of an elevated packet loss for KanGuRou. As displayed in Figure 10.6, KanGuRou has a packet loss of roughly 70%, almost the double of the packet loss presented by GeoM. It means that for GeoM there are more packets being forwarded, and consequently a higher level of energy consumption. On the other hand, the average hop count for GeoM is a bit lower compared to KanGuRou, as seen in Figure 10.7. It is explained by the fact that GeoM decides to duplicate the packets slightly earlier in order to enable a fast progress towards the destination sinks.

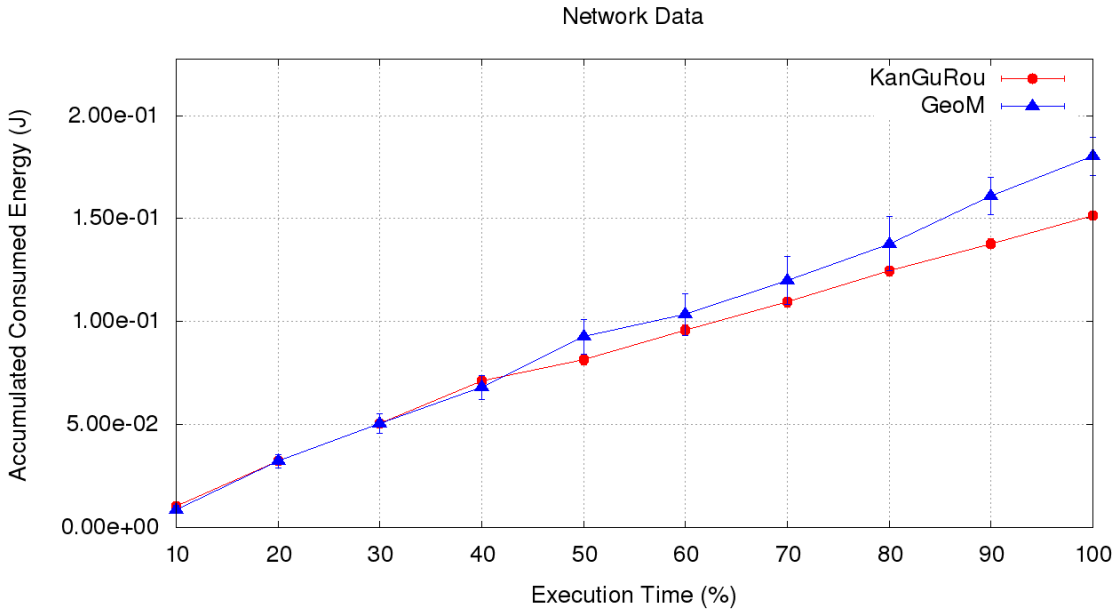


Figure 10.5: Energy consumption over time for the node that consumed the maximum energy at the end of the execution (identical MAC configuration)

The huge difference in packet loss is also explained by the existence of faulty links, which heavily affects KanGuRou in comparison to GeoM. KanGuRou bases its routing decisions on the distance and the transmission energy cost. With this strategy, the same routes are used over and over again. If faulty links are present in some routes, it is enough to drastically increase the packet loss. In GeoM, routes may change from one transmission

to another, due to the metric related to the energy consumption. In the case of a faulty link, GeoM is less impacted. Moreover, since GeoM duplicates the packets slightly earlier, it means that each duplicated packet has a smaller group of destination sinks. As each generated packet must be delivered to all sinks, if the packet is lost before the duplication, the impact on the packet loss is much higher.

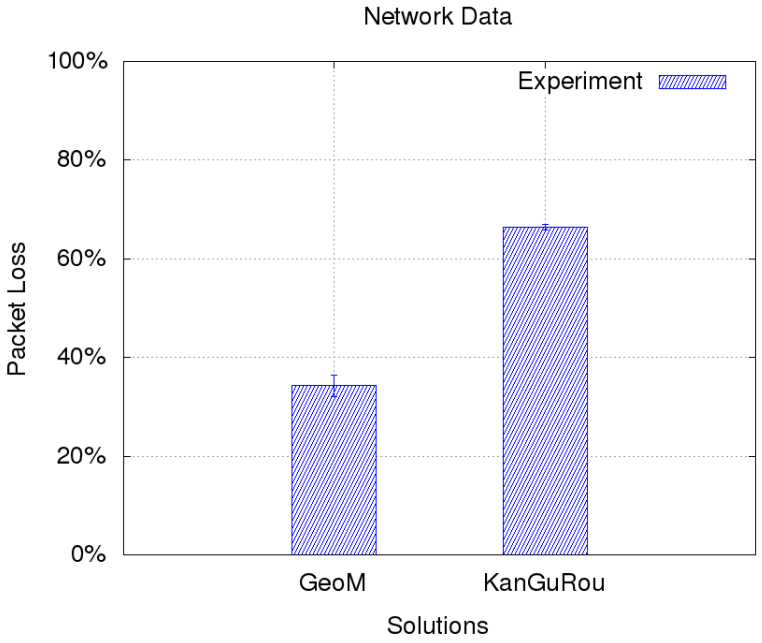


Figure 10.6: Packet loss (identical MAC configuration)

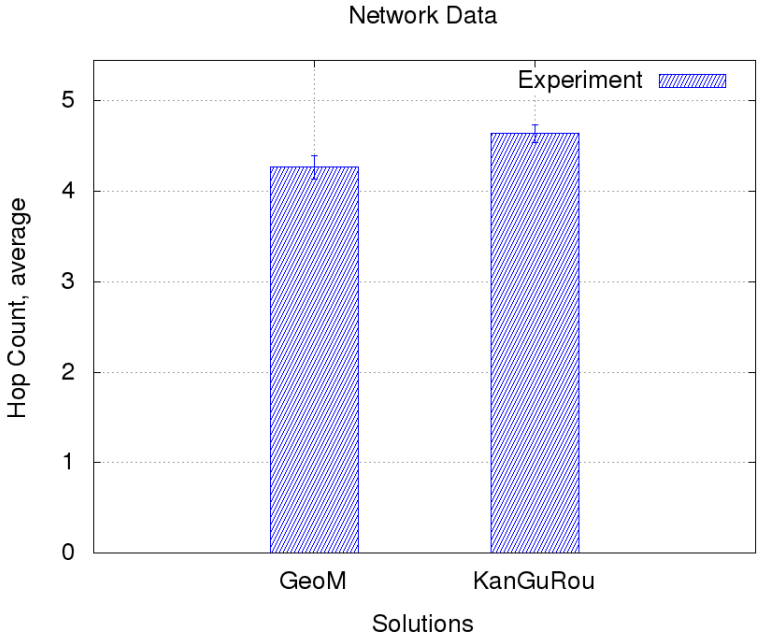


Figure 10.7: Average hop count (identical MAC configuration)

In order to have a better comparison point for the maximum energy consumption, we re-executed the experiments for GeoM using a different MAC configuration. We modified the MAC configuration related to transmission acknowledgments and retries. The new con-

figuration ignores the acknowledgment phase, and does not wait for the confirmation of the reception, consequently no transmission retry is executed. The applied modifications meant to provide a similar packet loss compared to KanGuRou, so the maximum energy consumption could be analyzed under the same condition in terms of number of packets being forwarded. As we can see in Figure 10.8, GeoM had a similar performance in terms of packet loss when using a different MAC configuration.

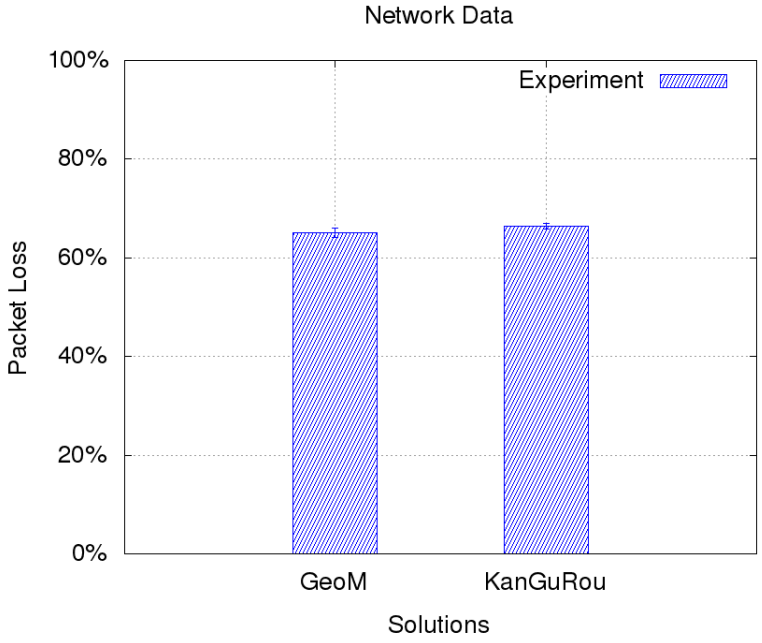


Figure 10.8: Packet loss (GeoM with different MAC configuration)

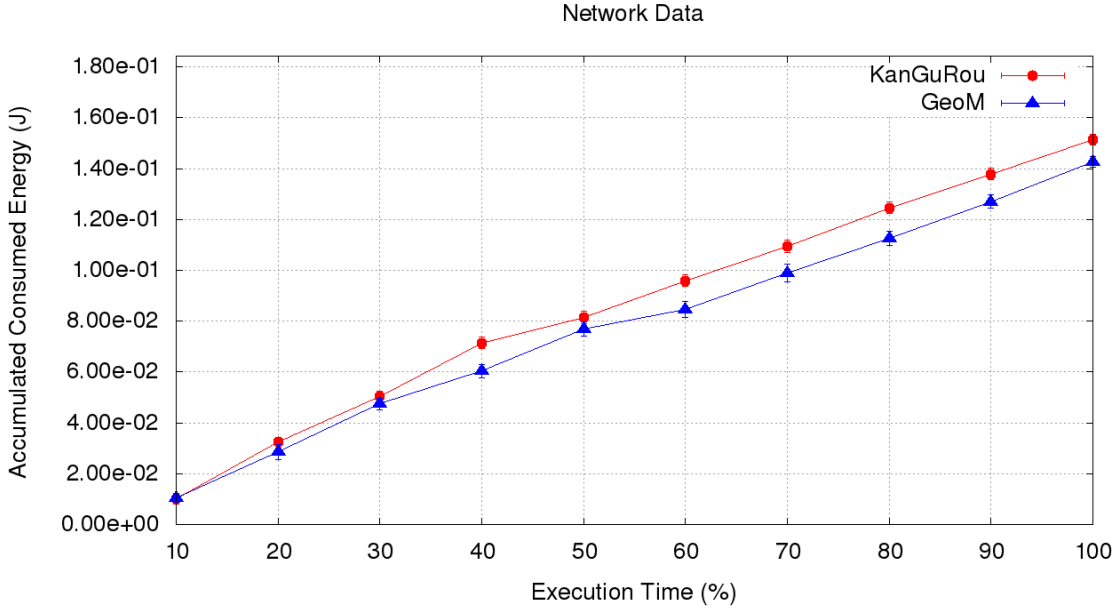


Figure 10.9: Energy consumption over time for the node that consumed the maximum energy at the end of the execution (GeoM with different MAC configuration)

Considering the alternative MAC configuration for GeoM, we can see in Figure 10.9 that

the results for the energy consumption over time for the node that consumed the maximum energy at the end of the execution show a different behavior. This time, GeoM presents a marginally lower consumption of 5.8% compared to KanGuRou. It means that under a similar packet delivery rate GeoM would be able to extend the network lifetime. However, under identical MAC configurations, GeoM has a better QoS compared to KanGuRou.

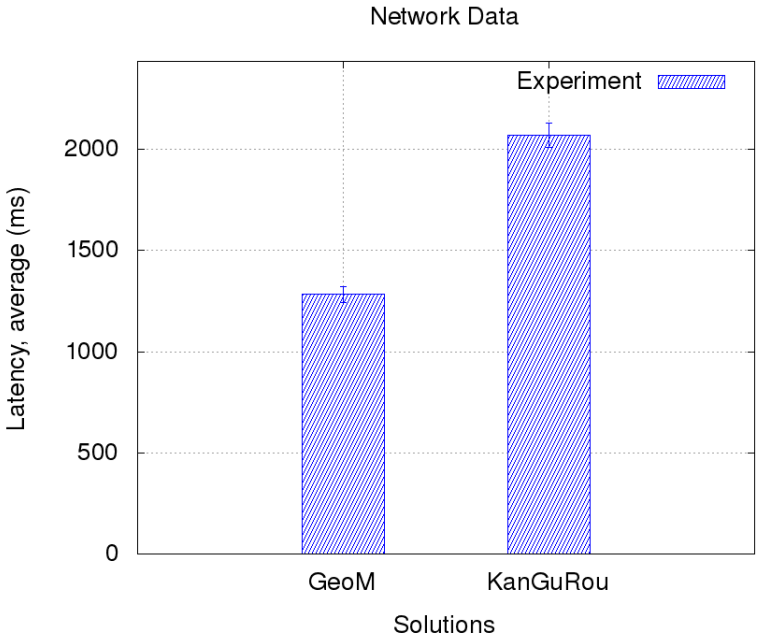


Figure 10.10: Average latency (identical MAC configuration)

When we analyze the latency results for the identical MAC configuration, we can see in Figure 10.10 that GeoM is able to reduce the average latency by approximately 38%. As already mentioned, GeoM moves the packet towards the sinks in a faster way, since duplication takes place slightly earlier, which translates to fewer hops, and consequently a reduction of the average latency.

10.3.1.2/ LONG RUN

For the long run, the maximum energy consumption results for GeoM and KanGuRou are almost equivalent, shown by Figure 10.11. The packet loss is virtually the same, as seen in Figure 10.12. Since both solutions executed during six hours, the packet loss stabilized at approximately 60%. Unfortunately, due to faulty nodes, none of the packets could be delivered to all sinks.

The energy consumption over time for the node that consumed the maximum energy at the end of the execution shows that both solutions have the same behavior, as in Figure 10.13.

The average latency result shows a much better performance for GeoM in comparison to KanGuRou, with a performance gain of approximately 65%, as presented in Figure 10.14. Once again, it is explained by the fact that the hop count in GeoM is smaller, due

to the packet duplication at an earlier stage.

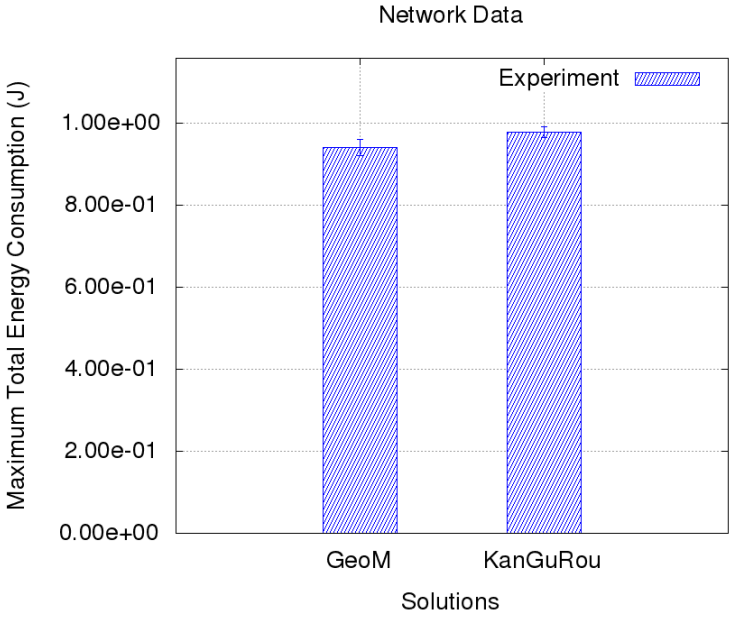


Figure 10.11: Maximum energy consumption (long run)

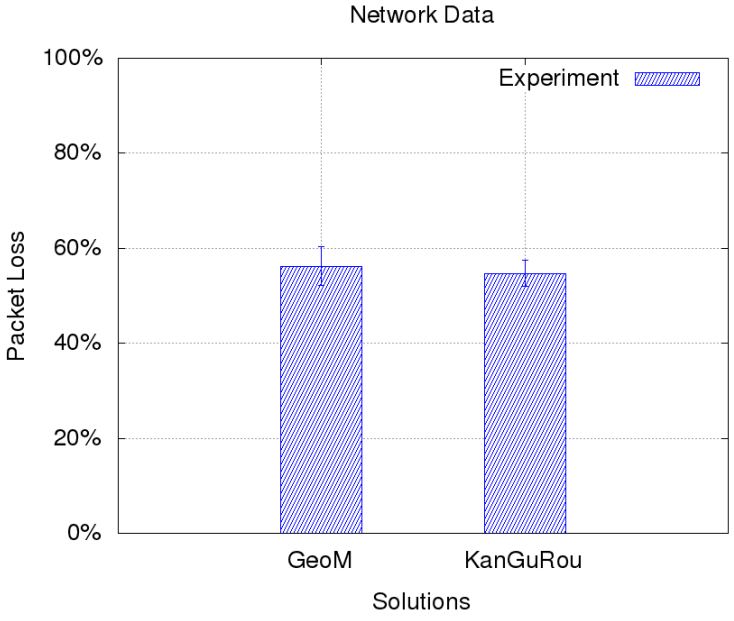


Figure 10.12: Packet loss (long run)

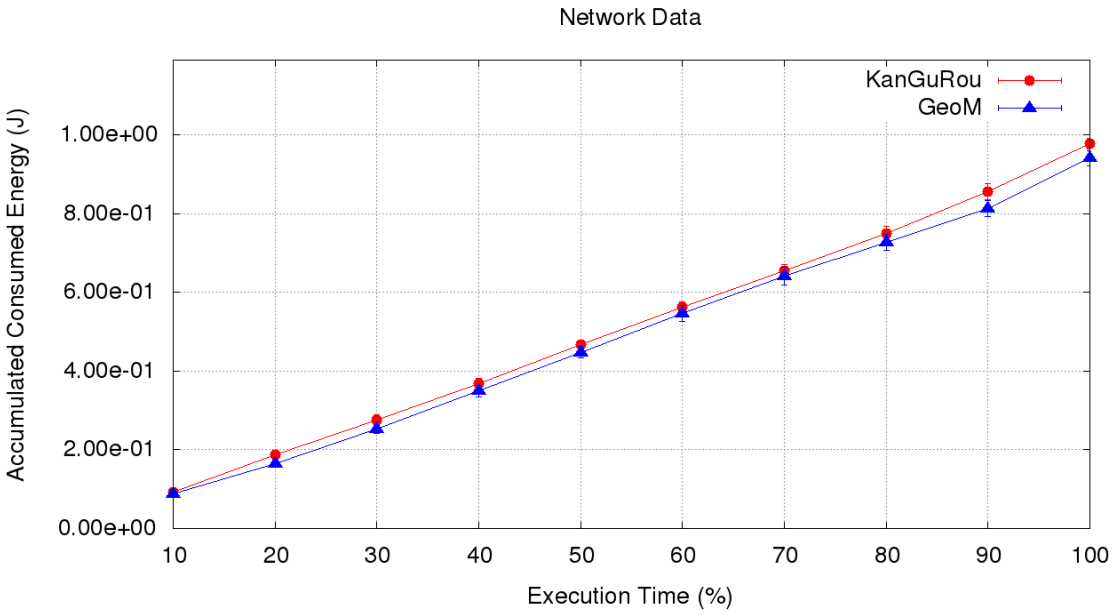


Figure 10.13: Energy consumption over time for the node that consumed the maximum energy at the end of the execution (long run)

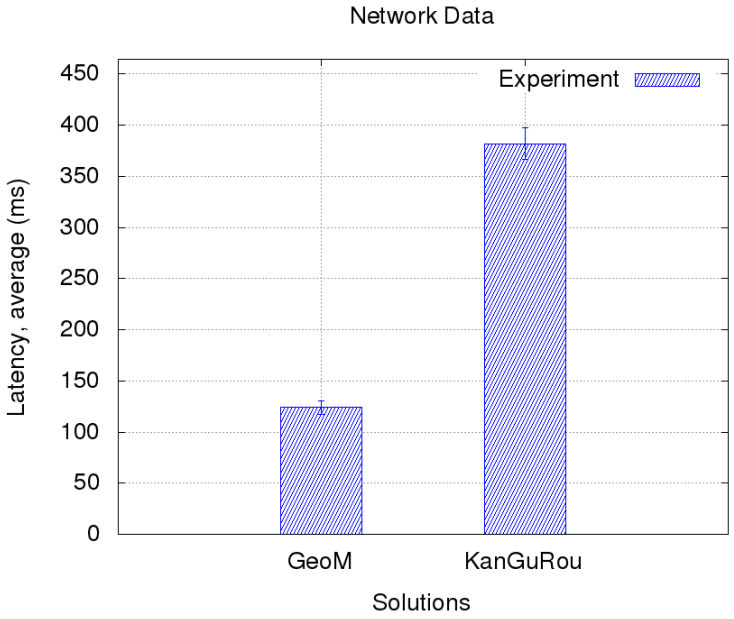


Figure 10.14: Average latency (long run)

10.3.2/ SIMULATION RESULTS

The real-life experiment is limited in terms of network size and physical deployment. Moreover, the faulty nodes and links bring a high level of unpredictability of results for the experiments. The real-life results show a marginal performance gain in terms of

maximum energy consumption, which is mostly a consequence of the small size of the network and the regularity of the physical deployment. In that sense, it is important to identify a correlation between the real-life experiments and the simulations, in order to overcome the limitations of the testbed and extrapolate the testing scenarios. To do so, we tried to create and execute simulation scenarios that would be close to the physical network.

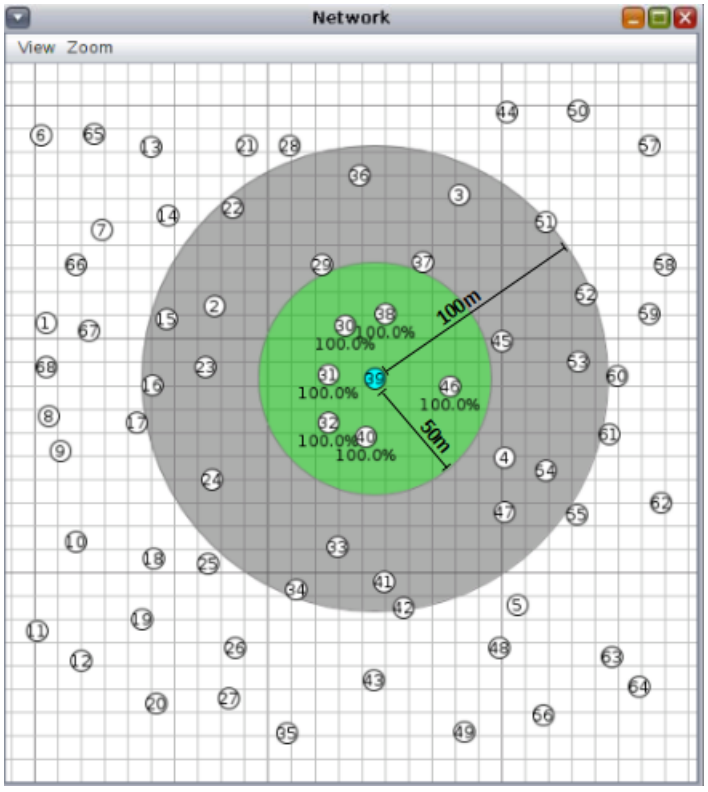


Figure 10.15: Physical deployment for the simulation

For the simulations, 100 different networks with 63 sensor nodes and 5 sinks were randomly generated. Figure 10.15 presents one such randomly generated network. Similarly to the real-life experiment, each node has approximately 8 neighbors in its communication range. However, the link quality and the interference level are not the same. The communication range is 50m with a bidirectional link and the interference range is of 100m. The simulation environment is much more stable compared to the real-life experiment. This stability can be evidenced by the packet loss results, as seen in Figure 10.16. GeoM and KanGuRou share similar packet loss results for the simulation. Both solutions are executed with the same MAC settings.

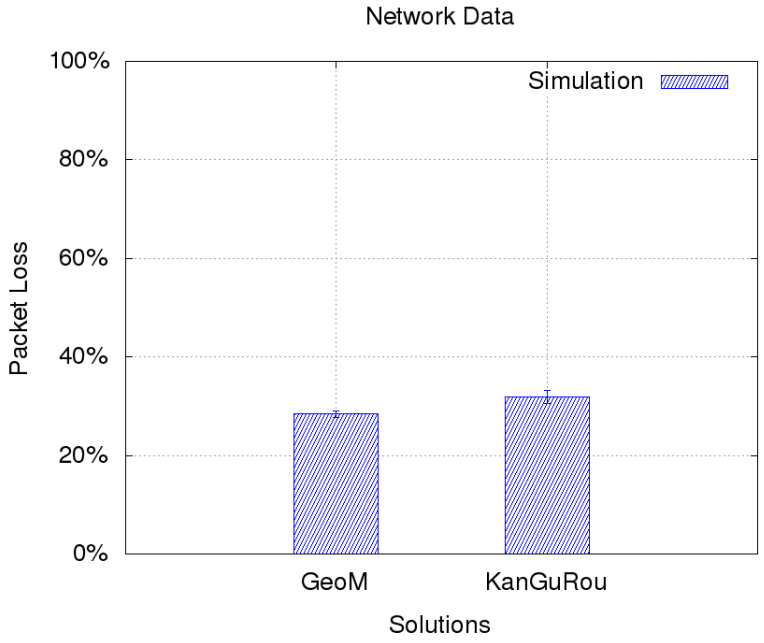


Figure 10.16: Packet loss (simulation)

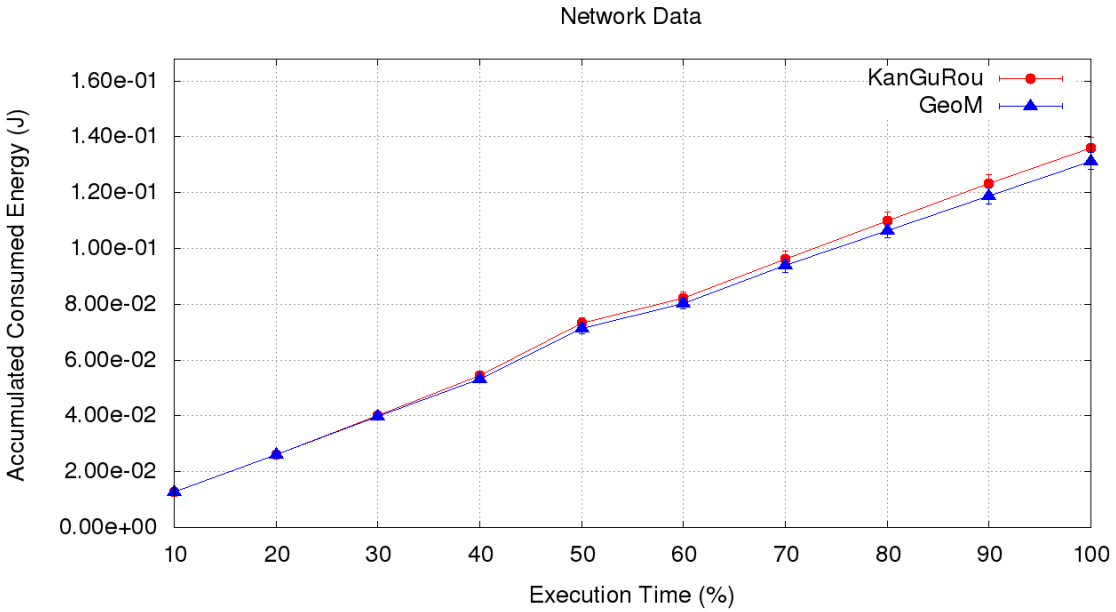


Figure 10.17: Energy consumption over time for the node that consumed the maximum energy at the end of the execution (simulation)

Because of the stable environment, we are able to compare the maximum energy consumption with the identical MAC configuration. We can see in Figure 10.17 that GeoM and KanGuRou have almost the same evolution of the energy consumption for the node having the most depleted battery level by the end of the simulation. This is the expected behavior for a network with 63 nodes and 5 sinks. In Chapter 9, GeoM was validated with

different network sizes, going from 50 up to 1000 sensor nodes. It was noticed that GeoM presents much better performances in terms of maximum energy consumption for larger networks, with gains from 5% up to 26% for networks with void areas and from 23% up to 53% for networks without void areas. The comparison with the real-life results allows us to conclude that the performance presented by GeoM is consistent with the results obtained from the simulation.

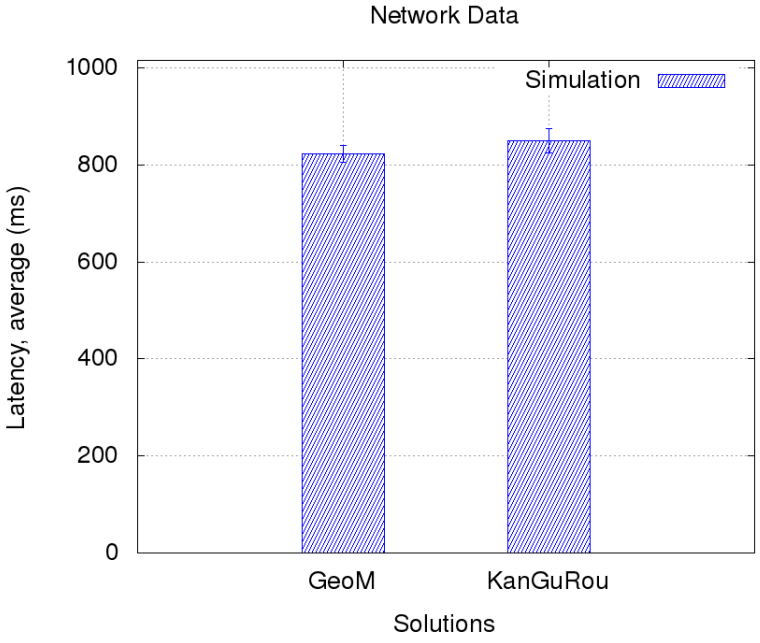


Figure 10.18: Average latency (simulation)

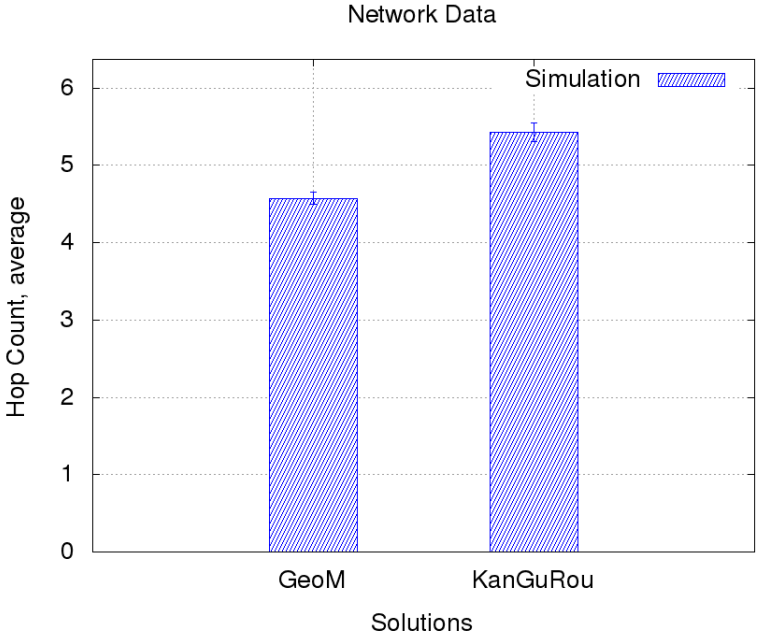


Figure 10.19: Average hop count (simulation)

The same comment applies for the results related to the average latency. We can see in Figure 10.18 that GeoM and KanGuRou have similar results, with a difference of 3.3%. However, for the simulations presented in Chapter 9, with larger networks, we can observe better results for GeoM in terms of average latency, with performance gains of 10%. We can observe in Figure 10.19 that the average hop count for GeoM is smaller compared to KanGuRou, which for the routing perspective normally results in lower latency.

10.4/ CONCLUSION

This chapter presented the real-life results for the Geographic Multicast (GeoM) routing protocol in comparison to KanGuRou. The results were obtained with experiments carried out within the FIT IoT-Lab platform with the WSN430 mote type. Additionally, new simulations were executed considering the characteristics of the real-life experiments in order to establish a comparison point between the real-life results and the simulations. All simulations were executed with COOJA simulator.

Both GeoM and KanGuRou solutions were analyzed under the same criteria. The results show that GeoM has a better performance in comparison to KanGuRou in terms of latency, hop count, packet loss and maximum energy consumption. The correlation between the real-life experiments and the simulations shows that the results are consistent and comparable.

CONCLUSION, PERSPECTIVE AND PUBLICATIONS

CONCLUSION

In this work we were interested in the communication optimization in multi-sink wireless sensor networks. We worked on three main issues of the WSN: longevity (network lifetime), reliability (packet delivery) and timeliness (latency). We identified the specificity connected to the existence of multiple sinks and proposed a classification category for existing works in MS-WSN. We also proposed three strategies for packet routing considering different communication schemes (unicast, anycast and multicast).

Our first contribution took aim at the latency issue in time sensitive communications. We proposed MAC-aware routing protocols for time-constrained applications, with packets being preventively routed through the lowest latency paths using unicast communication scheme.

The second contribution addressed the optimization of the latency and the network lifetime along with the increase of the reliability through a k-anycast protocol. We proposed a geographic routing protocol that considers a linear combination of network metrics during the decision process of the next hop. Packets are forwarded over exactly k sinks, with targets and duplications being defined on the fly.

Our third contribution is a geographic routing protocol that is capable of forwarding a generated packet to all sinks in the network, increasing the reliability, while reducing the latency and balancing the energy consumption. The algorithm is divided into three phases: filtering, selection and forwarding. The algorithm searched for common forwarders for the same group of sinks, in order to avoid the early duplication of the packets. Our solution does not require much knowledge of the network topology, but it uses broadcast messages to periodically advertise relevant information.

We can summarize our contributions in Table 10.2 using the classification criteria proposed in Chapter 3.

Comm. Scheme	*Path Strategy*	*Sink Decision*	*Structure*	*Packet Duplic.*	*Contributions*
Unicast	Single path	Fixed	Tree	No	DB-DD, DB-DD-FBC, DB-DD-DBC [108]
k-Anycast	Multi-path	On the way	Graph	On the way	GeoK [109]
Multicast	Multi-path	At source	Graph	On the way	GeoM [110]

Table 10.2: Summary of the contributions based on the routing solutions classification

We also presented real-life results for our third contribution. The execution of real-life experiments provided meaningful insights on the performance of our solution. Along with simulations, it was an important tool in the process of protocol validation. The tests were executed using the FIT IoT-LAB infrastructure in Strasbourg. The real-life results show that GeoM improves the network performance, presenting better results for the compared to an existing solution.

After all presented results, we can safely say that there are many particularities related to routing solutions for networks with multiple sinks. We can see from the results that the algorithms designed for a given communication scheme, even with multiple capabilities (as the k-anycast), do not always show the best results in particular scenarios (such as $k = |S|$). We could also see that the deployment of multiple sinks may also improve the reliability, depending on the routing strategy. And because of that, we can see the need of designing tailored routing protocols, capable of answering to the specificities of a wireless sensor network with multiple sinks.

PERSPECTIVES

During the development of this work we considered assumptions that helped us focus on finding solutions to specific problems. One of the assumptions was related to the bidirectional aspect of the communication links (symmetric links). We could identify that in real networks the bidirectional link is not always assured due to numerous reasons (obstacles, interference, differences in radio range etc). In that sense, we point out the importance of investigating MS-WSN solutions with heterogeneous radio ranges.

We also identified the importance of considering the link quality, interference and collisions during the routing decision, since they have a high effect on metrics such as latency and energy consumption. The routing metrics must account for the link quality in order to avoid delays and increase reliability.

In our work, for the latency optimizations, we were focused on minimizing the average end-to-end delay. This is important for time sensitive applications that require the information to be delivered as soon as possible. However, there is another class of applications that require a regular cadence of the delivery time. It means that the information must be delivered in a regular interval, with a precise schedule. An example of such application would be an industrial IoT solution for an assembly line, monitoring and triggering events at a regular pace. In this case, the challenge is related to the regularity of the delivery time, assuring that the data from each sensor node is delivered following a time schedule. In that sense, we believe that a solution for this problem would be a MAC-aware routing solution considering a k-anycast communication scheme with a contention-free MAC.

We also identify opportunities in mobile MS-WSN, where the sinks are stationary and the sensor nodes are mobiles. A routing solution could consider a 1-anycast communication scheme as the strategy for the packet delivery. In this case, the challenge would be the identification of the destination sink and the definition of the routing path within the moving sensor nodes. The advantage in using 1-anycast communication scheme is related to the fact that the destination is not fixed, allowing the packet delivery to any of the sinks.

LIST OF PUBLICATIONS

LEAO, Lucas, FELEA, Violeta, et GUYENNET, Herve. MAC-Aware Routing in Multi-Sink WSN with Dynamic Back-Off Time and Buffer Constraint. In : New Technologies, Mobility and Security (NTMS), 2016 8th IFIP International Conference on. IEEE, 2016. p. 1-5.

LEAO, Lucas, FELEA, Violeta. Latency and Lifetime Optimization for k-anycast Routing Algorithm in Wireless Sensor Networks. In : Ad Hoc Networks and Wireless (AdHoc-Now), 2018 17th International Conference on. Springer, 2018. p. 39-50.

LEAO, Lucas, FELEA, Violeta. Latency and Network Lifetime Trade-off in Geographic Multicast Routing for Multi-Sink Wireless Sensor Networks. In : Mobile, Secure and Programmable Networking (MSPN), 2018 4th International Conference on - presented.

LEAO, Lucas, FELEA, Violeta, et GUYENNET, Herve. Performances of Geographic Multicast Approach on Real-life Platform. Submitted to ”2019 Wireless Days (WD) - Ad Hoc and Sensor Networking” - in reviewing process.

The source code of the implemented approaches are available at:
https://gitlab.com/lucas.augusto/geo_multi-sink

BIBLIOGRAPHY

[1] RAWAT, P., SINGH, K. D., CHAOUCHI, H., AND BONNIN, J. M. **Wireless sensor networks: a survey on recent developments and potential synergies**. *The Journal of supercomputing 68*, 1 (2014), 1–48.

[2] ATZORI, L., IERA, A., AND MORABITO, G. **The internet of things: A survey**. *Computer networks 54*, 15 (2010), 2787–2805.

[3] PELLENZ, M. E., JAMHOUR, E., PENNA, M. C., SOUZA, R. D., AND BRANTE, G. **A power assignment method for multi-sink wsn with outage probability constraints**. In *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on* (2014), IEEE, pp. 533–540.

[4] BURATTI, C., CONTI, A., DARDARI, D., AND VERDONE, R. **An overview on wireless sensor networks technology and evolution**. *Sensors 9*, 9 (2009), 6869–6896.

[5] SCILAB ENTERPRISES. **Scilab: Free and open source software for numerical computation**. http://www.scilab.org/. Accessed: 2017-05-10.

[6] CONTIKI OS. **The open source os for the internet of things**. http://www.contiki-os.org/. Accessed: 2018-03-20.

[7] FACILITY, F. I. T. **FIT IoT-LAB**. https://www.iot-lab.info/. Accessed: 2018-03-20.

[8] TOWNSEND, C., AND ARMS, S. **Wireless sensor networks: Principles and applications**. In *Sensor Technology Handbook*, J. S. Wilson, Ed. Elsevier, Burlington, 2005, ch. 22, pp. 575–589.

[9] YICK, J., MUKHERJEE, B., AND GHOSAL, D. **Wireless sensor network survey**. *Computer networks 52*, 12 (2008), 2292–2330.

[10] AKYILDIZ, I. F., SU, W., SANKARASUBRAMANIAM, Y., AND CAYIRCI, E. **Wireless sensor networks: a survey**. *Computer networks 38*, 4 (2002), 393–422.

[11] MELODIA, T., VURAN, M. C., AND POMPILI, D. **The state of the art in cross-layer design for wireless sensor networks**. In *International Workshop of the EuroNGI Network of Excellence* (2005), Springer, pp. 78–92.

[12] L. LOUAIL AND V. FELEA. **Routing-aware TDMA scheduling for wireless sensor networks**. In *12th Annual Conference on Wireless On-demand Network Systems and Services (WONS)* (2016), IEEE, pp. 9–16.

[13] L. LOUAIL AND V. FELEA. **Latency optimization through routing-aware time scheduling protocols for wireless sensor networks**. *Computers & Electrical Engineering* (2016), 418–440.

[14] C. SUH, Y. KO AND D. SON. **An Energy Efficient Cross-Layer MAC Protocol for Wireless Sensor Networks**. In *APWeb Workshops* (2006), pp. 410–419.

[15] M. HEFEIDA, T. CANLI AND A. A. KHOKHAR. **CL-MAC: A Cross-Layer MAC protocol for heterogeneous Wireless Sensor Networks**. *Ad Hoc Networks 11*, 1 (2013), 213–225.

[16] C. CHOU AND K. CHUANG. **CoLaNet: A Cross-Layer Design of Energy-Efficient Wireless Sensor Networks**. In *ICW/ICHSN/ICMCS/SENET* (2005), pp. 364–369.

[17] K. HEURTEFEUX, F. MARANINCHI AND F. VALOIS. **AreaCast: A cross-layer approach for a communication by area in Wireless Sensor Networks**. In *ICON* (2011), pp. 112–117.

[18] L. LOUAIL, V. FELEA, J. BERNARD AND H. GUYENNET. **MAC-aware routing in wireless sensor networks**. In *International Black Sea Conference on Communications and Networking (BlackSeaCom)* (2015), IEEE, pp. 225–229.

[19] O. LANDSIEDEL, E. GHADIMI, S. DUQUENNOY AND M. JOHANSSON. **Low Power, Low Delay: Opportunistic Routing meets Duty Cycling**. In *IPSN* (2012), pp. 185–196.

[20] L. LOUAIL AND V. FELEA. **Routing and TDMA Joint Cross-Layer Design for Wireless Sensor Networks**. In *International Conference on Ad-Hoc Networks and Wireless* (2016), Springer, pp. 111–123.

[21] S. S. KULKARNI, A. IYER AND C. ROSENBERG. **An address-light, integrated MAC and routing protocol for wireless sensor networks**. *IEEE/ACM Trans. Netw. 14*, 4 (2006), 793–806.

[22] D. CHEN, J. DENG AND P. K. VARSHNEY. **A state-free data delivery protocol for multihop wireless sensor networks**. In *WCNC* (2005), pp. 1818–1823.

[23] POE, W. Y., AND SCHMITT, J. B. **Self-organized sink placement in large-scale wireless sensor networks**. In *IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)* (2009), IEEE, pp. 1–3.

[24] JAIN, T. K., SAINI, D. S., AND BHOOSHAN, S. V. **Lifetime optimization of a multiple sink wireless sensor network through energy balancing**. *Journal of Sensors* (2015).

[25] HIROMORI, A., UCHIYAMA, A., YAMAGUCHI, H., AND HIGASHINO, T. **Deadline-aware data collection in CSMA/CA-based multi-sink wireless sensor networks**. In *6th International Conference on Mobile Computing and Ubiquitous Networking (ICMU)* (2012), vol. 12, pp. 284–294.

[26] HUANG, Z., CHENG, Y., AND LIU, W. **A novel energy-efficient routing algorithm in multi-sink wireless sensor networks**. In *IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (2011), IEEE, pp. 1646–1651.

[27] CHEN, S., COOLBETH, M., DINH, H., KIM, Y.-A., AND WANG, B. **Data collection with multiple sinks in wireless sensor networks**. In *International Conference on Wireless Algorithms, Systems, and Applications* (2009), Springer, pp. 284–294.

[28] CHENG, S.-T., AND CHANG, T.-Y. **An adaptive learning scheme for load balancing with zone partition in multi-sink wireless sensor network**. *Expert Systems with Applications 39*, 10 (2012), 9427–9434.

[29] ERMAN, A. T., MUTTER, T., VAN HOESEL, L., AND HAVINGA, P. **A cross-layered communication protocol for load balancing in large scale multi-sink wireless sensor networks**. In *International Symposium on Autonomous Decentralized Systems (ISADS)* (2009), IEEE, pp. 1–8.

[30] HAMID, A., EHSAN, S., AND HAMDAOUI, B. **Rate-constrained data aggregation in power-limited multi-sink wireless sensor networks**. In *International Wireless Communications and Mobile Computing Conference (IWCMC)* (2014), IEEE, pp. 500–504.

[31] MURUGAN, K., AND PATHAN, A.-S. K. **Prolonging the lifetime of wireless sensor networks using secondary sink nodes**. *Telecommunication Systems 62*, 2 (2016), 347–361.

[32] SAFA, H., MOUSSA, M., AND ARTAIL, H. **An energy efficient genetic algorithm based approach for sensor-to-sink binding in multi-sink wireless sensor networks**. *Wireless networks 20*, 2 (2014), 177–196.

[33] SHAH-MANSOURI, V., MOHSENIAN-RAD, A.-H., AND WONG, V. W. **Lexicographically optimal routing for wireless sensor networks with multiple sinks**. *IEEE Transactions on Vehicular Technology 58*, 3 (2009), 1490–1500.

[34] SIA, Y. K., GOH, H. G., LIEW, S.-Y., AND GAN, M.-L. **Spanning multi-tree algorithm for node and traffic balancing in multi-sink wireless sensor networks**. In *12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)* (2015), IEEE, pp. 2190–2195.

[35] ODDI, G., PIETRABISSA, A., LIBERATI, F., DI GIORGIO, A., GAMBUTI, R., LANNA, A., SURACI, V., AND DELLI PRISCOLI, F. **An any-sink energy-efficient routing protocol in multi-hop wireless sensor networks for planetary exploration**. *International Journal of Communication Systems* (2015), 1–25.

[36] ZHOU, H., QING, D., ZHANG, X., YUAN, H., AND XU, C. **A multiple-dimensional tree routing protocol for multisink wireless sensor networks based on ant colony optimization**. *International Journal of Distributed Sensor Networks* (2012), 1–10.

[37] BOLER, C., AND YENDURI, S. **Resilient multi sink networks using simplistic hop based routing**. In *11th International Conference on Information Technology: New Generations (ITNG)* (2014), IEEE, pp. 192–195.

[38] MENG, M., WU, X., JEONG, B.-S., LEE, S., AND LEE, Y.-K. **Energy efficient routing in multiple sink sensor networks**. In *International Conference on Computational Science and its Applications (ICCSA)* (2007), IEEE, pp. 561–566.

[39] EGHBALI, A. N., JAVAN, N. T., DARESHOORZADEH, A., AND DEHGHAN, M. **An energy efficient load-balanced multi-sink routing protocol for wireless sensor networks**. In *10th International Conference on Telecommunications (ConTEL)* (2009), IEEE, pp. 229–234.

[40] AWANG, A. **Multi-sink routing using path loss in multihop wireless sensor networks**. In *17th Asia-Pacific Conference on Communications (APCC)* (2011), IEEE, pp. 139–144.

[41] WANG, C., AND WU, W. **A load-balance routing algorithm for multi-sink wireless sensor networks**. In *International Conference on Communication Software and Networks (ICCSN)* (2009), IEEE, pp. 380–384.

[42] JIANG, H., AND SUN, R. **Energy optimized routing algorithm in multi-sink wireless sensor networks**. *Appl. Math 8*, 1L (2014), 349–354.

[43] CARELS, D., DERDAELE, N., DE POORTER, E., VANDENBERGHE, W., MOERMAN, I., AND DEMEESTER, P. **Support of multiple sinks via a virtual root for the RPL routing protocol**. *EURASIP Journal on Wireless Communications and Networking* (2014), 1–23.

[44] FAROOQ, M. O., SREENAN, C. J., BROWN, K. N., AND KUNZ, T. **RPL-based routing protocols for multi-sink wireless sensor networks**. In *IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (2015), IEEE, pp. 452–459.

[45] HUNG, M. C.-C., LIN, K. C.-J., CHOU, C.-F., AND HSU, C.-C. **EFFORT: energy-efficient opportunistic routing technology in wireless sensor networks**. *Wireless communications and mobile computing 13*, 8 (2011), 760–773.

[46] GAO, D., LIU, Y., ZHANG, F., AND SONG, J. **Anycast routing protocol for forest monitoring in rechargeable wireless sensor networks**. *International Journal of Distributed Sensor Networks* (2013).

[47] LIU, H., ZHANG, Z.-L., SRIVASTAVA, J., AND FIROIU, V. **Pwave: A multi-source multi-sink anycast routing framework for wireless sensor networks**. In *International Conference on Research in Networking* (2007), Springer, pp. 179–190.

[48] TAN, D. D., AND KIM, D.-S. **Dynamic traffic-aware routing algorithm for multi-sink wireless sensor networks**. *Wireless Networks 20*, 6 (2014), 1239–1250.

[49] PAONE, M., PALADINA, L., SCARPA, M., AND PULIAFITO, A. **A multi-sink swarm-based routing protocol for wireless sensor networks**. In *IEEE Symposium on Computers and Communications (ISCC)* (2009), IEEE, pp. 28–33.

[50] RAZZAQUE, M. A., AND HONG, C. S. **Load and energy balanced geographic routing for sensor networks**. In *10th International Conference on Advanced Communication Technology (ICACT)* (2008), vol. 2, IEEE, pp. 1419–1422.

[51] MITTON, N., SIMPLOT-RYL, D., AND STOJMENOVIC, I. **Guaranteed delivery for geographical anycasting in wireless multi-sink sensor and sensor-actor networks.** In *28th Annual IEEE Conf. on Computer Communications (INFOCOM)* (2009), pp. 2691–2695.

[52] OZEN, S., AND OKTUG, S. **Adaptive sink selection for WSNs using forwarder set based dynamic duty cycling**. In *11th Annual IEEE International Conference on Sensing, Communication, and Networking Workshops (SECON Workshops)* (2014), IEEE, pp. 7–12.

[53] WANG, Y., AND TAN, H. **Distributed probabilistic routing for sensor network lifetime optimization**. *Wireless Networks 22*, 3 (2016), 975–989.

[54] CHAKRABORTY, S., CHAKRABORTY, S., NANDI, S., AND KARMAKAR, S. **Fault resilience in sensor networks: distributed node-disjoint multi-path multi-sink forwarding**. *Journal of Network and Computer Applications 57* (2015), 85–101.

[55] GAO, D., LIN, H., AND LIU, X. **Routing protocol for k-anycast communication in rechargeable wireless sensor networks**. *Computer Standards & Interfaces 43* (2016), 12–20.

[56] MITTON, N., SIMPLOT-RYL, D., VOGE, M.-E., AND ZHANG, L. **Energy efficient k-anycast routing in multi-sink wireless networks with guaranteed delivery**. In *International Conference on Ad-Hoc Networks and Wireless* (2012), Springer, pp. 385–398.

[57] CICIRIELLO, P., MOTTOLA, L., AND PICCO, G. P. **Efficient routing from multiple sources to multiple sinks in wireless sensor networks**. In *European Conference on Wireless Sensor Networks* (2007), Springer, pp. 34–50.

[58] HE, X., KAMEI, S., AND FUJITA, S. **Autonomous multi-source multi-sink routing in wireless sensor networks**. *Information and Media Technologies 7*, 1 (2012), 488–495.

[59] HEINZELMAN, W. R., CHANDRAKASAN, A., AND BALAKRISHNAN, H. **Energy-efficient communication protocol for wireless microsensor networks**. In *33rd Annual Hawaii International Conference on System Sciences (HICSS)* (2000), IEEE, pp. 1–10.

[60] DORIGO, M., AND DI CARO, G. **Ant colony optimization: a new meta-heuristic**. In *Congress on Evolutionary Computation (CEC)* (1999), vol. 2, IEEE, pp. 1470–1477.

[61] WINTER, T., THUBERT, P., BRANDT, A., HUI, J., KELSEY, R., LEVIS, P., PISTER, K., STRUIK, R., VASSEUR, J. P., AND ALEXANDER, R. **RPL: IPv6 routing protocol for low-power and lossy networks**. RFC 6550, https://www.rfc-editor.org/info/rfc6550, 2012. Accessed: 2017-03-20.

[62] BOUKERCHE, A., AND DAREHSHOORZADEH, A. **Opportunistic routing in wireless networks: Models, algorithms, and classifications**. *ACM Computing Surveys (CSUR) 47*, 2 (2015), 22.

[63] KABARA, J., AND CALLE, M. **MAC protocols used by wireless sensor networks and a general method of performance evaluation**. *International Journal of Distributed Sensor Networks* (2012).

[64] HUANG, P., XIAO, L., SOLTANI, S., MUTKA, M. W., AND XI, N. **The evolution of MAC protocols in wireless sensor networks: A survey**. *IEEE communications surveys & tutorials 15*, 1 (2013), 101–120.

[65] MACEDO, M., GRILO, A., AND NUNES, M. **Distributed latency-energy minimization and interference avoidance in TDMA wireless sensor networks**. *Computer Networks 53*, 5 (2009), 569–582.

[66] SOUA, R., LIVOLANT, E., AND MINET, P. **MUSIKA: A multichannel multi-sink data gathering algorithm in wireless sensor networks**. In *9th International Wireless Communications and Mobile Computing Conference (IWCMC)* (2013), IEEE, pp. 1370–1375.

[67] LEE, S. H., AND CHOI, L. **SPEED-MAC: speedy and energy efficient data delivery MAC protocol for real-time sensor network applications**. *Wireless Networks 21*, 3 (2015), 883–898.

[68] COLESANTI, U. M., SANTINI, S., AND VITALETTI, A. **DISSense: An adaptive ultralow-power communication protocol for wireless sensor networks**. In *International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)* (2011), IEEE, pp. 1–10.

[69] POLASTRE, J., HILL, J., AND CULLER, D. **Versatile low power media access for wireless sensor networks**. In *2nd International Conference on Embedded Networked Sensor Systems* (2004), ACM, pp. 95–107.

[70] OYMAN, E. I., AND ERSOY, C. **Multiple sink network design problem in large scale wireless sensor networks**. In *IEEE International Conference on Communications* (2004), vol. 6, IEEE, pp. 3663–3667.

[71] KHAN, M. M., LODHI, M. A., REHMAN, A., KHAN, A., AND HUSSAIN, F. B. **Sink-to-sink coordination framework using RPL: Routing protocol for low power and lossy networks**. *Journal of Sensors*, 1 (2016), 1–11.

[72] XU, X., AND LIANG, W. **Placing optimal number of sinks in sensor networks for network lifetime maximization**. In *IEEE International Conference on Communications (ICC)* (2011), IEEE, pp. 1–6.

[73] POE, W. Y., AND SCHMITT, J. B. **Placing multiple sinks in time-sensitive wireless sensor networks using a genetic algorithm**. In *14th GI/ITG Conference Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB)* (2008), VDE, pp. 1–15.

[74] DAI, S., TANG, C., QIAO, S., XU, K., LI, H., AND ZHU, J. **Optimal multiple sink nodes deployment in wireless sensor networks based on gene expression programming**. In *2nd International Conference on Communication Software and Networks (ICCSN)* (2010), IEEE, pp. 355–359.

[75] FLATHAGEN, J., KURE, Ø., AND ENGELSTAD, P. E. **Constrained-based multiple sink placement for wireless sensor networks**. In *IEEE 8th International Conference on Mobile Adhoc and Sensor Systems (MASS)* (2011), IEEE, pp. 783–788.

[76] SAFA, H., EL-HAJJ, W., AND ZOUBIAN, H. **Particle swarm optimization based approach to solve the multiple sink placement problem in WSNs**. In *IEEE International Conference on Communications (ICC)* (2012), IEEE, pp. 5445–5450.

[77] KIM, H., SEOK, Y., CHOI, N., CHOI, Y., AND KWON, T. **Optimal multi-sink positioning and energy-efficient routing in wireless sensor networks**. In *International Conference on Information Networking* (2005), Springer, pp. 264–274.

[78] BLAGOJEVIC, M., GEILEN, M., BASTEN, T., AND HENDRIKS, T. **Fast sink placement for gossip-based wireless sensor networks**. In *IEEE 31st International Performance Computing and Communications Conference (IPCCC)* (2012), IEEE, pp. 110–119.

[79] BOULIS, A., AND OTHERS. **Castalia: A simulator for wireless sensor networks and body area networks**. *NICTA: National ICT Australia* (2009).

[80] MATHWORKS. **The language of technical computing**. https://www.mathworks.com/products/matlab.html. Accessed: 2017-05-10.

[81] MOSEK APS. **MOSEK**. https://www.mosek.com/. Accessed: 2017-05-10.

[82] GNU OCTAVE. **GNU Octave scientific programming language**. https://www.gnu.org/software/octave/. Accessed: 2017-05-10.

[83] OSTERLIND, F., DUNKELS, A., ERIKSSON, J., FINNE, N., AND VOIGT, T. **Cross-level sensor network simulation with COOJA**. In *Proceedings of 31st IEEE Conference on Local Computer Networks* (2006), IEEE, pp. 641–648.

[84] TITZER, B. L., LEE, D. K., AND PALSBERG, J. **Avrora: Scalable sensor network simulation with precise timing**. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)* (2005), IEEE, pp. 477–482.

[85] ERIKSSON, J., ÖSTERLIND, F., FINNE, N., TSIFTES, N., DUNKELS, A., VOIGT, T., SAUTER, R., AND MARRÓN, P. J. **COOJA/MSPSim: interoperability testing for wireless sensor networks**. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques (ICST)* (2009), Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, pp. 1–27.

[86] NS-2. **Network simulator 2**. http://nsnam.sourceforge.net/. Accessed: 2017-05-10.

[87] NS-3. **Network simulator 3**. https://www.nsnam.org/. Accessed: 2017-05-10.

[88] OMNET. **OMNeT++ simulation manual**. http://www.omnetpp.org/. Accessed: 2017-05-10.

[89] RIVERBED TECHNOLOGY. **OPNET**. http://www.riverbed.com/. Accessed: 2017-05-10.

[90] HAMMOODI, I., STEWART, B., KOCIAN, A., AND MCMEEKIN, S. **A comprehensive performance study of OPNET modeler for ZigBee wireless sensor networks**. In *Third International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST)* (2009), IEEE, pp. 357–362.

[91] SCALABLE NETWORK TECHNOLOGIES. **QUALNET**. http://web.scalable-networks.com/qualnet-network-simulator-software. Accessed: 2017-05-10.

[92] TINYOS ALLIANCE. **TOSSIM**. http://docs.tinyos.net/index.php/TOSSIM. Accessed: 2017-05-10.

[93] FRABOULET, A., CHELIUS, G., AND FLEURY, E. **Worldsens: development and prototyping tools for application specific wireless sensors networks**. In *6th International Symposium on Information Processing in Sensor Networks (IPSN)* (2007), IEEE, pp. 176–185.

[94] WANG, C., AND WU, W. **A light-weighted operating system with deadlock prevention strategy for wireless sensor nodes**. In *WRI International Conference on Communications and Mobile Computing (CMC)* (2009), vol. 1, IEEE, pp. 578–583.

[95] CROSSBOW. **MICA2 datasheet**. https://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf. Accessed: 2017-05-10.

[96] MEMSIC. **MICAz datasheet**. http://www.memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf. Accessed: 2017-05-10.

[97] MEMSIC. **TelosB datasheet**. http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf. Accessed: 2017-05-10.

[98] CORPORATION, M. **TmoteSky datasheet**. http://www.eecs.harvard.edu/$\sim$konrad/projects/shimmer/references/tmote-sky-datasheet.pdf. Accessed: 2017-05-10.

[99] TEXAS INSTRUMENT. **eZ430-RF2500T datasheet**. http://www.ti.com/tool/ez430-rf2500t. Accessed: 2017-05-10.

[100] ZIOUVA, E., AND ANTONAKOPOULOS, T. **Csma/ca performance under high traffic conditions: throughput and delay analysis**. *Computer communications 25*, 3 (2002), 313–321.

[101] ROYER, E. M., AND PERKINS, C. E. **Ad-hoc on-demand distance vector routing**. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications* (1999), vol. 2, pp. 90–100.

[102] KAFI, M. A., DJENOURI, D., BEN-OTHMAN, J., AND BADACHE, N. **Congestion control protocols in wireless sensor networks: a survey**. *IEEE Communications Surveys & Tutorials 16*, 3 (2014), 1369–1390.

[103] CHEN, D., AND VARSHNEY, P. K. **A survey of void handling techniques for geographic routing in wireless networks**. *IEEE Communications Surveys & Tutorials 9*, 1 (2007), 50–67.

[104] ELHAFSI, E. H., MITTON, N., AND SIMPLOT-RYL, D. **End-to-end energy efficient geographic path discovery with guaranteed delivery in ad hoc and sensor networks**. In *PIMRC 2008* (2008), p. 000.

[105] HEISSENBÜTTEL, M., BRAUN, T., BERNOULLI, T., AND WÄLCHLI, M. **BLR: beacon-less routing algorithm for mobile ad hoc networks**. *Computer communications 27*, 11 (2004), 1076–1086.

[106] BUETTNER, M., YEE, G. V., ANDERSON, E., AND HAN, R. **X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks**. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems* (2006), ACM, pp. 307–320.

[107] PAPADOPOULOS, G. Z., KRITSIS, K., GALLAIS, A., CHATZIMISIOS, P., AND NOEL, T. **Performance evaluation methods in ad hoc and wireless sensor networks: a literature study**. *IEEE Communications Magazine 54*, 1 (2016), 122–128.

[108] LEAO, L., FELEA, V., AND GUYENNET, H. **MAC-aware routing in multi-sink WSN with dynamic back-off time and buffer constraint**. In *8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)* (2016), IEEE, pp. 1–5.

[109] LEÃO, L., AND FELEA, V. **Latency and lifetime optimization for k-anycast routing algorithm in wireless sensor networks**. In *International Conference on Ad-Hoc Networks and Wireless* (2018), Springer, pp. 39–50.

[110] LEÃO, L., AND FELEA, V. **Latency and network lifetime trade-off in geographic multicast routing for multi-sink wireless sensor networks**. In *International Conference on Mobile, Secure, and Programmable Networking* (2018), – presented, pp. 1–12.

[111] YOUNIS, M., AND AKKAYA, K. **Strategies and techniques for node placement in wireless sensor networks: A survey**. *Ad Hoc Networks 6*, 4 (2008), 621–655.

[112] DUCROCQ, T., HAUSPIE, M., MITTON, N., AND PIZZI, S. **On the impact of network topology on wireless sensor networks performances: Illustration with geographic routing**. In *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on* (2014), IEEE, pp. 719–724.

[113] FARAH, E., ABDALLAH, A., AND SHAHROUR, I. **Sunrise: large scale demonstrator of the smart water system**. *International Journal of Sustainable Development and Planning 12*, 1 (2017), 112–121.

List of Figures

List of Tables

IV

ANNEXES

A

NETWORK GENERATION

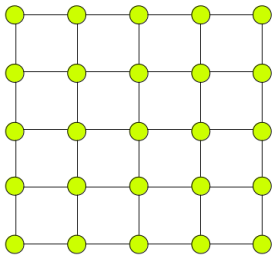
This annex presents a brief investigation of the network generation strategies that was performed with the purpose of supporting the generation of networks for our simulations.

As presented in [111], node placement has much influence in the network performance. Commonly, in simulated environments, nodes are randomly placed following specific constraints, such as radio range, number of nodes per area etc. According to [112], the random placement is generally used for convenience, since it can be easily generated. In real implementations, nodes are placed according to the application needs.

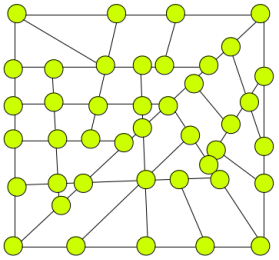
In this chapter we present a basic definition of the different network deployment strategies found in literature. We also present our random network generator, along with a comparison between a random generated network and a real deployment.

A.1/ NETWORK DEPLOYMENT

The placement of nodes may follow a grid-wise layout, a lengthwise, a geometric shape etc. It depends on the final application and what is planned to be measured. According to [111], the position of the nodes significantly impacts the network lifetime. As for instance, a random placement following a uniform distribution may lead to an early disconnection compared to a gradient distribution centered at the sink. In a uniform distribution the nodes in the sink vicinity deplete their energy in a faster way. On the other hand, a network tends to have a longer lifetime with a gradient distribution of nodes, where the highest density is in the vicinity of the sink, with a decrease of the deployment density with the distancing from the sink.



(a) Manhattan city like deployments



(b) European city deployments

Figure A.1: City deployment examples

In [112] authors demonstrate that the performance of routing strategies may vary according to the network topology. The authors compared the results of completely random deployments, Manhattan city like deployments (Figure A.1a), which is a grid-like deployment, and european city deployments (Figure A.1b), which can be seem as random-grid-like deployment, since the nodes are distributed in an organized way, but not strictly organized as in a pure grid deployment. The test results show that random and european city deployments behave differently compared to Manhattan city deployments. Because of that, authors suggest to design the routing protocol considering the deployment type required by the final application.

In terms of network deployment in real-life, we can identify three different scenarios:

- application-specific topology: nodes are placed according to the application needs;
- efficient placement: nodes are placed to achieve maximum performance on the predefined objectives (network lifetime, connectivity, latency, etc);
- random topology: nodes are randomly deployed. It normally happens when manual deployment is impracticable.

Although authors in [112] suggest that routing protocols must be optimized to the type of deployment, the generation of networks using random deployment seems to be the most appropriate option for the validation of routing protocols. The performance of the network tends to be improved with efficient deployments, thus it is important to execute the tests for the scenarios where the deployment was not improved, such as in the random topology.

A.2/ RANDOM DEPLOYMENTS FOR SIMULATION

For the random generation of network deployments, we implemented the Algorithm 14 capable of generating completely random networks and random-grid networks. We consider a random-grid deployment as the random deployment of nodes over a grid. It means that nodes are randomly placed within a grid cell. The algorithm accepts input parameters to generate networks with specific characteristics, such as density of the deployment, area size, communication range, number of sensor nodes and number of sinks. Along with the parameters, the algorithm also allows the activation of the defined rules: *minimal distance rule* and *neighbor rule*. The *minimal distance rule* tells the algorithm that the selected place must be at a minimal distance to all already placed nodes. The objective with this rule is to better distribute the nodes over the area and avoid a concentration of nodes. The *neighbor rule* is used to make sure that for each placed node, at least one neighbor exists. The objective of this rule is to reduce the risk of disconnected networks, with isolated nodes.

The variable e_{total} represents one edge of the deployment area. If the generation considers a grid for the deployment, the variable e_{min} represents the edge of one cell of the grid. The algorithm executes the placement strategy until all nodes are placed (line 5). The algorithm randomly selects a position within a cell area (lines 8 and 9). The selected position is analyzed against the rules only if they are activated (line 12). The algorithm checks if the position is at a minimum distance from all other placed nodes and if it is in the radio range of at least one node. Finally, the position is included as a valid node placement (line 26) and the algorithm moves to another cell area in the grid (line 31).

We consider the network density as the average number of nodes in an area equivalent to the communication range ($\frac{\pi \times range^2}{A} \times |WSN|$). All the deployments were created using our network generator developed for SciLab [5], with Mersenne Twister algorithm as the random numbers generator.

Algorithm 14 $WSN = netGen(sinks, sensors, range, minDist, grid, A, neighborR, minDistR)$

Input: $sinks$: number of sinks, $sensors$: number of sensors, $range$: radio range, $minDist$: minimal distance between nodes, $grid$: number of cells (must be a square number), A: deployment area, $neighborR$: if the neighbor rule must be applied, $minDistR$: if the minimal distance rule must be applied

Output: WSN: the generated network

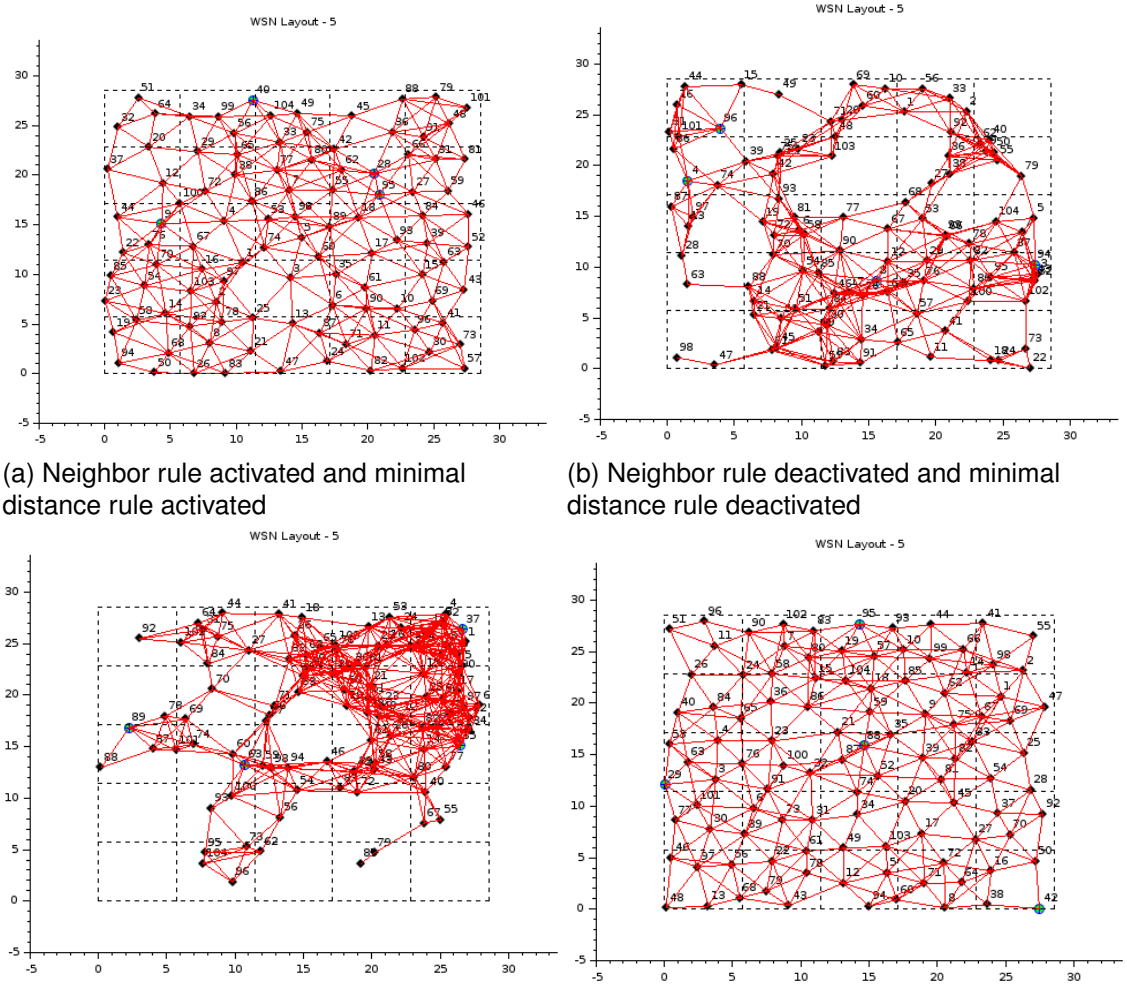
$WSN \leftarrow \emptyset$ /* WSN holds the set of sensors and sinks positions: $\{(x_1, y_1)(x_2, y_2)...(x_n, y_n)\}$ */
$e_{total} \leftarrow sqrt(A)$
$e_{min} \leftarrow e_{total} / sqrt(grid)$
$y_{axis} \leftarrow e_{min}$; $x_{axis} \leftarrow e_{min}$
while $size(WSN) < (sinks + sensors)$ **do**
  repeat
    $nodeCreated \leftarrow false$
    $x \leftarrow random((x_{axis} - e_{min}), x_{axis})$
    $y \leftarrow random((y_{axis} - e_{min}), y_{axis})$
    $pos \leftarrow (x, y)$
    if $WSN \cap \{pos\} = \emptyset$ **then**
      if $minDistR$ **or** $neighborR$ **then**
        $minDistT \leftarrow true$
        $neighborT \leftarrow false$
        for $n \in WSN$ **do**
          if $distance(n, pos) \leq minDist$ **then**
            $minDistT \leftarrow false$
            break
          end if
          if $distance(n, pos) \leq range$ **then**
            $neighborT \leftarrow true$
          end if
        end for
      end if
      if $WSN = \emptyset$ **or** ((**not** $minDistR$ **or** $minDistT$) **and** (**not** $neighborR$ **or** $neighborT$)) **then**
        $WSN \leftarrow WSN \cup \{pos\}$
        $nodeCreated \leftarrow true$
      end if
    end if
  until not $nodeCreated$
  if $y_{axis} \leq e_{total}$ **and** $grid > 1$ **then**
    if $x_{axis} \leq e_{total}$ **then**
      $x_{axis} \leftarrow x_{axis} + e_{min}$
    else
      $x_{axis} \leftarrow e_{min}$
      $y_{axis} \leftarrow y_{axis} + e_{min}$
    end if
  else
    $y_{axis} \leftarrow e_{min}$
    $x_{axis} \leftarrow e_{min}$
  end if
end while
return WSN

A.2.1/ COMPLETELY RANDOM DEPLOYMENTS

The networks were generated considering the same input in terms of network size, density and network area (Tables A.1 and A.3), but varying the activation/deactivation of the generation rules (*neighbor rule* and *minimal distance rule*). Figure A.2 considers networks generated with a density of 10, and Figure A.3 considers a density of 4.

Table A.1: Completely random network details

Network size	Sinks	Sensors	Network area	Density	Communication range
104	4	100	816 m^2	10	5 m



(a) Neighbor rule activated and minimal distance rule activated

(b) Neighbor rule deactivated and minimal distance rule deactivated

(c) Neighbor rule activated and minimal distance rule deactivated

(d) Neighbor rule deactivated and minimal distance rule activated

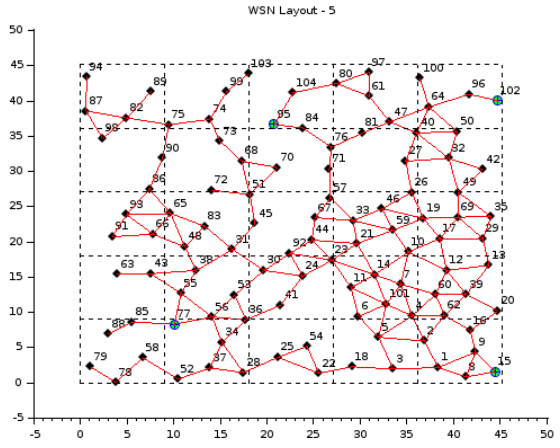
Figure A.2: Completely random network deployment for a density of 10

Table A.2: Completely random network comparison

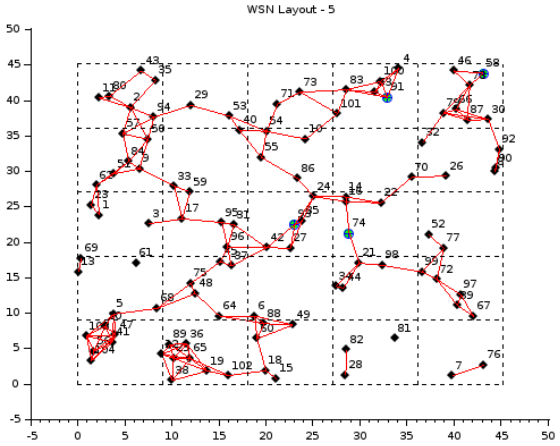
	Figure A.2a	Figure A.2b	Figure A.2c	Figure A.2d
Average number of neighbors	7.60	8.13	12.44	7.25
Standard deviation	1.98	2.92	5.68	1.91
Median	8.00	8.00	13.00	7.50

Table A.3: Completely random network details

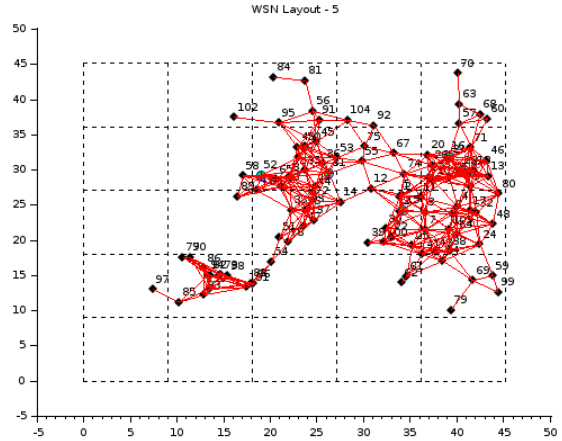
Network size	Sinks	Sensors	Network area	Density	Communication range
104	4	100	2042 m^2	4	5 m



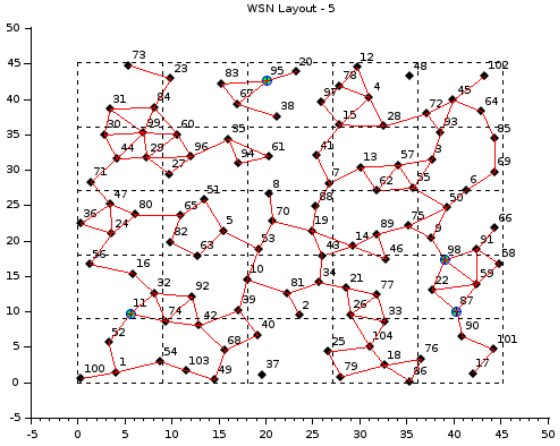
(a) Neighbor rule activated and minimal distance rule activated



(b) Neighbor rule deactivated and minimal distance rule deactivated



(c) Neighbor rule activated and minimal distance rule deactivated



(d) Neighbor rule deactivated and minimal distance rule activated

Figure A.3: Completely random network deployment for a density of 4

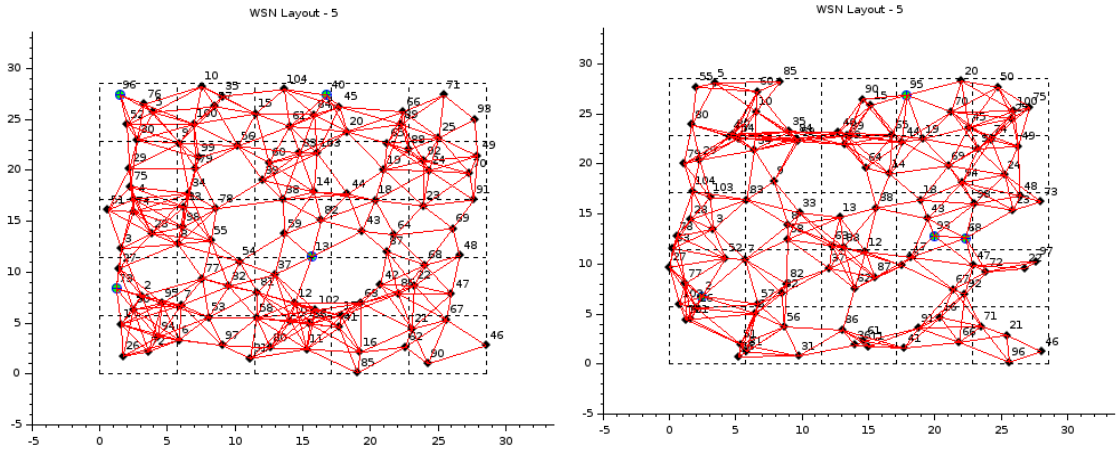
Table A.4: Completely random network comparison

	Figure A.3a	Figure A.3b	Figure A.3c	Figure A.3d
Average number of neighbors	3.27	3.21	8.96	2.73
Standard deviation	1.33	1.46	4.00	1.07
Median	3.00	3.00	9.00	3.00

Tables A.2 and A.4 show the average number of neighbor nodes, the standard deviation and the median. If we observe only the average number of neighbors, the networks generated with all rules deactivated present the closest value to the desired density. However, most of the generated networks are not connected (the network connectivity is verified by checking if from each node of the network it is possible to reach all other nodes by at least one path). Moreover, the standard deviation is more elevated, which means that some of the nodes have more neighbors than the desired density, creating many void areas. The effect of the minimal distance rule is noticeable in Figure A.2. When this rule is activated, the node distribution over the area is improved, we can confirm that with the smaller standard deviation. However, the neighbor rule does not seem to have a big impact on the final network. It is because of the network density, that is already elevated. The effect of the neighbor rule is much more perceived in Figure A.3. We can notice that when the neighbor rule is deactivated, the network is not connected, presenting some isolated nodes.

A.2.2/ RANDOM-GRID DEPLOYMENTS

Once again, the networks were generated considering the same inputs presented in Tables A.5 and A.7. However, this time the grid size was set to 25 cells, in order to generated random-grid networks. Figures A.4 and A.5 present the results of the generated networks. We present and evaluate only the results of the networks with either both rules activated or deactivated, since those are the configurations that offers better outputs in terms of coverage and connectivity.



(a) Neighbor rule activated and minimal distance rule activated

(b) Neighbor rule deactivated and minimal distance rule deactivated

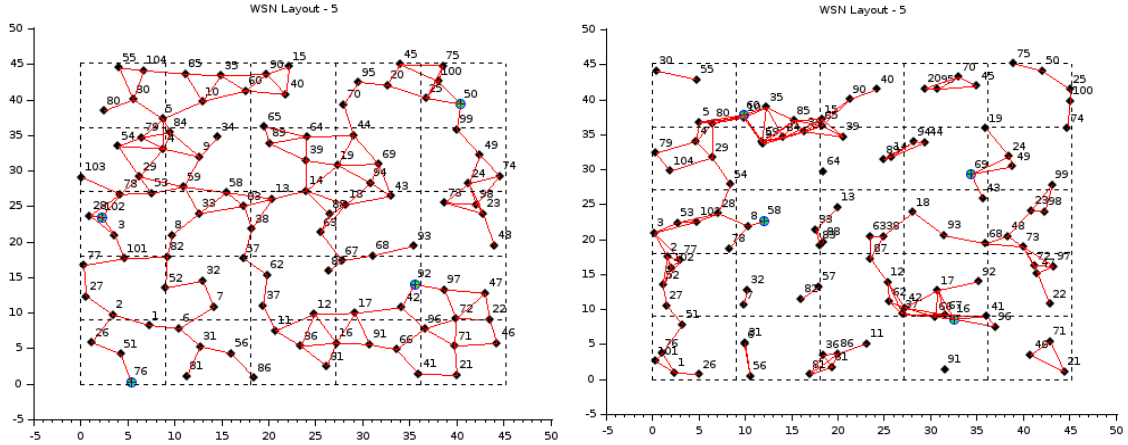
Figure A.4: Random-grid network deployment for a density of 10

Table A.5: Random-grid network details

Network size	Sinks	Sensors	Network area	Density	Grid size	Communication range
104	4	100	816 m^2	10	25	5 m

Table A.6: Random-grid network comparison

	Figure A.4a	Figure A.4b
Average number of neighbors	7.96	7.50
Standard deviation	2.18	2.06
Median	8.00	7.00



(a) Neighbor rule activated and minimal distance rule activated

(b) Neighbor rule deactivated and minimal distance rule deactivated

Figure A.5: Random-grid network deployment for a density of 4

Table A.7: Random-grid network details

Network size	Sinks	Sensors	Network area	Density	Grid size	Communication range
104	4	100	816 m^2	4	25	5 m

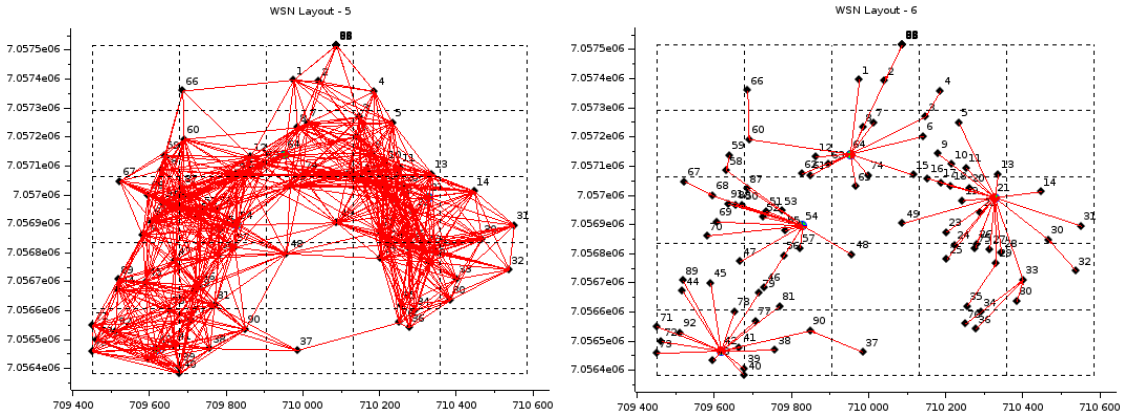
Table A.8: Random-grid network comparison

	Figure A.5a	Figure A.5b
Average number of neighbors	3.06	2.92
Standard deviation	1.15	1.66
Median	3.00	3.00

Since the random-grid deployment considers placing nodes evenly among the cells, the minimal distance rule does not have a large impact on the final deployment. However, the neighbor rule remains an important strategy to assure the connectivity and avoid isolated nodes, especially for the low density network. We can notice in Tables A.6 and A.8 that the average number of neighbor nodes is better when the neighbor rule is activated.

A.3/ COMPARISON: REAL-LIFE AND SIMULATED DEPLOYMENT

In order to be able to test our solutions with a network deployment reflecting characteristics of a real network deployment, we decided to analyze a real deployment against the networks generated by our tool. Figure A.6 presents a real-life deployment based on the geographic position data extracted from the work in [113]. It is a representation of a monitoring system conceived to keep track of water flow in a water distribution network, as a way to detect water leakage and consumption. The network comprises 92 sensor nodes and 4 sinks, with all sensor nodes sending packets to one of the sinks. The packets are forwarded to the closest sink, as presented in Figure A.6b.



(a) Graph view (b) Routing tree view

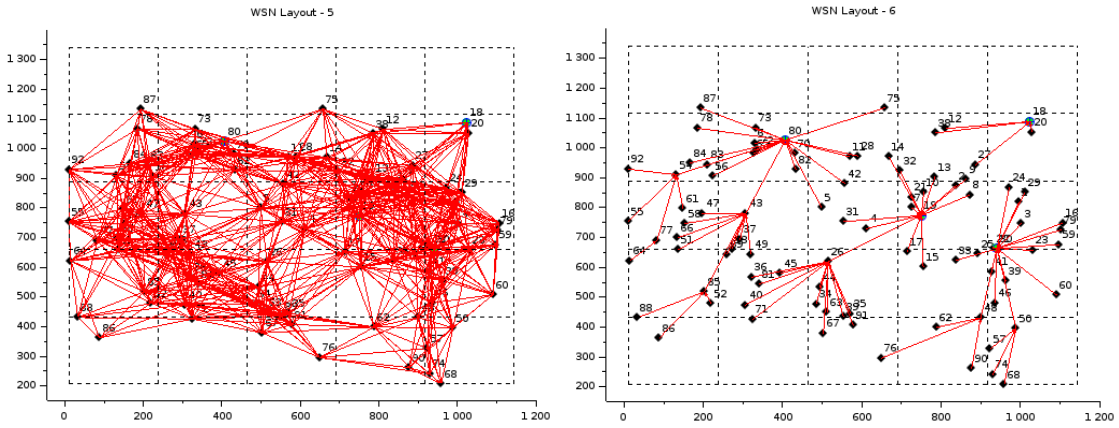
Figure A.6: Real-life deployment

We can observe in Table A.9 the network data. The real deployment presents a node density of 20.19 and a node has an average number of neighbors of 21.07.

Table A.9: Real-life network deployment details

Network size	92
Sinks	4
Sensors	88
Deployment area	1288225 m^2
Communication range	300 m
Density	20.19
Average number of neighbors	21.07

The simulated deployment was generated based on the network data from the real-life deployment in Table A.9. A completely random network was generated considering a network density of 20 nodes, close to the 20.19 presented by the real deployment. The rule for the minimal distance between nodes was deactivated and the rule for at least one neighbor was activated. The generated network can be seen in Figure A.7.



(a) Graph view (b) Routing tree view

Figure A.7: Simulated deployment

Table A.10: Simulated network deployment details

Network size	92
Sinks	4
Sensors	88
Deployment area	1288225 m^2
Communication range	300 m
Density	20.19
Average number of neighbors	21.20

We can notice in Table A.10 that the network details for the network density and the average number of neighbors are quite similar to the real deployment. The physical placement is not identical, but some of the characteristics are reproduced, such as the existence of nodes at a 3-hop distance of the closest sink, and the existence of uncovered areas, as presented in Figure A.7b.

A.4/ CONCLUSION

The node deployment in wireless sensor networks is already research topic all alone. Allied with a routing protocol, the optimal placement of nodes may result in reduced latency and better efficiency in energy consumption. However, the random generation of networks is an important strategy for the validation of new routing protocols. In this chapter we presented the strategies applied for the generation of random networks with the purpose of validating our routing solutions. All the generated networks for all solutions followed the same generation method, applying the random-grid strategy with the minimal distance and neighbor rule activated, in order to assure the network connectivity and coverage.

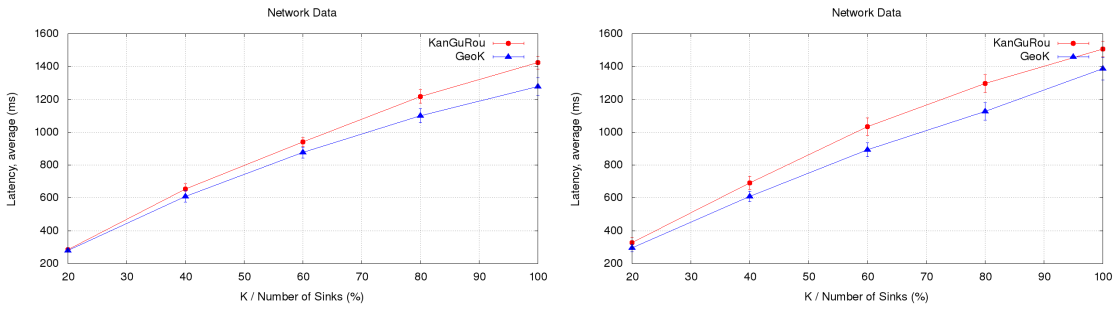
B

GeoK performance

In order to provide a more complete view of the GeoK performance in comparison to KanGuRou, we present in this annex the results concerning the latency and maximum energy consumption for all evaluated network sizes and values of k for both types of networks, without void areas and with void areas. The results already presented in Chapter 8 are not repeated.

B.1/ Latency: 50 to 250 sensor nodes

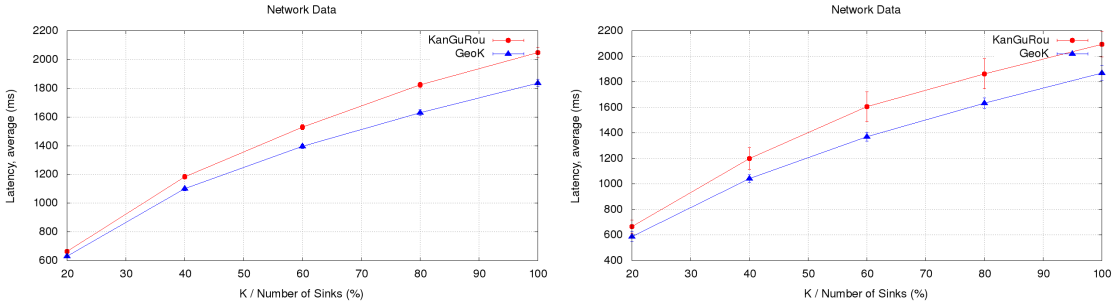
In terms of latency, GeoK has always a better performance in comparison with KanGuRou. We can notice in Figures B.1, B.2, B.3, B.4 and B.5 that the performance gain increases in favor of GeoK with the increase of the network size and of the values of k.



(a) Network **without void** areas

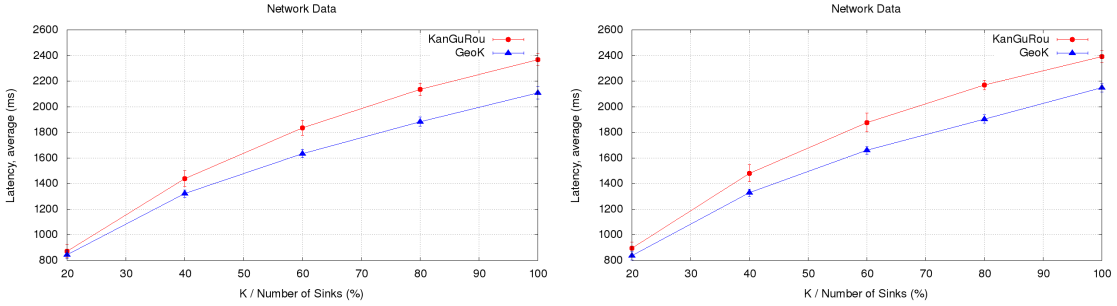
(b) Network **with void** areas

Figure B.1: Network with 50 sensor nodes, sinks at 10% of the sensors and k varying with a step of 20% of the number of sinks



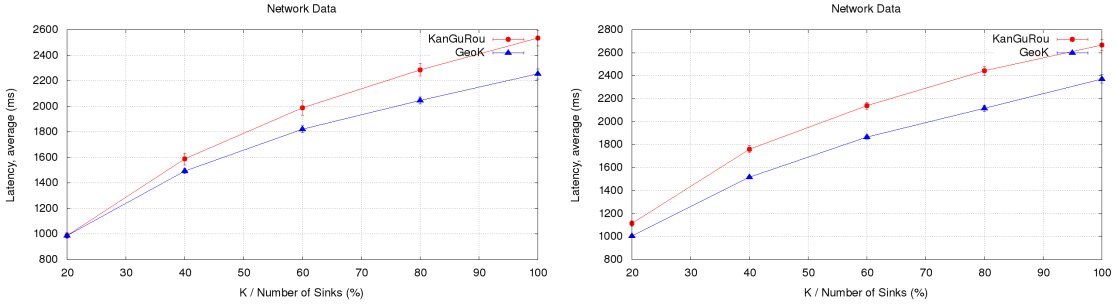
(a) Network **without void** areas (b) Network **with void** areas

Figure B.2: Network with 100 sensor nodes, sinks at 10% of the sensors and k varying with a step of 20% of the number of sinks



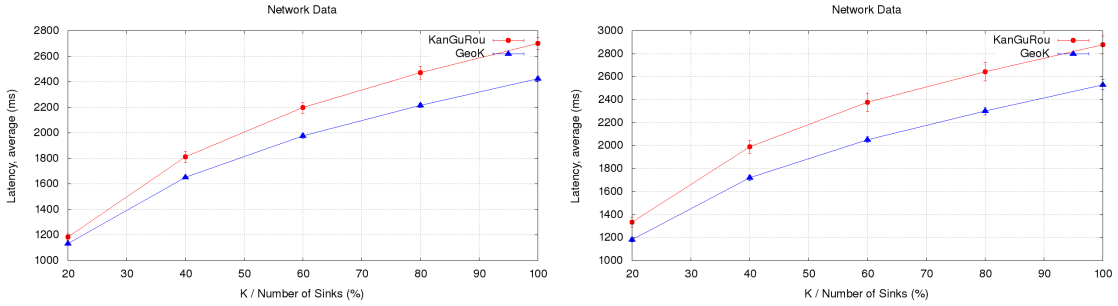
(a) Network **without void** areas (b) Network **with void** areas

Figure B.3: Network with 150 sensor nodes, sinks at 10% of the sensors and k varying with a step of 20% of the number of sinks



(a) Network **without void** areas (b) Network **with void** areas

Figure B.4: Network with 200 sensor nodes, sinks at 10% of the sensors and k varying with a step of 20% of the number of sinks

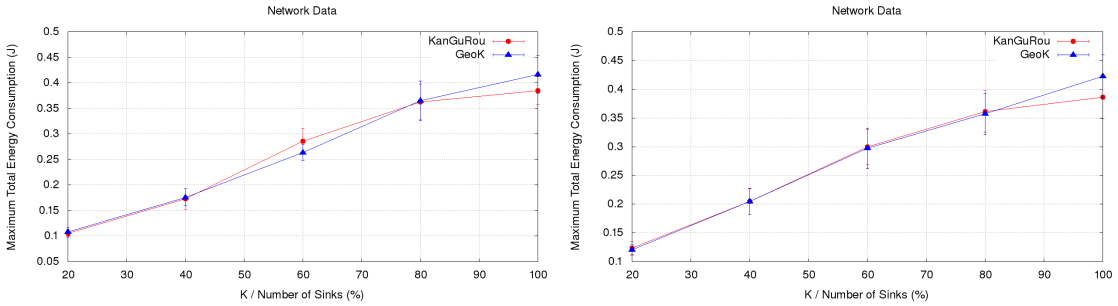


(a) Network **without void** areas (b) Network **with void** areas

Figure B.5: Network with 250 sensor nodes, sinks at 10% of the sensors and k varying with a step of 20% of the number of sinks

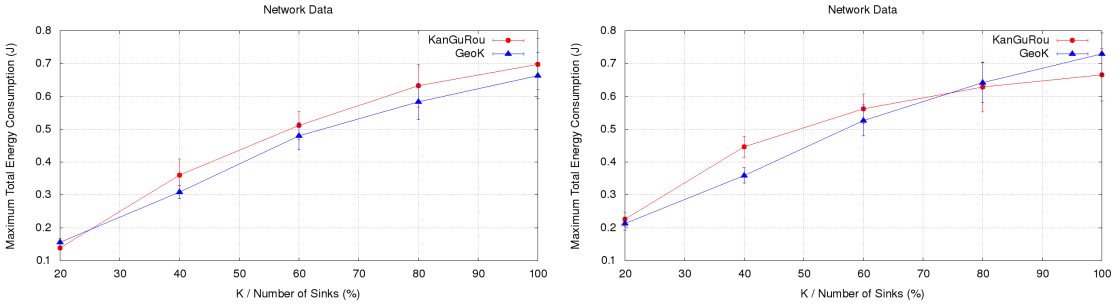
B.2/ MAXIMUM ENERGY CONSUMPTION: 50 TO 250 SENSOR NODES

In terms of maximum energy consumption, in Figures B.6, B.7, B.8, B.9 and B.10 we can see that GeoK has a better performance for larger networks with smaller values of k. It is explained by the fact that GeoK duplicates the packets earlier than KanGuRou in order to reach the sinks in a faster way. Consequently, the increase of k also increases the number of packets and the energy consumption.



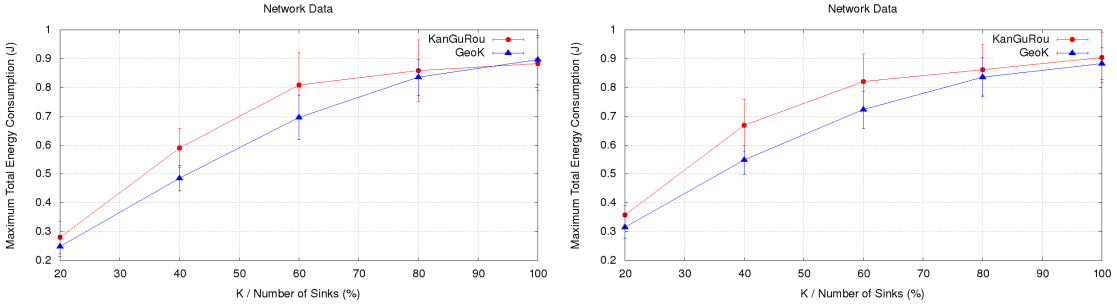
(a) Network **without void** areas (b) Network **with void** areas

Figure B.6: Network with 50 sensor nodes, sinks at 10% of the sensors and k varying with a step of 20% of the number of sinks



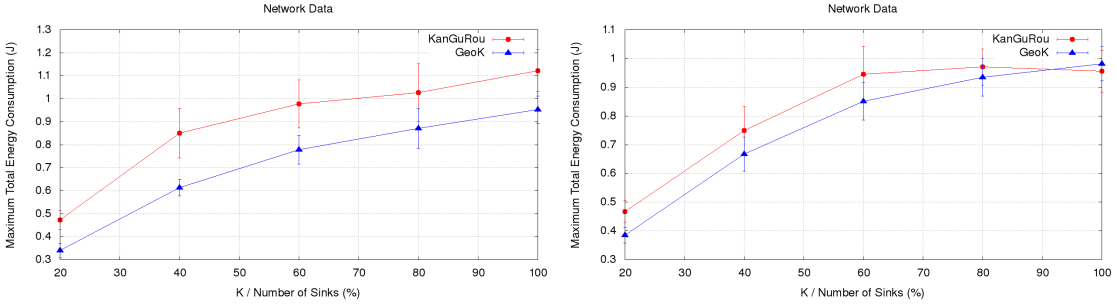
(a) Network **without void** areas (b) Network **with void** areas

Figure B.7: Network with 100 sensor nodes, sinks at 10% of the sensors and k varying with a step of 20% of the number of sinks



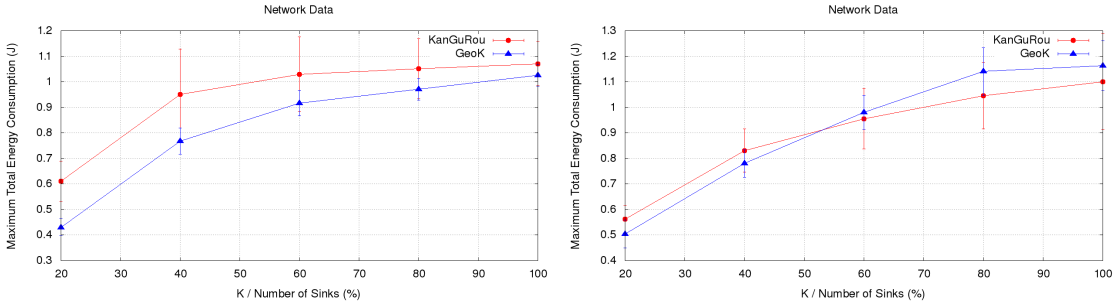
(a) Network **without void** areas (b) Network **with void** areas

Figure B.8: Network with 150 sensor nodes, sinks at 10% of the sensors and k varying with a step of 20% of the number of sinks



(a) Network **without void** areas (b) Network **with void** areas

Figure B.9: Network with 200 sensor nodes, sinks at 10% of the sensors and k varying with a step of 20% of the number of sinks



(a) Network **without void** areas

(b) Network **with void** areas

Figure B.10: Network with 250 sensor nodes, sinks at 10% of the sensors and k varying with a step of 20% of the number of sinks

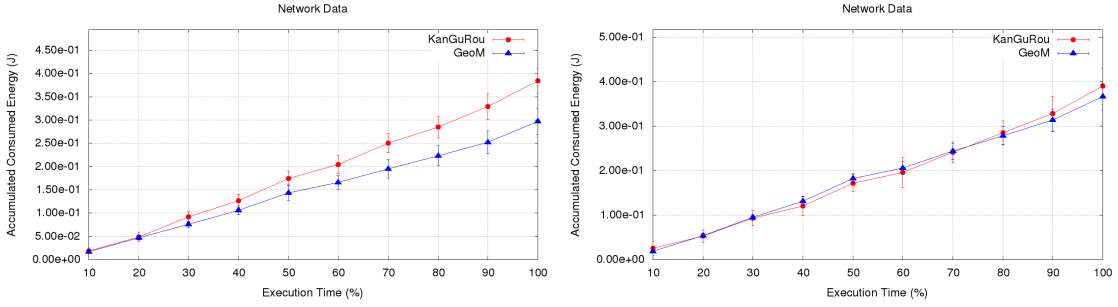
C

GeoM performance

We present in this annex the results concerning the maximum energy consumption over time and the distribution of nodes in terms of the consumed energy for all evaluated network sizes for both networks without void areas and with void areas. The results already presented in Chapter 9 are not repeated.

C.1/ Maximum energy consumption over time: 50 to 250 sensor nodes

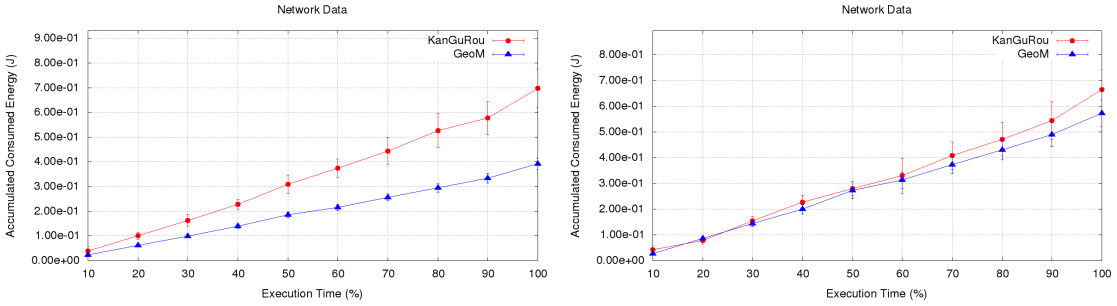
We can notice in Figures C.1, C.2, C.3, C.4 and C.5 that GeoM has a better performance in terms of maximum energy consumption, with the increase over time being much smoother than KanGuRou for networks without void areas. However, the results for networks with void areas show that GeoM has a similar behavior compared to KanGuRou. As already explained in Chapter 9, this behavior is a consequence of the recovery mode, since the path during the recovery mode is always the same.



(a) Network **without void** areas

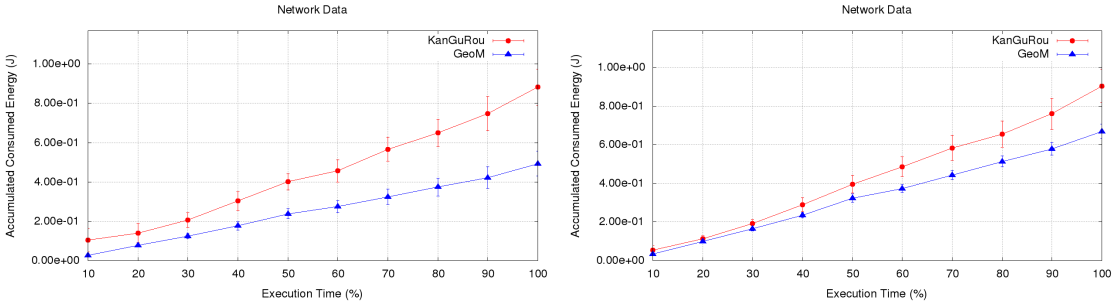
(b) Network **with void** areas

Figure C.1: Network with 50 sensor nodes, sinks at 10% of the sensor nodes



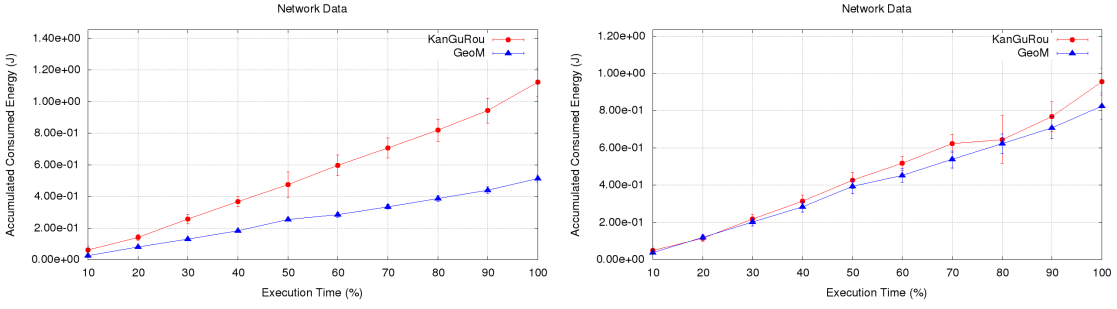
(a) Network **without void** areas (b) Network **with void** areas

Figure C.2: Network with 100 sensor nodes, sinks at 10% of the sensor nodes



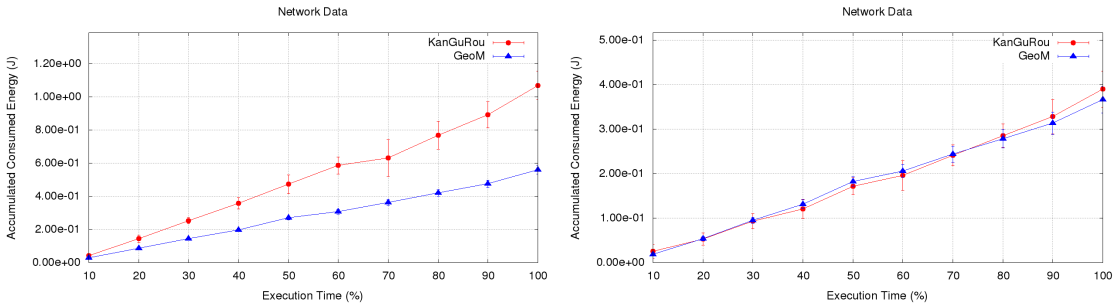
(a) Network **without void** areas (b) Network **with void** areas

Figure C.3: Network with 150 sensor nodes, sinks at 10% of the sensor nodes



(a) Network **without void** areas (b) Network **with void** areas

Figure C.4: Network with 200 sensor nodes, sinks at 10% of the sensor nodes

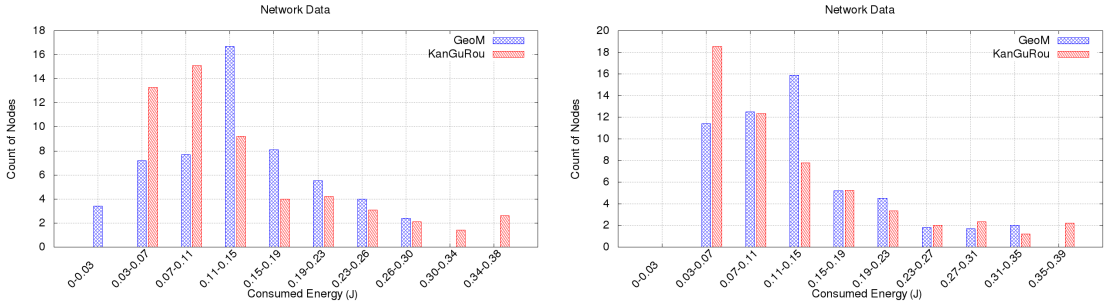


(a) Network **without void** areas (b) Network **with void** areas

Figure C.5: Network with 250 sensor nodes, sinks at 10% of the sensor nodes

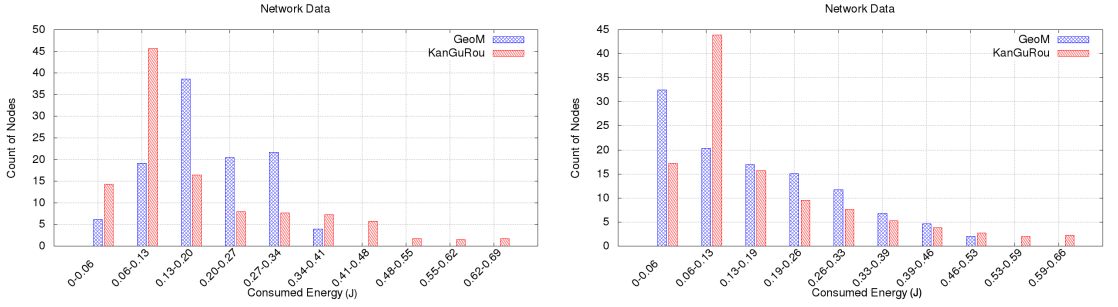
C.2/ NODES DISTRIBUTION IN TERMS OF ENERGY CONSUMPTION: 50 TO 250 SENSOR NODES

As presented in Figures C.6, C.7, C.8, C.9 and C.10, GeoM is capable of balancing the energy consumption in the neighborhood of the forwarder node, avoiding the overuse of the same nodes. We can see that even for the networks with void areas, GeoM performs better than KanGuRou. It is possible to notice that KanGuRou has nodes progressing towards higher intervals of the consumed energy in all scenarios.



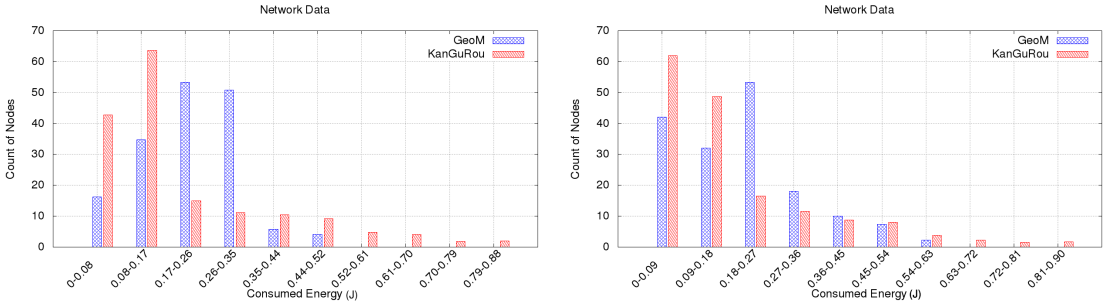
(a) Network **without void** areas (b) Network **with void** areas

Figure C.6: Network with 50 sensor nodes, sinks at 10% of the sensor nodes



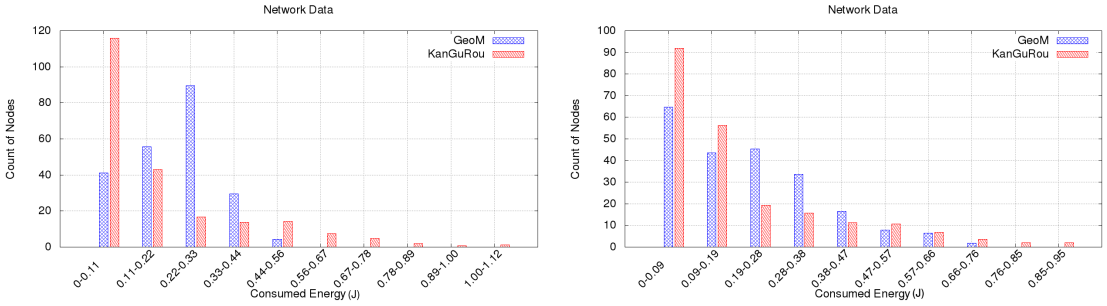
(a) Network **without void** areas (b) Network **with void** areas

Figure C.7: Network with 100 sensor nodes, sinks at 10% of the sensor nodes



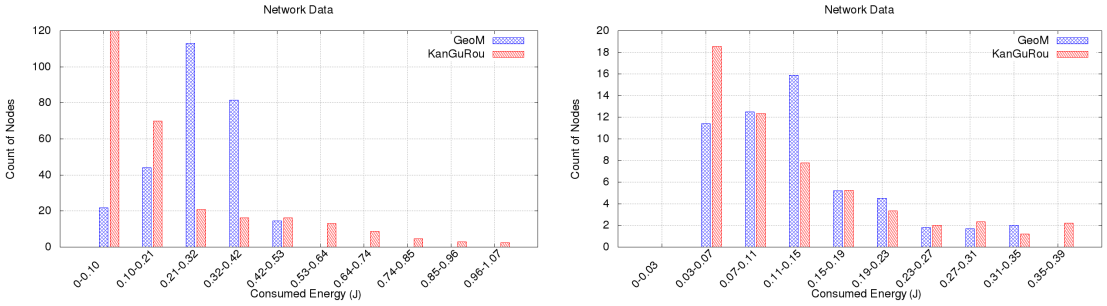
(a) Network **without void** areas (b) Network **with void** areas

Figure C.8: Network with 150 sensor nodes, sinks at 10% of the sensor nodes



(a) Network **without void** areas (b) Network **with void** areas

Figure C.9: Network with 200 sensor nodes, sinks at 10% of the sensor nodes



(a) Network **without void** areas (b) Network **with void** areas

Figure C.10: Network with 250 sensor nodes, sinks at 10% of the sensor nodes

Document generated with LaTeX and:
the LaTeX style for PhD Thesis created by S. Galland — http://www.multiagent.fr/ThesisStyle
the tex-upmethodology package suite — http://www.arakhne.org/tex-upmethodology/