UNIVERSIDADE NOVA DE LISBOA
OMNIS CIVITAS CONTRA SE DIVISA NON STABIT

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

ANDRÉ FIALHO FERREIRA ROSA

Bachelor Degree in Science and Computer Engineering

COMMUNICATION PRIMITIVES FOR WIRELESS AD HOC NETWORKS

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
November, 2021

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

COMMUNICATION PRIMITIVES FOR WIRELESS AD HOC NETWORKS

ANDRÉ FIALHO FERREIRA ROSA

Bachelor Degree in Science and Computer Engineering

Adviser: João Carlos Antunes Leitão
Assistant Professor, NOVA University of Lisbon

Examination Committee:

Chair: João Manuel dos Santos Lourenço
Associate Professor, NOVA University of Lisbon

Rapporteur: Hugo Alexandre Tavares Miranda
Associate Professor, University of Lisbon

Adviser: João Carlos Antunes Leitão
Assistant Professor, NOVA University of Lisbon

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
November, 2021

Communication Primitives for Wireless Ad Hoc Networks

Acknowledgements

Throughout the development of this thesis, I received a great deal of support that was crucial for its completion. As such, I take this opportunity to express my appreciation to everyone that made a significant contribution.

In the first place, I want to convey a special thank you to Jéssica Serrano for all the incessant and unwavering encouragement offered, as well as for being a sympathetic ear and reviewing this dissertation.

Moreover, I intend to express my heartfelt gratitude towards my family, especially my parents and my brothers, who supported my academic endeavors throughout the years. I would also like to highlight the vital role my pets had during this venture by providing daily companionship and cheerful distractions.

In addition, I would like to extend my sincere thanks to my colleagues and friends, in particular Rúben Silva and Hugo Mochão, for the esteemed time spent together full of laughs and jokes throughout my academic years. I take this opportunity to add that I am sorry for spilling your beverages, it was not on purpose.

I also wish to state a great thank you to my senior colleague Pedro Ákos for all the help and assistance provided throughout the elaboration of this thesis, especially during the experimental evaluation of my work.

Finally, I would like to convey my most profound appreciation to my advisor, Professor João Leitão, for his invaluable guidance and advice, always seasoned with a pinch of humor, which challenged me to elevate my work to a higher level. Working with him since my early academic years allowed me to deepen my knowledge in computer science far beyond what is covered in the courses and gain new insights and experience regarding scientific research.

The work presented in this thesis was partially supported by FC&T through NOVA LINCS (grant UIDB/04516/2020) and NG-STORAGE (PTDC/CCIINF/32038/2017).

"To attain knowledge, add things everyday. To attain wisdom, remove things every day." (Lao Tzu)

Abstract

In recent times, wireless *ad hoc* networks have been becoming increasingly relevant due to their suitability for Internet-of-Things (IoT) applications. These networks are comprised of a set of devices that communicate directly with each other through the wireless medium, without relying on any pre-existing infrastructure. In deployments scattered across a wide area, the devices are unable to reach all other devices directly, and thus, they must cooperate by retransmitting messages on behalf of other devices, to enable network-wide communications. In this sense, practical abstractions that distributively coordinate the communications between the network's participants —*Communication Primitives* —are fundamental pieces to devise distributed applications and services for these networks. Throughout the literature, *Neighbor Discovery*, *Broadcast*, and *Routing* emerge as key abstractions/services to support these primitives.

Although numerous instances and improvements of these three services have been proposed over the years, no single solution can be considered the most suitable in all scenarios due to these networks being highly dynamic, having a heterogeneous nature, and being affected by environmental conditions (e.g., noise in the wireless medium). Hence, it is essential to identify how the different proposed solutions relate to each other to better compare, combine, or dynamically select their distinct strategies in different settings. However, unveiling the relations between them is quite challenging due to their highly heterogeneous specifications and assumptions. This observation motivated us to devise, for each service, a generic conceptual framework that abstracts its solutions' common elements while offering parameters to capture the behavior of particular instances (i.e., proposed solutions). In addition, to address the lack of systematic experimental evaluation of these services on real networks, we conducted an experimental evaluation of some representative solutions on a wireless *ad hoc* network formed by commodity devices, leveraging prototypes of our frameworks. The results revealed new interesting insights that had not been previously studied in the context of communication primitives for these networks.

Keywords: Wireless Ad Hoc Network, Broadcast, Routing, Neighbor Discovery, Framework, IoT

Resumo

Nos últimos tempos, as redes *ad hoc* sem-fios têm vindo a tornar-se cada vez mais relevantes devido à sua aptidão para aplicações na Internet das Coisas (IoT). Estas redes são compostas por um conjunto de dispositivos que comunicam diretamente entre si através do meio sem-fios, sem recorrer a nenhuma infraestrutura pré-existente. Em implantações espalhadas ao longo de uma ampla área, os dispositivos são incapazes de alcançar diretamente todos os outros e, por conseguinte, têm de cooperar retransmitindo mensagens em nome de outros dispositivos, afim de possibilitar comunicações por toda a rede. Neste sentido, abstrações práticas que distributivamente coordenem as comunicações entre os participantes da rede — *Primitivas de Comunicação* — são peças fundamentais para conceber aplicações e serviços distribuídos para estas redes. Ao longo da literatura, *Descoberta de Vizinhos*, *Difusão* e *Encaminhamento* emergem como as abstrações/serviços chave para suportar estas primitivas.

Embora inúmeras instâncias e melhorias destes três serviços tenham sido propostas ao longo dos anos, nenhuma solução singular pode ser considerada a mais adequada em todos os cenários devido a estas redes serem extremamente dinâmicas, terem um natureza heterógena e serem afetadas por condições ambientais (e.g., ruído no meio sem-fios). Deste modo, é essencial identificar como as diferentes soluções se relacionam entre si para melhor comparar, combinar ou selecionar dinamicamente as suas estratégias distintas em diferentes contextos. Porém, revelar as relações entre si é deveras desafiante devido à sua grande diversidade de especificações e pressupostos. Esta observação motivou-nos a desenvolver, para cada serviço, uma *framework* genérica que abstrai os elementos comuns das suas soluções enquanto oferece parâmetros para capturar o comportamento de instâncias particulares (i.e., soluções propostas). Adicionalmente, de forma a endereçar a falta de avaliação experimental sistemática destes serviços em redes reais, realizámos uma avaliação experimental de algumas soluções representativas numa rede *ad hoc* sem-fios formada por dispositivos comuns, recorrendo a protótipos das nossas *frameworks*. Os resultados revelaram novas informações interessantes que não foram estudadas anteriormente no contexto de primitivas de comunicação para estas redes.

Palavras-chave: Redes Ad Hoc Sem-fios, Difusão, Encaminhamento, Descoberta de Vizinhos, *Framework*, IoT

Contents

List of Figures

List of Tables

Glossary

Acronyms

Symbols

Chapter 1

Introduction

Over the years, we have been witnessing the emergence of the *Internet-of-Things* (IoT): ubiquitous networks of interconnected everyday objects (e.g., vehicles, buildings, household appliances, wearables) embedded with the ability to perform computations and exchange data with other devices [1, 2]. IoT devices have become notably widespread across a multitude of use cases, being predicted that this growth will tend to rise even further in the near future [3]. Within these use cases, a vast amount of IoT applications depends on Cloud services and its deployments often rely on conventional wireless networks [4]. This architecture, however, is unsuitable for several IoT scenarios due to the following inherent limitations.

On the one hand, conventional wireless networks, leveraging on technologies such as the well-known WiFi [5] and 3G/4G/5G [6–8], are centralized and heavily rely on a pre-existing (wired) infrastructure (i.e., network backbone), containing one or more devices that coordinate the access and forward the traffic within the network. While this structure provides some benefits, namely being fairly reliable and providing high-speed and high-bandwidth transmissions, it also inhibits the flexibility of the networks since it constrains the mobility of the network's participants to areas covered by the static backbone infrastructure, and it hinders the network deployment and relocation.

On the other hand, Cloud Computing [9] is becoming a bottleneck on the performance of many IoT applications due to the ever-increasing amounts of data produced and consumed by IoT devices [3], which are rendering the Cloud unable to collect, process, and reply in useful time [10]. This hindrance sparks the need to mitigate network congestion and decrease latency. In addition, there is also the intent to reduce applications' costs by reducing Cloud resources' usage [10], and to escape from the Cloud's unsatisfactory data privacy protection [9–11].

The demand to offload computations from the Cloud motivates a paradigm shift

towards Edge Computing [11, 12], which exploits the computational capabilities of peripheral network devices that are located near end-users. However, since any device outside a Cloud's data center has the potential to contribute to Edge Computing, this paradigm has a broad spectrum of materializations [11, 13, 14]. In this sense, *wireless ad hoc networks*, emerge as a more flexible and robust platform than conventional wireless networks for materializing Edge Computing in the context of IoT. These networks are suitable candidate for deployment in situations with inadequate, inexistent, unavailable, or debilitated network infrastructures [1, 2, 15, 16], such as: areas affected by natural disasters; remote area exploration; environmental monitoring; traffic management; autonomous and smart vehicles; smart cities and homes; health care applications; military applications; and Personal Area Networks. In this regard, IoT has been inducing the contemporary reemergence of wireless *ad hoc* networks due to their eligibility to address the limitations of current IoT deployments.

1.1 Problem Statement

On wireless *ad hoc* networks, the devices, also called nodes, are typically scattered through a wide area, being unable to directly reach all the others with their transmissions — forming a multi-hop network. Consequently, the nodes must cooperate, by retransmitting messages on behalf of other nodes, so that they reach their intended destination(s).

When devising distributed applications and services [1, 2, 15] tailored to these networks, it is necessary to coordinate the node's cooperation in a distributed fashion. Nevertheless, that process can be quite challenging, and thus it becomes crucial to conceive practical abstractions that encapsulate the coordination of the communications between the network's participants — *Communication Primitives*.

Among the several communication primitives found throughout the literature of wireless *ad hoc* networks [15, 17–20], *Broadcast* and *Unicast* emerge as the most relevant. On the one hand, broadcast consists of disseminating a message throughout the network to deliver it to all the nodes [21]. On the other hand, unicast consists of sending a message towards a single destination. To enable this primitive, the nodes may need to rely on a *Routing* service which dynamically selects a sequence of nodes to forward messages towards the intended destinations [22]. Moreover, at the heart of numerous instances of broadcast and routing services, there often lies another crucial service: *Neighbor Discovery*. Neighbor discovery consists of detecting and evaluating the (wireless) links that a node can establish with other nearby nodes with whom it can directly communicate, i.e., its *neighbors* [23]. As such, *Neighbor Discovery*, *Broadcast*, and *Routing* emerge as the most key abstractions/services to support communication primitives for these networks.

Although the literature on neighbor discovery, broadcast, and routing is quite extensive, with numerous solutions and improvements proposed over the years — that explore or combine different techniques, due to the highly dynamic and heterogeneous nature of these networks (e.g., dynamic topology, movement patterns, communication

patterns) — no single solution can be considered as the most adequate in all scenarios. For this reason, it becomes essential to identify how the different solutions relate among each other to better compare, combine, or dynamically select their distinct strategies, in an effort to devise better solutions for the different key services for wireless *ad hoc* networks. However, unveiling the relations among them is quite challenging due to their highly heterogeneous specifications and assumptions, which hinders the ability to reason about their similarities and trade-offs. Although some attempts were already made in the past [24–30], they focus only on particular situations and do not address the overall problem completely.

Furthermore, the vast majority of solutions have only been evaluated resorting to simulations [31–35], since they provide an accessible, inexpensive, and controlled evaluation environment. Nonetheless, even the most detailed simulations are unable to capture the particular characteristics of real wireless *ad hoc* environments [36, 37], usually employing inaccurate models, not considering hardware limitations of wireless interfaces, or ignoring external sources of interference in the wireless medium. Although real testbeds have been employed in the past to evaluate some solutions [38–44], they generally either have static grid topologies with equidistant nodes and without external interference, which is highly unrealistic to occur in real *ad hoc* networks; or they are limited to a few nodes (less than 10), which are not enough to drive significant conclusions.

In light of the aforementioned challenges, the objectives of this thesis are two-fold. On the one hand, we intend to tackle the problem of relating the different solutions proposed for neighbor discovery, broadcast, and routing among themselves, and, on the other hand, we intend to address the lack of systematic experimental evaluation of their solutions on real *ad hoc* networks.

1.2 Contributions

To tackle the previous problems, we developed, for each key service, a conceptual framework that abstracts its common elements while offering a set of parameters that capture the behavior of particular instances. As such, these frameworks allow to better compare the distinct strategies and mechanisms of each solution. In addition, these conceptual frameworks can be materialized (i.e., implemented) to further simplify the development, implementation, and experimental evaluation of different instances of the three services for wireless *ad hoc* networks.

To address the lack of systematic experimental evaluation of these services on real networks, we conducted an experimental evaluation of some representative solutions on a wireless *ad hoc* network formed by commodity devices[1], leveraging prototypes of our frameworks. This experimental evaluation focused only on practical solutions, i.e.,

[1] a commodity device is a device that is relatively inexpensive and widely available.

solutions that can be implemented in commodity hardware since they do not require specialized hardware functionalities, such as: reporting the signal strength of each received message; Global Navigation Satellite Systems (GNSS) (e.g., GPS [45] or Galileo [46]); velocity and battery information; and transmission power selection. Despite that, our frameworks are still able to specify those solutions if such functionalities are available.

The three frameworks were designed to cooperate with each other in a cross-layer architecture due to the dependencies among them. In addtion, we intend these frameworks to operate at the application level [17], to:

- Allow each application or protocol to leverage specific tailored communication protocols for their particular needs.
- Avoid modifying the network stack of the devices, which is challenging and not always possible in commodity devices.
- Enable modifying and/or process messages (*in-network-processing*) before their retransmission to, for instance, append additional information (*piggybacking*), filter messages or data, aggregate information, or to monitor the data being transmitted, which most times are application-dependent tasks.

Security concerns related to the communication primitives are out of the scope of this thesis. For simplicity reasons, we assume that each node has only a single wireless interface, though extending the frameworks and implemented algorithms to support multiple interfaces per device is possible. We further assume that each node in the network is uniquely idendified by a unique number, which in practice can correspond to IP addresses, MAC addresses, Router-IDs, etc.

The proposed contributions of this thesis are the following:

- Provide three conceptual application-level frameworks for specifying the three essencial services to support communication primitives: one for neighbor discovery solutions, another for broadcast solutions, and the last for routing solutions.
- Provide a set of implementations, leveraging prototypes of the previous frameworks, of some of representative solutions of each service in the literature.
- Set forth an experimental comparison of the implemented solutions on a real wireless *ad hoc* network, formed by commodity devices.

The work reported in this dissertation originated the following publications:

- *Generalizing Wireless Ad Hoc Routing for Future Edge Applications* (To Appear)
  André Rosa, Pedro Akos Costa, and João Leitão
  18th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous 2021)
  Beppu, Japan, Nov. 2021

1.3 Dissertation Structure

The remainder of this dissertation is organized as follows:

- **Chapter** 2 provides an overview of the relevant literature. It starts with the related work on wireless *ad hoc* networks, discussing various types of these networks as well as the challenges inherent to the wireless medium. Then, it follows with a review of the most relevant neighbor discovery, broadcast, and routing solutions for wireless *ad hoc* networks, that have been presented in the literature.
- **Chapter** 3 presents the proposed conceptual frameworks, discussing its architecture and operation as well as providing examples of how to parametrize each one of these frameworks to specify representative solutions found on the literature.
- **Chapter** 4 details the methodology and discusses the results of the conducted experimental evaluation of the implemented solutions on a real *ad hoc* network, formed by commodity devices.
- **Chapter** 5 concludes this dissertation with some final remarks.

1.4 Research Context

The work developed on this thesis was conducted within the european project "LightKone — Lightweight Computation for Networks at the Edge" (Ref. 732505) and partially supported by FC&T through NOVA LINCS (grant UIDB/04516/2020) and NG-STORAGE (PTDC/CCIINF/32038/2017).

A portion of the work presented here was previously published and presented at SRDS [47].

Chapter 2

Related Work

Wireless *ad hoc* networks have already been heavily explored throughout the years, with a vast plethora of research dedicated to tackling their several challenges. This chapter comprises a literature review on the most relevant research related to the services to support communication primitives for these networks. As such, we start by presenting the characteristics and challenges of wireless networks (§2.1), followed by a review of the neighbor discovery (§2.2), broadcast (§2.3), and routing (§2.4) services, discussing their objectives and challenges, along with highlighting some distinguished solutions.

2.1 Wireless Networks

A wireless network is a communication abstraction comprised of a set of interconnected computational devices, called *nodes*, that communicate by exchanging messages through the wireless medium. These messages are transmitted over another abstraction called (wireless) *links*, that enable two or more nodes of the network to communicate directly without being physically connected.

Each node is uniquely identified within each network by a number (i.e., its ID), which can be, for instance, an IP address or a MAC address. The nodes can be classified as *hosts*, which are the destinations and sources of traffic, or *routers*, which forward traffic between the hosts. In turn, the links can be classified as either *bidiretional* (or *symmetric*) or *unidirectional* (or *asymmetric*) depending if messages can be, or not, sucessfully transmitted in both directions, respectivly.

According to their structure, wireless networks can be classified as either infrastructure-based, which we named *conventional wireless networks*, or infrastructure-less, called *wireless ad hoc networks*. Conventional wireless networks are centralized and supported by a wired infrastructure formed by interconnected routers, creating a wired backbone. This

organization attempts to preserve the advantages of wired networks, such as fairly reliable, high-speed, and high-bandwidth links, while offering flexibility to network hosts. However, this organization is not entirely flexible since it limits the hosts' mobility to only static areas covered by the routers and requires a pre-existing infrastructure which hinders the network deployment and relocation. On the other hand, we have wireless *ad hoc* networks, which, due to them being the focus of this thesis, we will delve into a detailed analysis next.

2.1.1 Wireless Ad hoc Networks

In wireless *ad hoc* networks, the devices communicate directly with each other without requiring any *a priori* infrastructure. As such, these networks are able to emerge as they are needed, i.e., they are formed in an *ad hoc* fashion.

Since the nodes are typically unable to directly reach all the others, they must cooperate by retransmitting messages on behalf of other nodes, for them to reach their intended destinations. In this regard, the nodes usually take the role of hosts and routers simultaneously.

The nodes are free to join or leave the network at will, at any given time, causing the network to varying in size over time. The nodes may also present multiple mobility patterns or no mobility at all. By virtue of these characteristics, the topology of wireless *ad hoc* networks is extremely dynamic, typically unstructured, and highly heterogeneous. Although this renders wireless *ad hoc* networks highly challenging to leverage, it also grants them high flexibility. Due to this remarkable flexibility, wireless *ad hoc* networks are highly versatile, being well-suited in multiple scenarios, which led to the emergence of different specialized types, which we delve into next.

2.1.1.1 Wireless Ad hoc Network Types

There are several types of wireless *ad hoc* networks that, despite having common characteristics, have enough particularities for specialized solutions to exist for each type. These types are not mutually exclusive, being possible to have hybrid networks. We highlight the following types:

Wireless Mesh Networks (WMNs) In WMNs [48], the routers form a network backbone to forward the hosts' traffic, resembling conventional wireless networks. However, unlike them, in WMNs, the backbone itself is a wireless *ad hoc* network, although hybrid settings with wireless and wired links can exist. In WSNs, the routers have little or no mobility and the hosts are mobile and usually resource-constrained. In addition, the hosts often rely on routers to communicate with other nodes, albeit sometimes they can also contribute to routing messages. Nowadays, WMNs are becoming extremely relevant to wirelessly extend existing network infrastructure. *Inter-Planetary Networks (IPNs)* [49] are a sub-type of WMNs adapted for the outer space.

Wireless Sensor Networks (WSNs) In WSNs [17, 50, 51], the nodes are usually resource-constrained and present little mobility. The nodes are also equiped with sensors to measure and collect information from the physical environment, e.g., temperature, air quality, and humidity, which is then usually aggregated into a single node, called the *sink node*. In general, minimizing the energy consumption to preserve the nodes' batteries is the main objective on these networks. *UnderWater Networks (UWNs)* [52] are a sub-type of WSNs tailored to operate underwater. Since electromagnetic waves are heavily attenuated in water, UWNs leverage acoustic waves to carry information incurring much higher propagation time.

Mobile Ad Hoc Networks (MANETs) In MANETs [15] the nodes are highly mobile, causing their topology to be extremely dynamic. *Vehicular Ad Hoc Networks (VANETs)* [53] are a sub-type of MANETs specialized on inter-vehicle communications. In VANETs, the mobility is much higher than in MANETs, and the nodes have specific movement patterns. *Flying Ad Hoc Networks (FANETs)* [16] are a sub-type of VANETs formed by UAVs[1]. In FANETs, mobility is much higher than in VANETs, and the node density is usually much lower. *Delay/Disruption Tolerant Networks (DTNs)*[2] [54] are another sub-type of MANET that experiences frequent network partitions. As such, full end-to-end routes are often impossible to determine, and thus nodes have to carry messages to forward them upon encountering other nodes.

Cognitive Radio Ad Hoc Networks (CRAHNs) In CRAHNs [55], the nodes leverage unoccupied licensed bands of the electromagnetic spectrum to avoid interference in congested unlicensed bands. As such, these networks require specialized hardware that is able to access licensed frequencies.

The taxonomy of these networks is summarized in Figure 2.1.

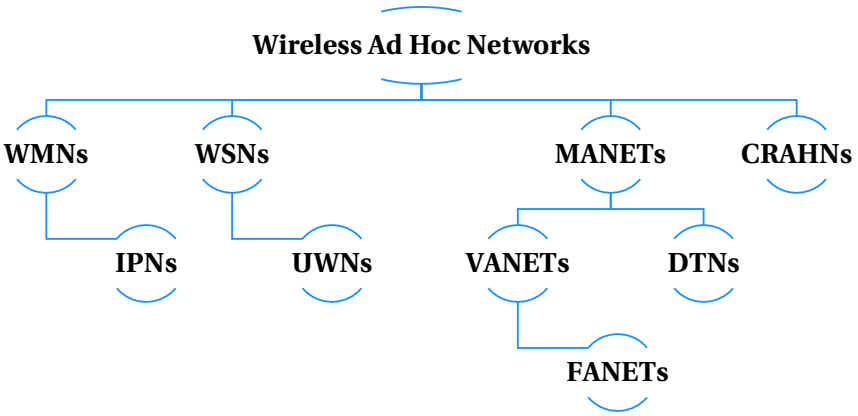


Figure 2.1: Taxonomy of wireless *ad hoc* networks.

[1] UAV stands for Unmanned Air Vehicle, commonly known as "drones".
[2] also known as Intermittently Connected Networks (ICNs)

2.1.2 The Wireless Medium

The wireless medium is anything through which information can be transmitted without relying on wires [6]. Typically, the wireless medium refers to radio waves; however, other parts of the electromagnetic spectrum [56, 57] and even acoustic waves [52] can be used. Information is encoded into these waves, forming *signals*. However, these spectrums are shared among everything (i.e., devices and objects). Thus, in the case of omnidirectional antennas, where transmissions are simultaneously propagated in all directions, whenever a transmission occurs, every device in the sender's vicinities may receive it, i.e., wireless communications have a *broadcast nature*. This phenomenon is commonly referred to as *one-hop broadcast* [58], which can be leveraged to abstract *one-hop unicast* (or *point-to-point*) and *one-hop multicast* links by having nodes to which messages are not destined ignoring them. However, due to this broadcast nature, wireless communications face many challenges, that need to be addressed to enable successful communications.

2.1.2.1 Challenges

Fading Fading corresponds to waves losing energy. The lower the energy, the more difficult it is to distinguishing between signals and noise, eventually being impossible to retrieve any information. The intensity of this energy is called the Received Signal Strength Indicator (RSSI) [6, 15] and its proportion over the intensity of background noise is called Signal-to-Noise Ratio (SNR) [6]. There are several types of fading caused by multiple factors [6], such as the distance to the transmission source (*Path-Loss Fading*), waves' attenuation and blockage by objects (*Shadow Fading*), the relative movement of the sender and receivers during transmissions (*Doppler Fading*), and wave's frequencies. The maximum distance at which signals from a given node are correctly received is called its *transmission range*. However, these ranges are usually not uniform and may fluctuate over time [6] due to all sources of fading and interference.

Interference Whenever waves overlap with each other, even if partially, the carried signals might get corrupted impossibilitating their retrieval [6]. This phenomenon is called *interference* and directly detecting it in the wireless medium is impossible since nodes cannot transmit and listen (on the same frequency) simultaneously, as their signals' intensity dominates all the others. Interference results from, e.g., nearby objects emitting waves (e.g., microwave ovens) or causing the reflection, scattering, and refraction of waves that causes signals to be propagated by multiple paths that interfere with each other [6]. A particular case of interference, called *collisions*, occurs when multiple nodes transmit simultaneously [6]. The *Hidden Terminal Problem*[3] [6, 59] is a particular situation that arises when multiple nodes, that are not within the range of each other yet are within range of a receiver, transmit simultaneously causing a *collision at the receiver*. To exemplify it, suppose we have three nodes **A**, **B**, and **C**, with **B** being within the transmission ranges

[3] "terminal" is another name used for network hosts.

of **A** and **C** yet they do not reach each other, as illustrated in Figure 2.2. When **A** and **C** transmit simultaneously they cause a collision, resulting in **B** not receiving any signal. In this case, **A** is *hidden* to **C** and vice-versa, since they are not aware of each other.

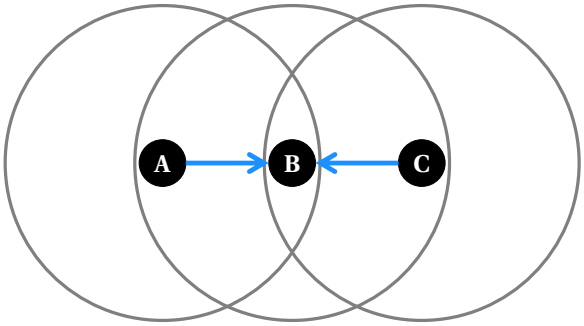


Figure 2.2: The Hidden Terminal Problem.

Contention To mitigate collisions, the nodes need to compete for the isolated usage of the wireless medium, resulting in some nodes postponing or even cancelling their transmissions —*Contention* [6] —and is enforced by *Medium Access Control* protocols (later discussed in §2.1.2.2). The *Exposed Terminal Problem* [59, 60] is a particular situation that leads to unnecessary contention and arises when nodes that would not cause collisions at the receiver contend. To exemplify it, suppose we have four nodes **A**, **B**, **C**, and **D** and that **A** and **D** are within **B**'s and **C**'s transmission ranges, as illustrated in Figure 2.3. When **B** is transmitting to **A**, if **C** intends to transmit to **D** it must wait eventough both **A** and **D** could simultaneously receive. There is a collision; however, only within the overlap of the senders' transmission ranges (*collision at the sender*). In this case, **C** is *exposed* to **B**.

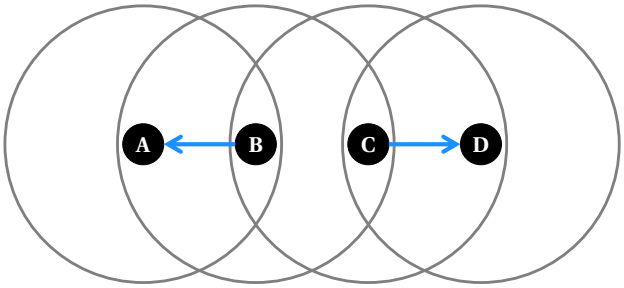


Figure 2.3: The Exposed Terminal Problem.

Gray Zones The *Gray Zone Problem* [61, 62] consists of the existence of zones located at the fringe of transmission ranges where some transmissions are successfully received while others are not (illustrated in Figure 2.4). This phenomenon causes unstable links and can happen, for instance, due to different data-rates (which have different frequencies) being used in each transmission (e.g., it happens in WiFi with one-hop broadcasts and unicasts [63]); different message sizes; or fading and interference.

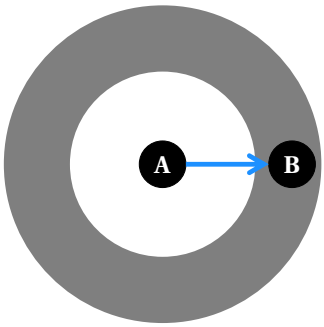


Figure 2.4: The Gray Zone Problem.

Network Partitions Interference and mobility may lead to the fragmentation of the network into several isolated zones called *partitions*. In these situations, nodes belonging to one partition are unable to communicate with nodes belonging to other partitions. These partitions can be temporary, lasting a few seconds, or long-lived.

Network Congestion Whenever nodes produce or receive more traffic than they can handle, they become *congested* [64] and start dropping messages. As such, congestion results in contention, message losses, increased latency, and low throughput, effectively impairing communications. Excessive contention or too many retransmissions to overcome collisions may lead to congestion that causes more contention and message losses.

2.1.2.2 Medium Access Control (MAC) Protocols

These protocols regulate the usage of the wireless medium [6, 65–67] by coordinating contention and overcoming losses with retransmissions. At this level, the data units exchanged between the nodes are called frames, which may contain messages or control information for the MAC protocol. We highlight the following MAC protocols:

Time-Division Multiplex Access (TDMA) TDMA [66–70] is a family of protocols that reserve, for each node, a time-slot where it is allowed to transmit. This scheduling usually guarantees that concurrent transmissions do not collide and it can be static or dynamic. When nodes transmit frequently, TDMA avoids collisions while fairly dividing the access to the medium. However, when only a fraction of nodes often transmits, it incurs needless delays, i.e., it suffers from the exposed terminal problem. Unfortunatly, TDMA requires the nodes to be synchronized [6, 67] and computing optimal time-slots is highly complex.

ALOHA In ALOHA [6] the nodes immediately transmit and wait for an acknowledgment (ACK) from the receiver. If no ACK is received, the sender attempts again after some random period. Unfortunatly, when nodes often transmit, ALOHA causes many collisions. As a result, several enhancements [6, 71, 72] were proposed to increase ALOHA's robustness, which usually combine ALOHA with some variant of TDMA. In addition, ALOHA is only employed for unicast transmissions.

Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA) In Carrier-Sense Multiple Access (CSMA) [6] each sender senses the medium before transmitting to avoid collisions. If there is an ongoing transmission, the sender waits a random period before attempting again, executing ALOHA otherwise. This strategy only avoids collisions at the sender, thus suffering from the hidden and exposed terminal problems. To tackle the former, CSMA/CA [6, 73, 74] extends CSMA with virtual sensing the medium, by exchanging two control frames. First, the sender transmits a Request-to-Send (RTS) to the receiver, which replies with a Clear-to-Send (CTS). If the sender receives the CTS, it transmits the message, attempting again after a random period otherwise. Any other node that receives an RTS or CTS defers its transmissions, thereby mitigating the hidden terminal problem. However, virtual sensing does not avoid this problem entirely since faded signals that can no longer be recovered can still cause interference. This occurence leads to some nodes not receiving the CTS, yet their transmissions colliding at the receiver nonetheless [74]. Furthermore, virtual sensing further amplifies the exposed terminal problem since all the nodes that receive a CTS abstain from transmitting, even if no unfavorable collision would occur. CSMA/CA is only employed in unicast transmissions.

Multiple Access with Collision Avoidance (MACA) MACA [60] is a protocol derived from CSMA/CA that attempts to mitigate exposed terminal problem by removing the physical sensing of the medium, only reling on virtual sensing since only collisions at the receiver prevent receptions. MACA further enhances the RTS/CTS exchange by: waiting only a short period after receiving an RTS, enough for the sender to receive the CTS; allow nodes to transmit despite receiving an RTS if they do not receive a CTS (further tackling the exposed terminal problem); include in the RTS and CTS the size of the message to be transmitted to enable the nodes to determine how long the medium is occupied; and bypass the RTS/CTS exchange whenever messages have similar sizes to those frames.

Broadcast Medium Window (BMW) BMW [75] complements CSMA/CA by addressing the hidden terminal problem in one-hop broadcasts. It requires every node to be aware of all the receivers, i.e., all other nodes with whom they have a bidirectional link (discussed in §2.2). First, a unique sequence number (SEQ) is assigned to each broadcast message waiting for transmission. Next, the sender selects a single receiver, using a Round Robin policy, with whom it executes CSMA/CA. However, the RTS is enriched with the SEQs of the messages waiting to be acknowledged by all receivers, and the CTS is enriched with the SEQ of the message the receiver is waiting for. The sender transmits the message whose SEQ is included in the CTS and waits for the ACK. All the other receivers may also receive the message; however, they do not respond with an ACK. The verification of reception by an unselected receiver is done in subsequent (broadcast) transmissions to that receiver when it is selected.

Frequency-Hopping Spread Spectrum (FHSS) FHSS [6] is a technique[4] to mitigate interference by continuously changing ("hopping") the frequency of the waves in a pseudorandom pattern. To achieve this, it requires the nodes to pre-establish the sequence of frequencies to use and be synchronized to switch frequencies simultaneously.

2.1.2.3 Wireless Technologies

There are multiple technologies that enable the creation of wireless *ad hoc* networks [1, 2, 6, 76]. Each of them presents distinct features, such as the frequency of waves, transmission ranges, available data-rates, and MAC protocol. The most common technologies are:

Wireless Fidelity (WiFi) WiFi [5, 6, 77] is the common name given to the IEEE 802.11 family of wireless standards. It typically operates at the 2.4 GHz and 5 GHz unlicensed frequency bands of the electromagnetic spectrum. The MAC protocol employed in WiFi results from a combination of CSMA/CA and MACA (§2.1.2.2). This family of standards has several amendments, providing different data-rates (raging from 1 Mbps to 1 Gbps), being WiFi considered a medium-range with relatively high-data-rate technology. WiFi is most well-known for creating conventional wireless networks; however, it also supports an *ad hoc* and mesh modes [78]. The mesh mode is distinguished from the *ad hoc* mode by directly supporting routing at the MAC level.

Bluetooth Bluetooth [79, 80] is a wireless technology designed with energy efficiency in mind. It specifies multiple classes, each with different transmission ranges (up to 200 m) and data-rates (from 125 kb/s to 2Mb/s), being considered a low-range and low-data-rate technology. Bluetooth operates at the 2.4 GHz unlicensed frequency band and employs a variant of FHSS (§2.1.2.2). It supports a master/slave and, more recently, multi-hop mesh topologies with a practically unlimited number of nodes.

IEEE 802.15.4 The IEEE 802.15.4 [81] standard targets low-power and low-data-rate (from 20 kbps to 250 kbps) devices, usually found in WSNs. It also operates at the 2.4 GHz unlicensed frequency band and employs CSMA/CA as MAC protocol, with an optional combination with TDMA. This standard serves as the basis for several other technologies; in particular, we highlight ZigBee [82] and 6LoWPAN [83]. ZigBee adds network functionalities, such as routing, being each network composed of three types of devices: the coordinator, routers, and end-devices. The coordinator forms and manages the network, routers forward messages between nodes, and end-devices produce data. 6LoWPAN specifies how to apply IPv6 over low-power networks based on the IEEE 802.15.4 standard to enable using standard Internet Protocols on these networks, unlike ZigBee. As such, it does not specify particular routing protocols to use.

[4] Since it is used to avoid interference, we are considering it along with MAC protocols. However, according to the OSI model, FHSS is part of the PHY layer instead of the MAC layer.

2.1.3 Discussion

In this section, we discussed the characteristics of wireless networks, in general, and of wireless *ad hoc* networks, in particular. We revied wireless *ad hoc* network types, the characteristics and challenges of the wireless medium, MAC protocols, and wireless technologies. Since in this thesis we intended to study communication primitives for wireless *ad hoc* networks formed by commodity devices that may present some mobility, we must select the most suitable network type and wireless technology.

All the discussed wireless *ad hoc* network types are intended for specific use cases. WSNs are comprised of resource-constrained (and often specialized) devices, limiting the strategies of communication protocols and thus are not suitable. Considering that we are not concerned with the network operating as a wireless backbone to other devices, WMNs are also not adequate. Since leveraging licensed bands requires specialized hardware, not available in commodity devices, CRAHNs are also not suitable. Finally, since we are not concerned with constant mobility, MANETs are not eligible as well. In this regard, we chose to focus on generic wireless *ad hoc* networks since they abstract from the other types' specific characteristics, thereby enabling devising more general solutions. Nonetheless, our solutions can be transposed and adapted to any other network types.

Considering the wireless technology, the majority of commodity devices is equipped with Bluetooth and WiFi antennas. ZigBee and 6LoWPAN are usually only present in specific hardware designed for WSNs. Among Bluetooth and WiFi, we chose to implement our frameworks for the latter since it is more appropriate to support distributed applications for commodity devices due to its higher data-rates. Nonetheless, our frameworks can be implemented over any wireless technology, in theory, to support specific use cases. We will resort to WiFi's *ad hoc* mode, instead of the mesh mode, to enable designing customizable and flexible neighbor discovery, broadcast, and routing solutions.

As previously stated, the WiFi's MAC protocol is a variant of CSMA/CA combined with MACA. However, it does not provide any hidden terminal prevention mechanism for one-hop broadcasts, which is heavily leveraged in (network-wide) broadcast, and suffers from the exposed terminal problem. Although, in theory, we could replace CSMA/CA with other protocols, this is usually not feasible in commodity devices since it involves modifying the communication stack (part of the Operating System) or firmware of wireless interfaces [84]. Furthermore, even though MAC protocols try to mitigate the wireless medium's challenges, they are not enough to ensure that communications are reliable. Therefore, the unreliability of communications through the wireless medium must be addressed by communication protocols for wireless *ad hoc* networks. In particular, these protocols must employ mechanisms to mitigate collisions and contention and identify unidirectional and unstable links. For those reasons, communication primitives might benefit from incorporating or adapting some techniques of MAC protocols, such as: backoff-periods to desynchronize transmissions, ACKs in intermediate transmissions, BMW's collision avoidance for one-hop broadcasts, or TDMA's transmission slots.

2.2 Neighbor Discovery

As previously stated, communication primitives coordinate the nodes' cooperation to deliver messages to their destinations. In this regard, a crucial piece in numerous solutions is each node being aware of the nodes with whom it can directly communicate — its *neighbors* or *neighborhood*. This information is particularly relevant in several broadcast (§2.3.2.2), routing (§2.4.2.2), MAC (§2.1.2.2), data aggregation [17] and distributed storage [15] protocols as well as contact-tracing applications [85].

Neighbor discovery is the process of detecting and evaluating the links between a node and its neighbors and is typically accomplished by each node transmitting, through one-hop broadcast, messages called *hellos* (or *beacons*) to announce its presence [86–88]. Hellos contain the sender's ID and other complementary information that characterize it, such as its hello transmission period, its coordinates, its velocity, or its willingness to forward messages of other nodes [86, 89–93]. Different solutions encapsulate different data in these messages.

Upon receiving a hello, the nodes become aware that they are within the sender's transmission range. However, the reverse cannot be assumed since the nodes' transmission ranges might be different and are not uniform and fluctuate over time (§2.1.2.1), leading to only one node being able to receive the messages sent by the other. In this case, the node that is unable to receive is a *unidirectional (or asymmetric) neighbor* of the other, yet the latter is not a neighbor of the first. When both nodes consider each other as unidirectional neighbors, they become *bidirectional (or symmetric) neighbors*. Detecting bidirectionality is crucial to enable two-way communications, being that unidirectional neighbors are often disregarded by most protocols that leverage neighborhood information. Bidirectionality is detected through the reception of a *hello acknowledgment (hack)* from a neighbor [87, 89, 94], which indicates that it successfully received a hello from the local node. Unlike hellos, hacks are destined to a specific neighbor; however, they are usually also one-hop broadcast to group several hacks into a single message. In addition, hacks also enable discovering the neighbors' neighbors, i.e., two-hop neighbors, by eavesdropping hacks meant for other nodes. This two-hop neighborhood information is used by some broadcast and routing solutions [86, 89, 95, 96].

In theory, some MAC protocols can perform neighbor discovery, e.g., through the RTS/CTS exchange, the reception of frames and ACKs, or MAC beacons [6]. However, often MAC protocols cannot be modified and are not exposed to upper layers[5], especially in commodity hardware [63, 84]. As such, neighbor discovery is usually performed by higher layers protocols.

Due to the highly dynamic topology of wireless *ad hoc* networks, neighborhoods frequently change over time, even without node mobility due to the particularities of the wireless medium [97]. As such, accurately determining and maintaining neighborhoods is rather complex and incurs many challenges.

[5] Do not inform upper layers of MAC layer information, such as acknowledgment reception.

2.2.1 Challenges

Discovery solutions must tackle several challenges to determine accurate neighborhoods without incurring excessive transmissions. We highlight the following:

Neighbor Detection The accuracy of neighbor detection dependents on the interval between consecutive hello transmissions. The smaller these intervals, the more accurate is detecting neighbors; however, the higher the risk of incurring collisions and contention (§2.1.2.1). In contrast, the larger these intervals, the higher the chance of nodes not being aware of some of their neighbors, i.e., *false negatives* [23]. Therefore, choosing proper intervals is not trivial. Thus, some solutions employ adaptative intervals, perform discovery only when needed, or piggyback hellos in outgoing messages to improve discovery performance [91]. Moreover, to quickly mitigate false negatives, some solutions have the nodes transmit a hello (after a small jitter) upon detecting a new neighbor [86].

Loss Detection Whenever a given number of hellos, called *hello misses*, are not received from a neighbor, the neighbor is considered lost (unreliable failure detection). If the misses are small, the nodes may prematurely consider loosing a neighbor, yet still being within its transmission range (*false negatives*) [63]. However, if the misses are high, the nodes are slow to detect losses, assuming they are neighbors with nodes that are not within their transmission range anymore, i.e., *false positives* [23]. Furthermore, larger hello intervals also cause slower loss detections, while lower intervals lead to quicker loss detections.

Bidirectionality Detection Similar to hellos, the strategy, intervals, and misses employed with hacks influence the accuracy of bidirectionality detection. In addition, bidirectionality also incurs in false negatives, when a bidirectional neighbor is not considered as such, and false positives, when a neighbor is considered bidirectional yet its not. The main hack exchange strategies are piggyback them with hellos [86], send in separate periodic messages [87], or immediately reply upon the reception of a hello [98].

Unstable Neighbors Occasionally, the nodes may sporadically receive hellos, leading to the detection of new neighbors that are lost shortly afterward. Moreover, these receptions may also happen in small enough intervals, yet still irregular, to not consider these neighbors lost. Both these cases lead to *unstable neighbors* which severely affect the performance of communication protocols [23, 97, 99]. In an attempt to tackle this phenomenon, some discovery solutions assign to each neighbor *link quality metrics* that express the likelihood of successfully communicating with that neighbor [23, 25, 99]. Each link has two values, one in each direction, according to the view of each node, since the conditions in each direction might be different. These metrics are used to favor some neighbors over others or even to filter "bad" neighbors by employing *link acceptance policies* [61, 86, 100].

Examples of these metrics can be the average RSSI or SNR of hellos [25, 86, 87] or the number of hellos received within a time window [25, 86, 87, 99].

Ghost Neighbors In case there are gray zones where one-hop broadcasts are received yet unicasts are not, the reception of hellos cause the detection of a *ghost neighbor* with whom it is impossible to communicate through unicast [61, 63]. This situation is especially relevant in WiFi since it employs different data-rates for one-hop broadcasts and unicasts. One way to tackle this is manually setting the data-rates to be equal [63] or to filter neighbors whose hellos have low SNR (or RSSI) [61]. However, not all devices support these features. Alternatively, link qualities can be measured with unicast traffic [25].

2.2.2 Discovery Algorithms and Protocols

Here we discuss existing discovery solutions, starting with the proposal of a classification scheme (§2.2.2.1), followed by examples (§2.2.2.2), and finishing with some frameworks (§2.2.2.3).

2.2.2.1 Classification

Discovery solutions can be classified according to their discovery strategy as:

- **Passive**: Hellos and hacks are piggybacked in outgoing messages, resulting in no additional transmitions to perform discovery [25, 99, 101]. However, this strategy is intrinsically dependent on the traffic pattern. If nodes transmit sporadically, the discovered neighborhoods may be short-lived. A particular case of passive discovery is when routing control messages also include discovery information [94, 98, 102].
- **Active**: Hellos and hacks are transmited in explicit discovery messages. There are two main active approaches:
  - **Periodic**: Discovery messages are periodically transmitted [86, 87]. The intervals between consecutive transmissions do not need to be static and may change over time. In fact, each transmission should employ a random jitter to avoid synchronizations with other nodes [103]. Periodic discovery is the most common type of discovery since it is quite robust. However, it is costly in the number of transmissions performed.
  - **Reactive**: Discovery is only performed when needed, by sending explicit discovery messages upon some event, e.g., by request from other protocol or changes to neighborhood information [101]. This discovery strategy incurs only a few transmissions; however, since there is no proactivity in maintaining neighborhoods, they can quickly become stale.
- **Hybrid**: Discovery is performed through a combination of the previous approaches, performing discovery by eavesdropping on ordinary traffic, complemented with

transmitting explicit discovery messages, for instance, when there is no recent traffic [25]. As such, this strategy decreases the number of transmissions while maintaining the robustness of the periodic approach.

2.2.2.2 Examples

Most recent studies on neighbor discovery focus on decreasing the node's energy consumption by turning off the wireless radio during some periods [104, 105]. These solutions are only relevant to energy-constrained devices and are thus beyond the scope of this thesis. We highlight the following discovery protocols:

Neighborhood Discovery Protocol (NHDP) NHDP [86] is a periodic discovery protocol that includes the hacks in the periodic hellos. Throught these hacks, NHDP determines two-hop neighborhoods. In addition, NHDP has optional reactive hello transmissions when the local neighborhood changes. NHDP does not specify any concrete link-quality metric, allowing any to be used, and has an optional link acceptance policy, called *link hysteresis*, which filters neighbors with low link quality. NHDP also contains an extension mechanism that allows exchanging additional information with neighbors by appending it in hellos and hacks, which is fundamental to the OLSR routing protocol [89].

Babel's Discovery Protocol Babel's routing protocol [87] is periodic and performs bidirectionality detection by transmitting hacks (called "I Heard You" or IHU) in periodic messages independent from hellos, usually with a higher period to reduce the amount of control traffic. It also has optional reactive hello transmissions when the local neighborhood changes and does not specify any link-quality metric, allowing any metric to be used. In addition, it also supports the transmission of hellos with unicast to perform link quality measurements or to overcome losses of one-hop broadcasted hellos.

BATMAN's Discovery Protocol BATMAN's discovery protocol [98] can be considered both periodic and passive. When isolating the discovery process separately from the remaining BATMAN protocol, it periodically transmits hellos, which are immediately replied to with reactive hacks. However, BATMAN optimizes discovery by combining it with disseminating routing control information, turning the discovery to be passive. In this case, starting the broadcast of control messages serve as an implicit hello, and the retransmition of those control messages serves as implicit hacks.

Turnover-based Adaptive HELLO Protocol (TAP) TAP [106] is a periodic discovery protocol that dynamically adapts each node's hello transmission interval by comparing the number of new neighbors detected within a time window, called the *turnover*, with the optimal turnover they should be detecting. When the turnover is lower than the optimal value, the hello interval is increased. When the turnover is higher, the hello interval is decreased. In this way, TAP enables tackling the overestimation and underestimation of

the hello interval each node should employ. However, the computation of the optimal turnover must be done beforehand and assumes that all nodes are moving at a constant speed and in a random direction, which may not be realistic in a real network.

Event-Based Hello Protocol (EHBP) This discovery protocol [101] is a hybrid protocol where the nodes only send hellos in case they recently sent any non-discovery traffic. Therefore, nodes are only detectable by others when they are transmitting regular traffic, thus decreasing the number of transmissions incurred by the discovery process. However, this strategy may lead to unstable neighborhoods in case traffic patterns are irregular.

2.2.2.3 Frameworks

Here we discuss some frameworks that capture the behavior of discovery protocols for wireless *ad hoc* networks. During the literature review, we only found a few frameworks related to neighbor discovery. Next, we discuss them:

Efficient and Accurate link-quality monitoR (EAR) EAR [25] is a neighbor discovery framework that performs link quality estimations through unicast transmissions to increase its accuracy. The framework is composed of three complementary strategies: passive, cooperative, and active monitoring. Active monitoring consists of explicitly exchanging discovery messages with neighbors. Passive monitoring consists of measuring the delivery ratio of messages received from a neighbor. Cooperative monitoring consists of measuring the delivery ratio of messages sent by the neighbor to other node[6]. The execution of EAR is divided into two phases: measurement period and update period. During the measurement period, each node estimates the receiving link quality with each neighbor by adaptively selecting one of the three previous strategies according to the traffic. At the end of this phase, the nodes may change the measured receiving quality with their neighbors (sent through hacks) if it changed. Next, during the update period, nodes process the link quality changes informed by the neighbors and update the recorded link qualities, in each direction, with the measured and exchanged information.

Spear Spear [24] is a neighbor discovery framework for WSNs (§2.1.1.1). It comprises four modules: application module, energy module, algorithmic module, and communication module. The application module gathers the application's latency constraints in neighbor discovery and informs the energy and algorithmic modules. The energy module determines the period in which nodes have their radio turned-on for listening to incoming messages, based on application constraints and the remaining battery. The algorithmic module determines the discovery schedule, i.e., when hellos should be sent, considering

[6] Requires promiscuous mode in the radio, which is not always supported in commodity devices.

the periods when the radio is enabled. The communication module performs the messages' transmissions, according to the discovery schedule, and registers the neighbors' information.

2.2.3 Discussion

In this section, we discussed neighbor discovery in wireless *ad hoc* networks. We started by reviewing its challenges and presented some discovery solutions and frameworks. Furthermore, we proposed a classification scheme for the discovery strategies.

In the course of our study on neighbor discovery, it emerged that discovery solutions share several design patterns in common, in addition to managing a neighbors table. A central piece in any discovery solution is the strategy employed to exchange hellos and hacks with the neighbors, which we named the *Discovery Pattern*. This strategy is what determines the type of solution according to our classification scheme. Examples of it are to piggyback hellos and hacks in other messages (passive), explicitly send them upon a periodic timer (periodic) or some other event (reactive), or any combination of these strategies (hybrid). Another relevant aspect of discovery solutions are the *Transmission Periods* which determine the intervals between consecutive discovery messages, which may be null (on non-periodic solutions), static (always the same period), or dynamic (adapting according to some algorithm). Other fundamental component in discovery solutions tailored for real networks is measuring the reliability of the links with neighbors, estimated trough *Link Quality* metrics. Several examples of such metrics were previously discussed in §2.2.1. An aspect strongly related to link quality are *Link Acceptance* policies, which are present in some solutions to filter (i.e., ignore) unstable neighbors. Finally, discovery solutions must be easily extensible, allowing to exchange extra information on the hellos and hacks, e.g., the nodes' coordinates, and enabling filtering some of these messages, e.g., invalid digital signatures. We named this component the *Discovery Context*.

We highlight a lack in the literature of frameworks to specify discovery solutions for wireless *ad hoc* networks. Of the five components we identified, the discussed frameworks only identify a few. EAR identifies the discovery pattern and the link quality metric. Its discovery pattern can have three variants: passive, cooperative (a particular type of passive), and active monitoring (periodic). EAR's discovery patterns, however, do not specify how to detect new neighbors but only how to accurately measure link quality with already found neighbors. Furthermore, this framework does not address transmission periods, link acceptance policies, nor discovery contexts. The Spear framework identifies three of our components: its algorithmic module resembles the discovery pattern combined with the transmission period, and the communication module is similar to the discovery context. Addressing the conservation of the devices' energy is out of the scope of this thesis, which is a main focus of Spear. Moreover, this framework does not address link quality metrics, link acceptance policies, nor diverse discovery patterns.

As a final note, we highlight that there is a lack of a generic and flexible discovery framework that considers the five components we identified and that the existence of such a framework is extremely relevant to study and improve the performance of protocols that levarage neighborhood information, in particular broadcast and routing solutions.

2.3 Broadcast

One of the most essential communication primitives in wireless *ad hoc* networks is the *network-wide broadcast*, or simply *broadcast* of a message [107, 108]. Broadcast is remarkably important for supporting a panoplia of distributed applications and services on these networks, in particular routing [89, 94, 98], multicast [109], geocast [110], resource discovery [111], and data replication [112].

Broadcast consists of delivering a message, sent by a single node called the broadcast *source*, to all the nodes belonging to the network (including the source itself) [113]. For this purpose, the nodes must distributively cooperate, by retransmitting each message, so that it is disseminated throughout the entire network. In this sense, (network-wide) broadcast often leverages one-hop broadcast to deliver messages to all the neighbors of each node with a single transmission. This strategy is particularly relevant since it conserves energy and reduces the wireless medium's occupation compared with retransmitting through unicast to each neighbor. However, it is crucial not to confuse the concepts of one-hop broadcast and (network-wide) broadcast since they have different scopes yet similar objectives. In fact, we can generalize both types of broadcast by defining *r-hops broadcast* as delivering a given message to all nodes within a radius of r hops from the source. As such, the r-hops broadcast encompasses both the one-hop broadcast ($r = 1$) and the network-wide broadcast ($r = \infty$). Such broadcast variant is employed by some routing solutions to limit the scope of the dissemination of some control messages [29, 114–121]. The r-hops broadcast primitive can be accomplished by appending a *Time-to-Live (TTL)* field in each message [90, 122], starting with the value r. Before being retransmitted, the new TTL value of each message will be the maximum TTL from all the copies of the message received, decremented by one. If the new TTL becomes equal to zero, the message is not further retransmitted.

During the dissemination of a broadcast message, due to relying on one-hop broadcast, it is natural that some nodes receive several copies of the same message through different neighbors, called their *parents* for that particular message [123]. In this regard, it is crucial to identify the reception of copies to avoid duplicate deliveries and to avoid endless retransmissions of the same message. This objective can be achieved by extending each message with a unique identifier (the message ID), which is recorded by the nodes that receive the message so that they only process and deliver it once. As such, upon receiving a message, the nodes simply record its reception and discard it if its ID was already recorded. This ID can be, for instance, randomly generated or a monotonically increasing sequence number unique for each broadcast source.

Unfortunately, the wireless medium's highly unreliable and dynamic nature allied with no MAC strategy for one-hop broadcasts (§2.1.2.1) hinder the reception of messages. Consequently, the broadcast of messages through the wireless medium is far more complicated than in wired networks, incurring many challenges.

2.3.1 Challenges

We highligh the following broadcast challenges:

Transmission Failures As previously stated, ensuring that all nodes deliver a message in a system where the nodes interact through the wireless medium is no easy matter since it is usual for transmissions not to be received by some nodes due to collisions. Furthermore, the lack of hidden terminal mitigation for one-hop broadcasts in the MAC protocol further exacerbates this phenomenon. However, employing mechanisms to overcome such losses (e.g., retransmissions with explicit acknowledgments) increases contention, collisions, and congestion. Therefore, in wireless *ad hoc* networks, ensuring that all nodes always deliver all messages is not a trivial task, and thus the broadcast is often *unreliable*[7], i.e., it strives to deliver each message to all nodes, yet it does not guarantee it [107]. Although this may seem problematic, a reliable broadcast is often unnecessary [89, 107]. The *reliability* of a broadcast algorithm corresponds to its effectiveness in delivering messages to all nodes, being that the higher its reliability, the better the algorithm [124].

Broadcast Storm Problem During the broadcast of a message throughout the network, neighboring nodes are more likely to attempt to retransmit it simultaneously, increasing contention and the probability of collisions occurring. Furthermore, during this process, it is natural that many retransmissions of some nodes are only received by nodes that had already received the message through previous transmissions, i.e., many retransmissions are *redundant*. These phenomena are collectively referred to as the well-known *Broadcast Storm Problem* [107], that arise whenever multiple nearby nodes blindly retransmit each message. With the goal to avoid broadcast storms, broadcast algorithms typically avoid explicit acknowledgment of messages and having all the nodes performing retransmissions of each message while exploring complementary mechanisms to maximize their reliability. There are two strategies to mitigate broadcast storms: *i*) decrease the number of redundant retransmissions, by inhibitting somes nodes from retransmitting each message, and *ii*) desynchronize the retransmissions of nearby nodes, by delaying them for a short random period called *jitter* [103] or *random assessment delay (RAD)* [125]. The *cost* of a broadcast algorithm corresponds to the number of retransmissions performed to disseminate each message. For all the nodes to receive a given message (*perfect reliability*), the set of nodes that retransmit it must form a Connected Dominating Set (CDS). The minimum cost that enables achieving perfec reliability, i.e., the *perfect cost*, is achieved

[7] Also called *best effort* or *probabilistic*

when this set of nodes form a Minimum Connected Dominating Set (MCDS) [126, 127]. However, determining MCDSs is highly complex, and thus broadcast solutions endeavor to approximate them. In this sense, employing jitters also allows nodes to acquire information through the reception of copies, which can guide their retransmission decision [96, 107, 128].

Delivery Latency The *latency* of a broadcast algorithm is defined as the time required for all nodes to deliver each message [107]. By employing jitters to avoid collisions, the propagation of each message gets delayed at each node and accumulates these delays at each hop traveled. Hence, the usage of jitters and the selection of the nodes to retransmit (so that messages travel through the shortest paths) have to be carefully configured to achieve the lowest latency. Broadcast latency is often overlooked; however, it is remarkably important in some cases, such as route discovery in routing protocols [94, 102].

In view of these challenges, broadcast solutions strive to simultaneously maximize the reliability while minimizing both the cost and the latency. However, these objectives are highly conflicting since they are inter-dependent. To obtain high reliability, broadcast solutions must avoid broadcast storms while attempting to reach all the nodes. As aforementioned, this is achieved by decreasing the number of redundant retransmissions (lower cost) and by increasing the maximum jitter (higher latency). Furthermore, the higher the number of retransmissions (higher cost), the higher the chances of collisions (lower reliability) and the larger the maximum jitter has to be to avoid them (higher latency). However, decreasing the cost might prevent losses from being compensated by redundant retransmissions and the nodes that retransmit from forming a CDS —a condition called *overcancelation* [129] —thereby penalizing the reliability. Moreover, redundant retransmissions (high cost) may be unavoidable while propagating messages through all the shortest paths to achieve lower latency or the nodes may prematurely decide to retransmit due to not having enough time to acquire sufficient information that enables them to cancel their retransmissions.

2.3.2 Broadcast Algorithms and Protocols

Here we discuss broadcast solutions that have been proposed to address the previous challenges. We start by presenting how these solutions can be classified (§2.3.2.1), followed by some examples (§2.3.2.2). Finally, we discuss some broadcast frameworks (§2.3.2.3).

2.3.2.1 Classification

To guide their decisions regarding the need to retransmit and computing jitters, broadcast solutions resort to several techniques that may leverage different information. As such, broadcast solutions can be classified according to the nature of this information. This

classification, however, is not exclusive, being that several solutions employ and combine multiple techniques of different types. We propose the following classification scheme:

- **Probabilistic**: In this type of solution, each node retransmits with a given probability. These probabilities can be static [15, 107, 130, 131] or dynamic [15, 125, 130, 132, 133]. Probabilistic solutions are more suitable for dense and homogeneous networks since they do not determine the importance of nodes for propagating messages [15].

- **Copy Aware**: This type of solution leverages information extracted from the reception of copies. This information can be implicit or explicitly piggybacked on messages. We highlight three relevant sub-types:
  - **Copy-Count-Based**: These solutions resort to the number of copies received, either to influence the decision to retransmit [15, 107, 123, 129, 134–136] or to compute jitters [125, 129, 137]. Copy-Count-Based solutions may fail to determine the importance of nodes for propagating messages [123, 129].
  - **Hop-Count Aware**: This type of solution leverages the number of hops traveled by each copy received to guide their decisions [15, 137, 138].
  - **Route Aware**: In this type of solution, each message carries the route(s) it traveled, i.e., carries the IDs of the nodes that retransmitted that message [96]. Typically, these routes have a maximum number of nodes (or hops) containing only the most recent retransmissions. A particular case of this is *Parent-Aware* solutions [123, 129], which piggyback in each message the parents (or only the first) so that the nodes that received the message know its two-hop routes.

- **Position Aware**: This type of solution resorts to the nodes' positions to estimate their Expected Additional Coverage (EAC) or to compute jitters. The positional information is typically carried in the messages. However, we distinguish this type from Copy-Aware since the information carried has positional semantics. The positions can either be exact or relative to other nodes, leading to two sub-types:
  - **Distance Aware**: These solutions leverage the relative distances between a node and its parents to compute its EAC [107, 123, 135, 139, 140]. If the distances are small (i.e., below a given threshold), the EAC will be small, and thus the node should not retransmit. Distances can be estimated from the RSSI, SNR, or the node's coordinates. Jitters may also be computed proportionately to the distances, giving priority to nodes further from their parents [129, 134].
  - **Location Aware**: These solutions resort to coordinates to obtain more accurate estimations of the EAC of each node [15, 107, 141, 142].

- **Energy Aware**: These solutions consider the battery level of the nodes (when nodes are mobile) and attempt to minimize the network's overall energy consumption during each broadcast [140, 143–146].

- **Neighbor Aware**: This type of solution leverages neighborhood information to base its decisions. The acquired neighborhoods can either be *global*, i.e., the complete network topology [140, 143, 144], or *local*, only the neighborhood of the local node and optionally of its neighbors [95, 96, 128, 145, 147–149]. Several neighbor-aware algorithms combine neighborhood knowledge with copy-aware strategies, being, in fact, hybrid solutions. Jitters may be computed from neighborhoods, giving priority to nodes with more neighbors [15, 128]. Regarding the retransmission decision, the solutions can be separated into:
  - **Self Decision**: Each node decides by itself whether to retransmit or not each message by estimating if its neighbors were already (or will be) covered by other retransmissions [95, 128, 133, 142, 148].
  - **Delegated Decision**: Each node selects a sub-set of its neighbor to retransmit each message next. There are many names given to these sets: *forwarding neighbors*[150], *forward nodes* [127, 149, 151], *forward lists* [148], *forward sets* [152], *cluster heads* and *gateways* [107], *multi-point relays (MPRs)* [89, 147, 153–155], or *broadcast relay gateways (BRGs)* [96]. For simplicity, we name them as *delegated neighbors*. The selection of delegated neighbors is where these solutions differ and can either be *static*, if they are independent of the messages received; or *dynamic* (copy-aware hybrid), if they are computed on a per-message basis.
  - **Hybrid Decision**: The retransmission decision is a combination of the two previous techniques, with nodes selecting some of their neighbors to retransmit next while allowing other neighbors to decide by themselves [27].

2.3.2.2 Examples

Most recent studies on broadcast focus on specific use cases [156]. However, since we intend our framework to be able to capture a broad spectrum of solutions, we mainly focus on classic solutions for generic wireless *ad hoc* networks. We highlight the following broadcast solutions:

Flooding In *Flooding* each node always retransmits each new message received [107, 108]. To mitigate broadcast storms, uniformely distributed jitters can be employed. Although being a simple and robust solution, it incurs many redundant retransmissions and may lead to broadcast storms despite the use of jitters. Flooding is the most simple broadcast algorithm and its name is often used to refer to the broadcast primitive [133, 147, 157].

Counter-Based Scheme (CBS) The usage of random jitters to mitigate collisions enables nodes to receive multiple copies of each message before retransmitting. As such, in CBS [107], the nodes abstain from retransmitting in case they received a number of copies above a given threshold since the EAC they would provide is negligible.

Gossip *Gossip* is a family of probabilistic algorithms [130]. In *Gossip1* each node retransmits with a given probability after a uniformely distributed jitter. Gossip1 suffers from bimodal behavior where sometimes only a few nodes near the source deliver the messages. To overcome this, *Gossip1 Hops* has nodes near the source to always retransmit. *Gossip1 Exp* is identical to Gossip1 except it employs a truncated-exponential distributed jitter [158]. *Gossip2* extends Gossip1 by having each node located in regions with low node density (using neighborhood information) retransmit with a higher probability. *Gossip3* extends Gossip1 Hops with an additional phase for the nodes that decided not to retransmit on the first phase. At the end of the second phase, the nodes execute CBS. The second phase enables nodes in sparse regions of the network to retransmit (as in Gossip2), using the number of copies received to estimate the number of neighbors.

Dynamically Adjusted Probabilistic Flooding (DAPF) *DAPF* [133] is a probabilistc copy-aware algorithm composed of four phases. At each phase, the nodes that did not retranmsit the current message yet execute CBS. If the result of CBS is to retransmit, each node retransmits with a probability given by $1/((n+1)-(in/3))$, where i is the current phase and n is the first parent's number of neighbors.

Hop Count-Aided Broadcasting (HCAB) *HCAB* [15, 138] is a hop-count-aware algorithm where each node retransmits, after waiting for the jitter, if no copy was received with a higher number of hops travelled than the first. The intuition is that if this condition is not met, then that message was already propagated through multiple directions, and thus the current node's retransmission has a higher chance of being redundant.

Hop Count Aware RAD (HCA-RAD) Extension The HCA-RAD Extension [125] improves CBS by i) increasing the number of copies each node may receive while ii) attempting to propagate messages through short paths. As such, i) allows more nodes to cancel their retransmissions and ii) requires fewer retransmissions overall to propagate messages throughout the network. For each new message received, each node waits for a jitter equal to the sum of t, a random value between 0 and ΔT, and the result of ΔT minus its first parent's t. For each copy received, $2\Delta T$ is added to the remaining jitter. When the jitter expires, each node executes CBS.

Distance-Based Scheme (DBS) *DBS* [107] is a distance-aware algorithm where each node retransmits if the distance between it and its first parent is above a threshold.

Distance-Based Adaptive Scheme (DibA) *DibA* [139] is a distance-aware algorithm that dynamically adapts to the network's local density using the number of copies received to estimate the number of neighbors. Upon receiving a new message, each node

waits for a random jitter. Whenever a new copy is received, each node increases a dynamic distance threshold D, individual to each message. After the jitter times-out, if the minimum distance to the parents is lower than D, the node does not retransmit.

Location-Based Scheme (LBS) *LBS* [107] is a location-aware algorithm where each node retransmits each message if the EAC provided by its retransmition, which is computed resorting to the coordinates of the nodes, is above a threshold.

Power-Aware Message Propagation Algorithm (PAMPA) PAMPA [15, 123, 134, 135] is a family of copy-count-based distance-aware algorithms. In the original PAMPA, after receiving a new message, each node waits for a jitter proportional to the distance to its parent (estimated trough the RSSI) and then executes CBS. In this way, nodes farther away from their (first) parents have higher retransmission priority, enabling closer nodes, that provide low EAC, to receive more copies. PAMPA-ATH/CP addresses PAMPA's overcancelation in heterogeneous topologies by only counting copies that either were received from nodes that share the first parent or with a higher RSSI than the first copy.

Flow-Aware Broadcasting Algorithm (FABA) *FABA* [129] is a PAMPA variant that attempts to mitigate even further overcancelations of PAMPA-ATH/CP in heterogeneous topologies. Whenever a new message is received, each node executes the original PAMPA. Upon receiving a copy, a new jitter t is computed (as in PAMPA) and, if t is higher than the current jitter, t replaces it. As such, the remaining jitter is increased, allowing nodes to potentially receive more copies. When the jitter expires, the nodes execute CBS. The nodes that did not retransmit enter a second phase, with the same duration as the last jitter. When this second phase ends, the nodes decide to retransmit as in PAMPA/ATH-CP.

Scalable Broadcast Algorithm (SBA) *SBA* [128] is a self-decision neighbor-aware algorithm that resorts to two-hop neighborhoods. SBA employs a random jitter that confers retransmission priority to nodes with more neighbors than their neighbors. When the jitter expires, each node does not retransmit if all its neighbors were covered by previous retransmissions. Since the more copies are received, the more likely this condition is met, SBA's jitter computation attempts to increase the number of nodes that receive many copies before deciding to retransmit. *Flooding With Self-Pruning* [148] is very similar to SBA but only resorts to one-hop neighborhoods and only considers the first copy received.

Lightweight and Efficient Network-Wide Broadcast (LENWB) LENWB [95] is a self-decision neighbor-aware algorithm that extends SBA by further verifying if all the neighbors not yet covered by previous retransmissions will be covered in the future. For this, LENWB establishes retransmission priorities that correspond to each node's number of neighbors, with node IDs used to break ties. As such, LENWB requires every node to be

aware of its neighbors, its neighbors' neighbors, and its two-hop neighbors' number of neighbors. LENWB uses a uniformly distributed random jitter instead of SBA's jitter.

Cluster-Based Scheme (CLS) *CLS* [107] is a neighbor-aware algorithm that distributively constructs groups of nodes, called *clusters*, where each node takes one of the roles: *head*, *gateway*, or *normal*. All the neighbors of a cluster head belong to a cluster defined by it. Nodes which belong to more than one cluster are considered gateways. Only cluster heads and gateways retransmit each new message received.

Multi-Point Relaying (MPR) *MPR* [89, 147] is a delegated-decision neighbor-aware algorithm that leverages two-hop neighborhoods. In this algorithm, the delegated neighbors are called *Multi-Point Relays (MPRs)* and are static. Each node computes its MPRs by selecting a subset of its neighbors whose retransmissions cover all its two-hop neighbors. Upon the reception of a new message, if the current node belongs to the MPRs of the message's parent, it retransmits the message. The lower the cardinality of MPRs, the lower the cost. However, computing the minimum MPRs is complex, especially if nodes have many neighbors, and thus approximations may be used.

Ad Hoc Broadcast Protocol (AHBP) *AHBP* [96] is a delegated-decision neighbor-aware algorithm where the delegated neighbors, which are called *Broadcast Relay Gateways (BRGs)*, are dynamically computed for each message leveraging two-hop neighborhoods and the routes traveled by the messages. BGRs are computed similarly to MPRs but removing from the known topology the nodes contained in the message's route and their neighbors, since they were already covered. Whenever a node receives a new message, if it was selected as a BGR by its first parent, it computes its set of BGRs and retransmits it. To address mobility, AHBP-EX also retransmits when the link with the parent is not known or it is not bidirectional. *Dominant Pruning* [148] resembles AHBP; however it only considers the parent and its neighbors instead of the routes.

2.3.2.3 Frameworks

Broadcast solutions share certain architectural patterns, which led to the origin of some frameworks to specify them. Next, we discuss the ones we found:

Generic Distributed Broadcast Scheme (GDBS) The authors of [26, 27] proposed a generic framework capable of specifying hybrid copy and neighbor-aware broadcast algorithms. This framework is composed by five parameters: k, h, *coverage condition*, *timing*, *priority function*, and *selection*. The k corresponds to the number of hops neighborhoods are obtained, and h corresponds to the maximum number of hops of the routes piggybacked in each message. The coverage condition is a predicate that dictates whether all the node's neighbors were already (or will be) covered by other retransmissions. This predicate leverages each node's local neighborhood, the traveled routes, and the delegated

neighbors (optionally piggybacked in each message). The timing parameter determines when to evaluate the coverage condition. It can be immediately evaluating after the reception of a new message or after some jitter. The priority function is used to predict future retransmissions of nodes and may be used by some coverage conditions. In case the coverage condition is *true*, the node cancels its retransmission. If the coverage condition is *false*, the selection parameter is executed to optionally select delegated neighbors, which are piggybacked in the message. Nonetheless, each node is allowed to not oblige to this delegation if it determines that its retransmission would be redundant.

Broadcasting from Static to Mobile networks (BSM) BSM [28] is a neighbor discovery and broadcast framework for VANETs and has six parameters: *beacon content*, *beacon frequency*, *CDS*, *timer*, *ack timer*, and *T*. The beacon content determines the information carried in hellos. The frequency of beacons specifies the hello periodicity. The CDS parameter locally determines if the local node belongs or not to a specific CDS of the network, which is performed after the reception of each hello. The timer computes jitters to employ before retransmitting and can be based on nodes belonging to the CDS, the number of neighbors that did not yet receive the message, or the distance to the centroid (i.e., mean position of all neighbors and itself). If neighbors are updated, the ongoing timers may also be updated. The ack timer defines the period that nodes should wait for ACKs. If some ACKs from the neighbors are not received within this period, the local node assumes that those nodes did not receive the message. The T parameter specifies the time each message should be cached. For each message, each node determines the neighbors already covered based on the copies received. This set is maintained for a period of T. This set may change with the addition and removal of neighbors and the reception or expiration of ACKs. After the timer expires, the node cancels its retransmission if all of its neighbors were already covered. If this set is later modified and becomes non-empty, the current node starts a new timer, after which it may retransmit again.

2.3.3 Discussion

In this section, we discussed broadcast in wireless *ad hoc* networks, highlighting the challenges and some of the most relevant solutions and frameworks. Additionally, we proposed a classification scheme for broadcast solutions based on the nature of the information they leverage to base their decisions.

During our literature review, it became clear that broadcast solutions share a similar structural pattern. At their core, these solutions contain a *Retransmission Policy* that governs the local decision of each node regarding its need to retransmit each new message received from another node. These policies strive to avoid redundant retransmissions, possibly attempt to propagate messages through the shortest paths, and/or ensure that each message is retransmitted if there are neighbors that have not received it yet. Examples of these policies are retransmit: always, with a given probability, if the number of

copies is low, if the distance to the parent is low, if all the neighbors were not yet covered, or if the current node is a delegated neighbor. We underline that some of the retransmission policies require the nodes to gather specific information (even if this information is imprecise) on their execution environment. In particular: *distance-aware* solutions need the relative distance between the local node and its parents; *location-aware* solutions require the coordinates of the nodes; *hop-count-aware* solutions require the number of hops traveled by each copy received; and *neighbor-aware* solutions require knowing the neighborhood relations with other nodes. We note that, contrary to counting message copies which can be easily handled at the broadcast protocol level, obtaining such information requires external support, e.g., an external configuration provided by users, support from the device's hardware (e.g., to extract the RSSI or coordinates), or a companion protocol being executed alongside the broadcast protocol. To these external sources of information, we called the *Retransmission Context*. As pointed out earlier, many solutions employ jitters to avoid simultaneous retransmissions of nearby nodes. In addition, some retransmission policies require a period to gather information before making their decision. Furthermore, some solutions dynamically update the currently set delay of each message after the reception of a new copy to postpone the evaluation of the retransmission policy so that the nodes are able to gather more information before making a decision. The previous observations imply that computing such jitters require sophisticated strategies, which might depend on the retransmission context, that assign retransmission priorities to different nodes, allowing nodes with lower priority to collect more information while mitigating collisions and minimizing the latency. As such, another inherent part of broadcast algorithms is a function that defines when the retransmission policy is evaluated for each message, which we called the *Retransmisson Delay*. Examples of these functions may be a delay: uniformly distributed, truncated-exponentially distributed, inversely proportional to the distance to the parent, or based on the number of neighbors. Finally, some solutions allow each node to latter reassess its retransmission decisions for each message, reevaluating the retransmission policy again to ensure that nodes that are crucial for propagating the message to other nodes retransmit. To the number of times the retransmission policy is evaluated, we called the *Retransmission Phases* of the algorithm, being that each phase is intercalated with retransmission delays. In the light of the aforementioned, the distinguishing behavior of specific broadcast solutions can be captured by specifying these four components.

Although some frameworks were already proposed, they fail to address all of the four components we identified, and hence we can conclude they lose expressiveness. The GDBS does not address solutions non-neighbor-aware, with dynamic timers, or with multiple phases. However, it identifies four broadcast contexts: the neighborhoods, the selection of delegated neighbors, node priorities, and the traveled routes. The GDBS's coverage condition is a family of broadcast policies, yet it is not flexible enough to specify, e.g., gossip and hop-count aware solutions. The GDBS's timing parameter matches with the broadcast delay; however, it does not address dynamic timers. The BSM framework

has the same limitations as GDBS. However, it identifies two broadcast contexts: the discovery protocol (with the beacon content and frequency); and the CDS, which determines if each node belongs to a specific network's CDS. BSM's broadcast policy retransmits if the node belongs to the CDS. The timer corresponds to the broadcast delay; however, it also does not support dynamic timers. We consider explicit ACKs and caching messages as orthogonal strategies to increase the reliability instead of being intrinsic parts of the solutions. Thus, the ack timer and T are irrelevant to our case.

To finish, we stress the lack of a framework in the literature that is flexible enough to specify the most diverse types of broadcast solutions through the four fundamental components that we identified.

2.4 Routing

The most notable communication primitive in a network is the *unicast* of a message, i.e., deliver a message, sent by a node called the *source*, to a single *destination*. In the context of multi-hop wireless *ad hoc* networks, if the destination is within the source's transmission range, messages can be directly sent with one-hop unicast. However, when this is not the case, the source has to entrust to other nodes the forwarding of the message towards the destination on its behalf. Similarly to one-hop unicast (§2.1.2), the simplest way to perform the (network-wide) unicast of a message is by broadcasting it, with all the other nodes which are not the destination ignoring it [22]. Unfortunately, this is a very wasteful solution since it occupies the medium with many useless retransmissions. Therefore, unicast primitives endeavor to forward messages only through a single sequence of interconnected nodes, called a *route*, towards the destination. The process of automatically discovering these routes is called (dynamic) *Routing* [22].

Each node that belongs to a given route must forward messages only to one of its neighbors that belongs to that route, called the *next-hop* (of that route). As such, *neighbor discovery* is essential for computing routes as it provides each node with properties of the wireless links with the other nodes which can be directly reachable by itself. One of such properties is the bidirectionality of communication [87, 89], i.e., both nodes can send and receive messages from each other, as it is often crucial to ensure two-way communication.

In general, there are usually multiple available routes from a given source to a given destination. In this sense, routing solutions typically strive to discover just the best routes to use by employing *cost* metrics to distinguish the routes. A route's cost is a function of the route's constituent links costs, usually the sum [87, 89] although other functions may be employed, such as the average of the constituent links' costs [159]. We distinguish between link quality and link cost since these two concepts have different, yet sometimes overlapping, semantics: the first expresses how reliable a link is to transmit information while the latter expresses how expensive it is to use that link to send information. In fact, link costs can be based on link qualities or other attributes (cost metrics will be discussed in §2.4.1). We note that, however, analogously to link qualities, link costs

are also directional, i.e., may present different values in each direction, to reflect the dissymmetry of sending messages in each direction [89].

Routes are computed in a distributed fashion trough the cooperation of the nodes, being that each node does not have to be aware of complete routes, only their next-hops. The routes known by each node are typically encoded in a (local) conceptual data structure called the *routing table*, which associates to each reachable destination the corresponding next-hop(s) [89, 160]. In addition, these structures may contain additional information in particular routing solutions to support the forwarding of messages. Nonetheless, there are few solutions that do not resort to these structures at all [93].

Unfortunately, due to the high dynamism of wireless *ad hoc* networks' topology, discovering routes and maintaining them accurate incurs many challenges.

2.4.1 Challenges

In this section, we discuss the challenges that routing solutions for wireless *ad hoc* networks must address. We highlight the following:

Route Discovery Discovering the best routes to a given destination requires simultaneously exploring several alternatives through the distributed cooperation of the nodes. Hence, it is quite expensive to perform. As such, route computation strategies strive to minimize the *routing overhead* [30], defined as the number of control messages transmitted to support route discovery, as it is essential to mitigate contention, congestion, and collisions. Furthermore, they also strive to minimize the *route discovery latency* [102], defined as the time elapsed to establish a route, as it is fundamental to maintain the routes accurate and available. Nonetheless, there is a trade-off between these two objectives, being that to achieve low latency, it is often required to incur high overhead. Since route discovery heavily relies on neighbor discovery and broadcast, adopting efficient broadcast and discovery strategies is fundamental to routing. For this reason, some protocols "merge" neighbor discovery with route discovery to reduce the overhead [94, 98, 100].

Bidirectional Routes The vast majority of routing solutions only consider bidirectional links to establish routes [87, 89, 94, 98, 160]. This approach has the advantage of routes allowing to forward traffic in both directions. However, ignoring unidirectional links may lead to "longer" routes than they could have been. As such, some improvements were proposed to consider link direction [161]. Another alternative is to establish tunnels (i.e., virtual links) over routes in the reverse direction of unidirectional links [162–164], which enables existing solutions to operate without modification. Nonetheless, leveraging unidirectional links may increase the overhead since it requires the discovery of two routes (one in each direction) if the source and destination intend bidirectional communication.

Quality of Service (QoS) QoS is a measurement of the overall performance and is intrinsically related to the routes' characteristics. The simplest cost metric corresponds to

the route's number of hops, which results in the shortest routes having the lowest number of retransmissions required to reach the destination. However, not always the shortest routes in the number of hops are the "best" to forward traffic due to the intermediary links' conditions. As such, it might be more suitable to select routes using more relevant metrics that ensure some QoS guarantees. These metrics may be computed from link qualities or other attributes. Examples of these metrics are: the link's expected number of transmissions to deliver a message (ETX) [25, 165], the link's expected transmission time (ETT) [166], the link's age [61, 159, 167], the link's stability [98, 159, 168], the congestion of the nodes [169], or the energy spent using the link [170]. Typically, the best routes are usually the ones with the lowest cost. However, some of these metrics (e.g., link stability) entail that the best routes are the ones with the highest cost. Multipath routing can further be used to ensure QoS guarantees by multiplexing the traffic through multiple routes simultaneously to: increase the aggregate throughput, mitigate congestion, increase the end-to-end reliability, or decrease the end-to-end latency [171].

Stale Routes Topology changes cause the routes to become outdated, or *stale* [160]. Stale routes can either be no longer the best —*sub-optimal routes* —, incuring inefficient propagations, or no longer be connected —*broken routes* —, resulting in non-useful transmissions and unsuccessful deliveries. Therefore, routing solutions strive to mitigate the occurrence and duration of stale routes. Since topology changes and delivery failures are frequent in wireless *ad hoc* networks, stale routes are natural, and the network might even never converge (i.e., all the nodes having the same view of the routes) [171]. As a result, some solutions employ periodic propagations of control messages to overcome these challenges at the expense of incurring high overhead [89]. Due to this high overhead, some other solutions disregard sub-optimal routes and only discover a new after they break [94, 102]. Another approach is to rely on multipath routing, immediately switching routes after a break without performing additional route discoveries [171]. *Routing loops* are a particular case of broken routes where messages are forwarded towards the same intermediary node more than once [87, 160, 172], incurring many wasteful transmissions. These loops may be short-lived, long-lived, or even persistent. Some solutions guarantee loop freedom [87, 94, 160, 173]. In case this is not guaranteed, decreasing the time routes are stale mitigates the occurrence of loops. Nonetheless, it should be appended to each message a TTL field for them to be eventually discarded.

Route Flapping *Route flapping* corresponds to nodes alternating their route to a given destination between different alternatives within a short period [160], increasing the overhead, the processing load on the nodes, and delivers messages out-of-order. Route flapping may be caused by *i*) unstable links or *ii*) the way control information is propagated. In the case of *i*), the routes that contain those links keep being frequently created and destroyed. Detecting and mitigating *i*) is the responsibility of the discovery protocol. In the case of *ii*), the propagation of control information leads to the discovery of shortest routes

that will shortly after be replaced with better routes after the arrival of new information (e.g., the control information encoding a better route may arrive later). To mitigate this phenomenon, nodes may delay the routing table's update to enable the reception of information from several paths and update only once —*damping* [160]. However, damping increases the route discovery latency.

Transmission Failures During message forwarding, transmissions might fail to be received by the next-hop due to the wireless medium's unreliability or due to congested nodes dropping messages. Typically, these failures are not a concern of routing and are dealt with by upper-layer protocols (e.g., TCP [174]). However, in wireless *ad hoc* networks, the "effort" employed in forwarding a given message is non-negligible due to the wireless medium's limitations. Moreover, employing end-to-end retransmissions greatly increases the effort and cost required to forward a message, exacerbating these difficulties. Thus, in these networks, it can be relevant to overcome transmission failures on a hop-by-hop basis. One strategy is having each next-hop replying with an ACK, either explicitly or implicitly (i.e., eavesdrop on retransmissions), allied with retransmit again if no ACK is received [87, 94, 159]. However, this might cause the issues it tries to overcome. Another strategy is having several neighbors cooperate to carry-out the forwarding [175, 176].

Scalability Scalability is defined as the system's ability to continue being efficient as the size of the system grows (i.e., number of nodes increases). There are two main factors that hinder the scalability of routing solutions: *i*) the increase in overhead to discover routes and mitigate staleness and *ii*) the increase of the number of entries in routing tables and auxiliary information each node has to maintain. To address *i*), some solutions limit the scope of control information's propagation. One alternative is to only propagate control messages to nearby nodes —*limit in space* [29, 177]. Another alternative is to propagate control messages more frequently to nearby nodes than farther nodes —*limit in time* [115, 178]. To tackle *ii*), some solutions do not pre-discover routes or only pre-discover them towards nearby nodes (through limiting propagations in space). Another strategy is to attempt to summarize entries in routing tables. However, unlike in wired and conventional wireless networks where the node IDs can be easily grouped into network prefixes [179], such task is non-trivial nor always possible within wireless *ad hoc* networks due to their dynamic topologies. Despite that, some solutions still attempt to perform such aggregation [179–182]. Moreover, multipath routing aggravates *ii*) since each routing table entry may contain more than one route.

2.4.2 Routing Algorithms and Protocols

Throughout the years, countless routing algorithms and protocols have been proposed, along with endless derivations and improvements. All these solutions endeavor to address

the previous challenges under the most diverse network scenarios. Next, we discuss classification schemes for these solutions (§2.4.2.1), followed by some examples (§2.4.2.2). Finally, we finish the section with a review of some routing frameworks (§2.4.2.3).

2.4.2.1 Classification

Due to the massive amount of routing techniques and strategies, routing solutions can be quite distinct from each other. For this reason, there are several classification criteria of routing solutions. We highlight five criteria: *route provision strategy*, *route computation strategy*, *route origin*, *forwarding policy*, and *location scheme*.

The **route provision** is defined as the scheduling of route discovery to a given destination (i.e., *when* routes are computed), and each solution is classified into:

- **Proactive (or Table-Driven)**: All the nodes periodically compute routes to all destinations so that they are immediately available when needed. Unfortunately, this strategy incurs high overhead, routing tables tend to be large, and, if the traffic is low, most of the routes may not even be used. Therefore, proactive solutions tend to be not scalable. Examples of this type of solutions are [87, 89, 98, 100, 160, 176, 179, 183, 184].

- **Reactive (or On-Demand)**: Each route to a given destination is only discovered when it is required. In this way, these solutions avoid high overhead in maintaining routes that are not used and, at the same time, also decrease the amount of information that each node has to maintain. Therefore, this type of solution tends to be more scalable than proactive routing. However, when the network topology is very dynamic, routes are regularly broken, leading to many consecutive route discoveries, which tend to be more expensive than proactive discovery strategies. Furthermore, these solutions also incur a high latency before forwarding traffic while routes are being discovered. Examples of this type of solutions are [94, 102, 159, 181, 185].

- **Hybrid**: Combines the two previous approaches by proactively maintaining routes to some destinations while reactively discovering routes to the remaining destinations. Hybrid solutions inherit part of the route availability of proactive solutions as well as the scalability of reactive solutions. Examples of this type of solution are [29, 173, 177].

Routing solutions can also be classified according to their **route computation** strategy (i.e., *how* routes are computed) into:

- **Distance-Vector**: The nodes execute some variant of the distributed Bellman-Ford algorithm [22] to compute the routes. In this algorithm, the routes are implicitly constructed through the propagation of *distance-vectors*. These distance vectors correspond to a set of reachable destinations and their associated costs. Each distance

vector may comprise a single or several destinations. Each node assigns as next-hop to each destination its neighbor which has the lowest sum among all the neighbors of the advertised cost (to that destination) with the link's cost between the node and that neighbor. This cost is then advertised in the node's distance-vector. Examples of this type of solution are [87, 94, 98, 160].

- **Link-State**: All the nodes gather the complete, or a connected sub-set, of the network topology and locally compute routes to all reachable destinations through some variant of the Dijkstra's algorithm [22]. To enable global topological knowledge, each node disseminates a small sub-set of its known topology (e.g., its neighbors) throughout the network. Examples of this type of solutions are [89, 100, 178, 183].

- **Link-Reversal**: The nodes distributively construct a Directed Acyclic Graph (DAG) over the network topology for each destination, with each directed path in the DAG corresponding to a route to the destination [173, 186]. Each edge of the DAG corresponds to a bidirectional link between two nodes with an assigned logical direction ("upstream", away from the destination, or "downstream", towards the destination). Any node that intends to forward a message chooses as next-hop any of its neighbors with whom they have a downstream edge. Examples of this type of solution are [173, 177, 186].

According to the **routes' origin** (i.e., *who* determines what route should be used), routing solutions can be classified as:

- **Non-Source**: The routes are distributively stored at each intermediary node. Whenever a node receives a message to forward, it makes its independent forwarding decision by checking its local routing table to obtain the next-hop. Examples of this type of solution are [87, 89, 94, 98, 160, 178].

- **Source**: The exact route that each message must travel is specified by the source node and piggybacked on the message. As such, only the source needs to store the route towards the destination. Each intermediary node that receives these messages checks the piggybacked route to obtain the next-hop instead of consulting its local routing table. An example of this type of solution is [102].

Routing solutions may also be classified regarding their **forwarding policy** (i.e., *how* the next-hop should be selected) as:

- **Non-Opportunistic**: For each route towards a given destination, a specific neighbor is pre-selected as the next-hop (this corresponds to unipath and multipath routing). This type of routing is employed in wired networks and on the majority of solutions for wireless *ad hoc* networks. Examples of this type of solution are [89, 94, 98, 102].

- **Opportunistic (or any-path)**: For each message, multiple potential next-hops, called *candidates*, are dynamically selected. These candidates must coordinate between themselves to determine the one that will forward the message. Typically, it is assigned to each candidate a priority that encodes its suitability in forwarding the message. In this way, opportunistic routing allows to quickly mitigate transmission failures by leveraging wireless communications' broadcast nature. Examples of this type of solution are [175, 176].

Routing solutions can also be classified according to their **location scheme** (i.e., *how* destinations are located within the network) as:

- **Topological**: The nodes are located by their IDs and routes are based on the network's topology. This type of routing can be further classified according to if any route aggregation scheme is employed into:
  - **Flat**: The network topology is unstructured, and the entries of the routing tables are not summarized. Examples of flat routing protocols are [87, 89, 94, 100, 102, 160, 173].
  - **Hierarchical**: Node IDs have hierarchical relations between themselves, allowing compact routing table entries. However, maintaining a network hierarchy is quite complex and costly in wireless *ad hoc* networks due to frequent topology changes. Examples of hierarchical routing protocols are [179, 187].
- **Geographic**: The nodes are located through their coordinates in some location system. Next-hops are dynamically selected for each message based on their proximity to the destination. Examples of geographic routing are [93, 188].

2.4.2.2 Examples

The majority of current studies on routing have focused on specific use cases [189]. However, because we want our framework to be able to capture a wide range of solutions, we mainly focus on traditional solutions for generic wireless *ad hoc* networks. We highlight the following routing solutions:

Optimized Link State Routing (OLSR) OLSR [89] is a (flat) proactive link-state (non-source non-opportunistic) routing protocol that consists in an optimized variant of conventional link-state routing protocols (OSPF [190] and ISIS [191]) for MANETs. The two main OLSR's features are its overhead reduction and its topology reduction. The overhead reduction is achieved by leveraging the MPR broadcast protocol (§2.3.2.2) rather than flooding, which incurs less cost while still guaranteeing complete coverage. The topology reduction consists of nodes gathering a reduced topology graph formed only by a sub-set of all the bidirectional stable links and their costs. OLSR allows the usage of any cost metric besides the conventional hop count. With this reduced topology, each

node locally computes the routes. This computation is less computationally expensive when compared to considering the complete global topology. The reduced topology is collected by each node broadcasting only a sub-set of its neighbors, called its *MPR Selectors*. A node's MPR Selectors correspond to the sub-set of its bidirectional neighbors that selected it as one of their MPRs (previously discussed in §2.3.2.2). The MPR Selectors (as well as the MPRs) are provided to each node through the NHDP discovery protocol (§2.2.2.2). OLSR's overhead and topology reductions increase its scalability compared to conventional link-state protocols since each node has to maintain less information per destination on the network, and less overhead is generated per topological dissemination. However, to overcome inconsistent states due to transmission failures, each node periodically broadcasts control messages, increasing its overhead.

Fisheye State Routing (FSR) FSR [178, 179, 192] is a (flat) proactive link-state (non-source non-opportunistic) routing protocol that relies on neighbor exchanges to disseminate topological information instead of broadcast. Each node periodically exchanges with its neighbors its known global topology. The propagation frequency of a known link is proportional to its distance to the current node, with distant links being propagated less frequently than closer ones to reduce the size and amount of the control messages exchanged. One or more distance thresholds are used to delimit different scopes of the global network topology, a strategy known as *fisheye scope*. In this way, the several topological changes between successive updates are grouped into a single message within each scope, thereby reducing the overhead caused by these changes. Furthermore, as the forwarded messages get closer to the destination, the more accurate the routes become since those intermediate nodes receive updates from the destination more frequently.

Fuzzy Sighted Link State (FSLS) FSLS [115] is a (flat) proactive link-state (non-source non-opportunistic) routing protocol which has a similar propagation philosophy as FSR. However, FSLS relies on the broadcast of the set of bidirectional neighbors of the nodes with a limited scope at different rates, instead of neighbor exchanges to propagate topological information. The rate of each broadcast scope is inversely proportional to its distance to the node, with closer nodes receiving updates more frequently than the more distant ones. By relying on broadcast instead of neighbor exchanges, FSLS incurs less propagation latency than FSR but higher overhead.

Topology Dissemination Based on Reverse-Path Forwarding (TBRPF) TBRPF [100] is a (flat) proactive link-state (non-source non-opportunistic) routing protocol that relies on neighbor exchanges to disseminate topological information instead of broadcast to decrease the overhead of conventional link-state protocols. In TBRPF, each node periodically sends to its neighbors a sub-tree of its source tree (i.e., a tree formed by all the node's best routes), named *reported tree (RT)*, which consists of all the bidirectional links

containing at least one node in the *reported neighbors set (RN)*. The *RN* is iteratively computed starting with the set of all the bidirectional neighbors u to which the current node is the next-hop of another neighbor in its shortest route towards u (i.e., equivalent to the MPR Selectors of OLSR). Next, every reachable node r is included in *RN* if the next-hop in the shortest route towards r is in *RN*. Finally, the current node is added to *RT*. Each node may report additional links in its *RT* to enable the discovery of additional routes. Furthermore, upon detecting a topological change, to mitigate staleness, each node immediately sends to the neighbors a control message containing only the modifications which the change caused in *RT*. The source tree of each node is then computed (with the Dijkstra's algorithm) from the graph resultant from the union of the neighbors' *RTs* and the links with them. Next, each node then computes its *RT* to announce in the subsequent scheduled transmission.

Babel Babel [87] is a (flat) proactive distance-vector (non-source non-opportunistic) routing protocol that guarantees loop-freedom. To guarantee loop-freedom, Babel filters route updates contained in the neighbors' distance-vectors through *feasibility conditions*, which only accept updates that are guaranteed not to cause routing loops. These conditions assert that a given update, received from some neighbor and encoding a route to some destination, is either more recent (i.e., higher sequence number) or has a lower cost than all the updates advertised by the local node to that destination. Updates with an infinite cost (i.e., unreachable destinations) are also not feasible. From the feasible updates to a specific destination, each node then selects its the next-hop. However, this selection must not consider sequence numbers to not cause flapping routes. In case a node does not have any feasible route towards some destination, it incurs *starvation*. To mitigate starvation, each node that detects it is in starvation sends a request to the destination to increase its sequence number. The new sequence number leads to the acceptance of subsequent updates since they will be fresher. These requests are the only situation that increases sequence number. Whenever a node detects a topology change (through the discovery protocol §2.2.2.2), it immediately advertises its updated distance-vector to decrease the convergence time. However, although this convergence is guaranteed to be loop-free, it does not necessarily discover optimal-routes. Nonetheless, through distance-vectors' periodic propagations, nodes eventually return to an optimal-state even though they may incur temporary starvation.

Better Approach To Mobile Ad hoc Networking (BATMAN) BATMAN [98] is a (flat) proactive distance-vector (non-source non-opportunistic) routing protocol where routes are selected based on their stability. A route's stability is estimated with the number of control messages sent by the route's destination, called *originator messages (OGMs)*, that were received through that route within a time window. In this way, the neighbor which forwarded more OGMs of a given destination is considered the more stable next-hop towards that destination. A route's stability is not explicitly included in the OGMs

themselves like conventional distance-vector costs and instead is implicit in the OGMs' receptions. In this sense, BATMAN can be considered an implicit distance-vector protocol. Each node periodically broadcasts new OGMs to enable all the other nodes to discover a route towards the broadcast source and estimate its stability. The dissemination of OGMs simultaneously performs neighbor discovery with bidirectionality detection by forcing all the broadcast source's neighbors to always retransmit. Subsequent retransmissions are only performed in case the current OGM is received from a bidirectional link and through the best next-hop towards the broadcast source. This broadcast strategy ensures that the propagation of OGMs causes the nodes to discover only the most stable bidirectional routes, through each neighbor, towards the broadcast source.

auto-adJustable Opportunistic acKnowledgment/timEr-based Routing (JOKER) JOKER [176] is a (flat) proactive distance-vector (non-source) opportunistic routing protocol derived from BATMAN. JOKER's differs from BATMAN's on two aspects: *i*) next-hop selection and *ii*) OGM broadcast interval. JOKER dynamically selects several candidates to forward each message, prioritizing the more distant neighbors, whereas BATMAN preselects only a single next-hop based only on the stability (*i*). This strategy aims to reduce the routes' number of hops and, consequently, the number of transmissions required to forward messages. The criteria for selecting candidates is to pick the n (parameter) best neighbors according to their suitability in being the next-hop, based on the stability, distance, and other metrics. These candidates are piggybacked in each message and then coordinate to decide which one will forward the message next. JOKER defines two types of candidate coordination: *a*) *ACK-based* and *b*) *Timer-based*. In *a*), each candidate replies with an ACK upon receiving a message. Next, the current node replies to the first ACK received with a notification to forward the message. All the other candidates abstain from transmitting. In *b*), upon each candidate receiving a message, they wait a period proportional to their priority with the one with the highest priority immediately forwarding. While the other candidates are waiting, they listen for the retransmission of the higher priority candidates. If no copy is detected, the candidate with the following priority forwards the message, abstaining otherwise. Additionally, JOKER adapts the OGM broadcast interval (*ii*) of each node according to the congestion the node is experiencing, being that in BATMAN, this interval is constant.

Ad hoc On-Demand Distance Vector (AODV) AODV [94] is a (flat) reactive distance-vector (non-source non-opportunistic) routing protocol that provides loop-freedom upon topology changes. To ensure this property, AODV uses sequence numbers in each route discovery to identify and invalidate stale information, a technique adapted from DSDV [160]. Whenever a node requires a new route towards a destination, it broadcasts a route request control message. Each copy of the route request accumulates the cost of each link it travels. The broadcast algorithm employed by AODV is a variant of flooding that guarantees whenever a node receives a copy of the route request from a bidirectional neighbor, that

message traveled exclusively through bidirectional links, i.e., the route traveled by the message is bidirectional (similarly to BATMAN). Each node that receives such copy inserts in its routing table a new entry towards the route request's source, with the next-hop being the bidirectional neighbor from which a copy was received with the lowest cost. When the destination receives the route request, it sends a route reply control message through its selected route (towards the route request's source). This route reply also accumulates the cost of the links it travels through, since the costs can be distinct in each direction. Upon the route reply reaching the route request's source, a bidirectional route between the two nodes is established. To detect link bidirectionality, AODV employs a hybrid neighbor discovery protocol, which "merges" certain message exchanges with the propagation of routing control messages (route requests and replies). This discovery protocol does not rely on periodic transmissions to detect the loss of neighbors. Instead, AODV employs ACKs on a hop-by-hop basis and assumes that if no ACK is received after several attempts, the link is unusable due to being either unidirectional or broken. As such, the route's next-hop is blacklisted by the current node for some time. When a route break happens, the node that detects it sends a route error control message through the remaining of the route, informing all the other intermediary nodes that the route is now invalid. If the source intends to send new messages, it performs a new route discovery.

Dynamic Source Routing (DSR) DSR [102] is a (flat) reactive distance-vector source (non-opportunistic) routing protocol. In DSR, the routing table contains full routes (i.e., the IDs of all intermediate nodes), being called *route cache*. If a node that intends to send a message to a given destination finds a route in its route cache, it piggybacks the entire route into the message and forwards it to the next-hop. All the intermediary nodes that receive the message consult the piggybacked route to obtain the next-hop. In case there is no route in the route cache, the source broadcasts a route request control message through flooding. Each route request's copy piggybacks the entire route it traveled. Upon any route request's copy being received by the destination, it reverses the piggybacked route and forwards a route reply through that reversed route (also piggybacking the route). As such, multiple routes are discovered simultaneously. DSR relies on hop-by-hop ACKs to detect link breakages, similarly to AODV. When the source receives route replies, it inserts the routes in its routing cache and starts forwarding messages through them. Additionally, DSR provides several optimizations to increase its performance.

Associativity-Based Routing (ABR) ABR [159] is a (flat) reactive distance-vector (non-source non-opportunistic) routing protocol that selects routes based on their expected longevity. The longevity of a route is estimated through its average number of stable links, being that the higher this number, the higher the chance of the route not breaking. To determine if each link is stable, each node establishes with each neighbor *associativity ticks*, that correspond to the amount of hello transmission intervals that have elapsed since they became neighbors. If the associativity ticks of a neighbor are above a certain

threshold, which depends on the nodes' average velocity and transmission power, the link is considered stable. Whenever a source node requires a route, it broadcasts a route request message through flooding. Similarly to DSR, the route request will piggyback the route it traveled through, along with the associativity of the intermediary links. Upon the destination receiving the route request, it selects the best route (i.e., the one with the highest average number of stable links), using the lowest amount of hops to break ties. Next, the destination forwards a route reply message towards the source through the selected route. Upon receiving the route reply, every node along the route inserts it into its routing table. By selecting routes based on their expected longevity, route breaks are minimized, thus decreasing the overhead. To mitigate the overhead even further, ABR employs a route reconstruction mechanism that attempts to quickly fix a broken route without broadcasting a new route request throughout the whole network. This mechanism corresponds to the node which detects the route break to broadcast a route request towards its next-hop, with a limited scope.

Zone Routing Protocol (ZRP) ZRP [29, 193] is a generic (flat) hybrid routing protocol where each node proactively computes routes to nearby nodes while reactively discovering routes to distant ones. Each node employs a proactive routing protocol only within a limited scope around itself, called its *routing zone*. These routing zones are defined as all the nodes up to a maximum number of hops, called the *zone radius*. Whenever a node intends to send messages to a destination outside its routing zone, it relies on the reactive routing protocol. However, instead of using broadcast to propagate the route requests, ZRP efficiently propagates these messages to the nodes at the border of the routing zones through a process called *bordercast*. After the destination is found, it sends a reply to the source containing the ID of all the peripheral nodes that forwarded the query. Therefore, these routes are encoded through a sequence of intermediary nodes, *zone radius* hops away from each other. Consequently, these routes are more stable than specifying all the intermediary nodes (as in DSR). Furthermore, this encoding can provide multiple routes to the destination. Since routing zones of neighboring nodes highly overlap, routing tends to perform relatively well. ZRP is a generic framework in the sense that practically any proactive protocol can be employed within the routing zones. The Independent Zone Routing (IZR) [114] framework extends ZRP to support independent zone radii for each node, allowing to decrease even more the overhead. This framework will be later discussed in §2.4.2.3.

Temporally-Ordered Routing Algorithm (TORA) TORA [173, 194] is a (flat) hybrid link-reversal (non-source non-opportunistic) routing protocol that decreases the overhead caused by topology changes by affecting only a localized small sub-set of the nodes while guaranteeing all routes remain loop-free. Each node may create its DAG proactively or create it when requested by other nodes (i.e., reactively). Unlike other reactive protocols, in TORA, the route traveled through each route request copy does not matter, and there

is no route reply sent. The purpose of the route request is to inform the destination to trigger its DAG creation. The DAG's creation is performed by flooding a control message that triggers the assignment of logical directions to the DAG's edges. The edges' directions are determined by each node's *heights*, being directed from the highest node to the lowest. These heights are totally ordered, and two distinct nodes cannot have the same height. The heights are composed of two parts: the *reference level* and an *offset* within the level. The reference level is used to define a new height's "global maximum" each time a node (other than de destination) becomes a "local minimum" (i.e., ends up with no downstream edge). This situation may occur due to losing neighbors, and this height adjustment process causes all the upstream edges (of that node) to be reversed, which may lead to other nodes also ending up with no downstream edge. These nodes, however, do not define a new level and instead define a higher sub-level within the new reference level. The offset is related to the distance in hops towards the destination and is used to assign heights during the propagation of a new reference level. This propagation behaves similarly to distance-vector protocols, accumulating the cost. The height of the destination is always zero. Each node maintains the heights of its neighbors so that they are able to determine the direction of its links. Unfortunately, TORA forces all the nodes to participate in all active routes, even those not interested in communicating with the destination. Furthermore, TORA requires a global synchronized clock (physical or logical) to establish the temporal order of topology changes, which is problematic to maintain.

Sharp Hybrid Adaptive Routing Protocol (SHARP) SHARP [177] is a (flat) hybrid (non-source non-opportunistic) routing protocol that determines the perfect equilibrium between proactive and reactive behavior for each node. This equilibrium is achieved by maintaining proactive routing zones only around nodes that are popular destinations. Within these zones, TORA is used to establish routes towards the center node. Each node has an independent zone radius, computed based on the traffic and the local network characteristics, and can specify the cost metric used to select the best routes towards itself. AODV is used by the nodes that are outside of the destination's routing zone.

Hierarchical State Routing (HSR) HSR [179, 187] is a hierarchical proactive link-state (non-source non-opportunistic) routing protocol that decouples the logical aggregation of nodes from the topological hierarchy. To each node is assigned one or more *hierarchical addresses (HIDs)* and a *logical address (LA)*. HSR induces a topological hierarchy, through recursive clustering, over the network. At each level of this hierarchy, the nodes elect cluster-heads to represent a group of nearby nodes in the next hierarchy level. Within each cluster, the nodes broadcast their (virtual) neighborhoods, as in other link-state protocols. The cluster-heads summarizes this information into virtual neighbors and broadcast it within the next higher-level cluster. The virtual links between virtual neighbors correspond to full routes in the real network between the two nodes. Therefore, all the nodes have real full routes towards their virtual neighbors. The hierarchical address

of a given node is composed of a sequence of the cluster-heads' IDs representing that node at each level, with the last being the regular ID of that node. Therefore, hierarchical addresses identify hierarchical relations among nodes based on the network's topology. With the destination's hierarchical address, the nodes forward traffic upstream in the hierarchy towards either a common cluster-head they share with the destination or to the highest cluster-head, which is a virtual neighbor of the destination's highest cluster-head. Since every node knows real full routes within the clusters they belong to, messages can be forwarded towards the destination's highest cluster-head. Then, this cluster-head forwards traffic downstream until it reaches the destination. Consequently, HSR can reduce the size of routing tables each node has to maintain. On top of this hierarchical topology, it is constructed a logical network, where LAs identify the nodes. The LAs are composed by a tuple $(subnet, host)$, where the subnet identifies the logical group the node belongs to, and the host uniquely identifies the node within the group, similarly to conventional IP addresses. These LAs are used by applications and upper-layer protocols to designate the nodes with whom they intend to communicate. Whenever a node has a message to send, it must first translate the LA into a correspondent HID. These relations are stored in a distributed location system which is consulted to perform the translation. This system is composed of at least one node per subnet, called *home agents*, which maintain the mappings between the LAs of the members of that subnet and its HIDs. These mappings are updated by the nodes whenever they change their HIDs. These home agents announce their HIDs to the cluster-heads of the highest level. When a node intends to perform this translation, it consults a home agent of the subnet to which the destination belongs. Since every node can forward messages to its highest cluster-head, it can consult the HID of the intended home agent, which, in turn, is consulted to obtain the HID of the destination. Nodes temporarily cache the destinations' HIDs for forwarding subsequent messages to mitigate the overhead incurring in address translation. Unfortunately, if nodes frequently change their HIDs, both the topological hierarchy and the logical network become infeasible to maintain. Furthermore, messages usually travel through longer routes (in hops) than the shortest route between them.

Greedy Perimeter Stateless Routing (GPSR) In GPSR, each node only has to maintain a list of neighbors and their correspondent coordinates. This knowledge is acquired through a hybrid neighbor discovery protocol combined with promiscuous mode. However, whenever a node intends to send a message, it must obtain the destination's coordinates through some external method, which incurs additional overhead and occupied storage. Regarding message forwarding, GPSR combines two strategies: *greddy forwarding* and *perimeter forwarding*. At first, greedy forwarding is employed in which each node selects as the next-hop its neighbor who is geographically closer to the destination. However, sometimes a node may be closer to the destination than any of its neighbors —*local maximum*. In these cases, it is required that the messages are temporarily forwarded geographically farther from the destination to overcome such local maximums. Therefore,

whenever a node encounters itself in this situation, GPSR switches to perimeter forwarding for that message. Perimeter forwarding consists of executing the *right-hand rule*: select as next-hop the neighbor whose link is the next, counterclockwise, to the link between the current node and its parent. Once the message is delivered to a geographically closer node to the destination than the node which induced the perimeter forwarding, GPSR switches back to greedy forwarding.

2.4.2.3 Frameworks

Here we discuss some proposed routing frameworks that capture the behavior of some routing protocols.

Independent Zone Routing (IZR) The IZR [114] framework (an improved version of ZRP [29], previously discussed) enables the hybridization of proactive and reactive protocols. In IZR, this hybridization is able to dynamically adapt the proportion between proactive and reactive behavior. This framework is composed by four elements: the *Intrazone Routing Protocol (IARP)*, the *Interzone Routing Protocol (IERP)*, the *Bordercast Resolution Protocol (BRP)*, and the *Zone Radius Determination Algorithm (ZRDA)*. The IARP is a proactive routing protocol that operates within each routing zone (for each node). To be used as IARP, existing proactive protocols must be adapted to propagate control information only to the nodes whom the current node is within their routing zone. These nodes are called the current node's sending zone. This propagation can be achieved with limited scope broadcast, for instance. The IERP is a reactive routing protocol that operates beyond each node's routing zone. To be used as IERP, existing reactive protocols must be adapted to disseminate route requests through BRP instead of broadcast. The BRP constructs a distributed bordercast tree, by leveraging the topological information acquired by IARP, to efficiently disseminate IERP's control messages (e.g., route requests). The ZRDA independently determines each node's zone radius based on local environment characteristics, such as the amount of control traffic forwarded.

Relay Node Set (RNS) Framework The authors of [30] proposed an analytical framework for comparing routing protocols' control overhead. This framework views each routing protocol as a handler of *relay node sets (RNSs)*: sets of nodes that retransmit control messages. Each protocol may manage more than one RNS at a time, e.g., each source and destination pairs having a distinct RNS. Four modules compose this framework: *i*) *RNS building*, *ii*) *RNS control message propagation*, *iii*) *RNS maintenance*, and *iv*) *unreliable control message transmission handling*. The *i*) selects which nodes will be responsible for retransmitting future control messages, i.e., constructing RNSs. In the case of AODV, an RNS is selected for each source-destination pair, and they correspond to the set of nodes who retransmitted the source's route request. In OLSR, there is a single RNS, and it is determined by the computation of MPRs (performed by the discovery protocol). The nodes

belonging to the MPRs will be the ones that will forward the routing control messages, i.e., they form the RNS. The *ii*) determines which nodes belonging to the considered RNS should forward each control message and if it is through unicast or (one-hop) broadcast. In the case of AODV, this corresponds to the nodes which forward route replies. These route replies are typically sent through unicast towards the route request's source. In OLSR, all the MPRs (i.e., RNSs) retransmit each control message containing a local view of the network topology. The *iii*) corresponds to the strategy employed to adapt the current RNSs upon topological changes. AODV does not employ any strategy to repair RNSs and instead recomputes new RNSs upon a route break. In OLSR, upon topological changes up to two hops, each node recomputes its MPRs, thus updating the RNS. The *iv*) is the mechanism used to overcome the wireless medium's unreliability on propagating control information. AODV use ACKS with multiple transmissions. OLSR relies on periodic transmissions. By individually studying the overhead incurred by each of these modules, this framework provides a method to compare each routing protocol's overall overhead.

Multi-mode Routing Protocol (MMRP) Framework The authors of [195] prosed a framework that independently selects the most suitable protocol for a given region of the network according to its local characteristics, allowing the coexistence of multiple routing protocols within the same network (called multi-mode routing). This framework is composed of three elements: the *Limited Information Dissemination (LID)* module, the *Pattern Extraction (PE)* and *Self Organization (SO)* module, and the *Multi-mode Routing Engine (MRE)*. The LID is responsible for disseminating accurate control information to nearby nodes and optionally disseminate information to farther nodes less frequently, similar to a combination of IZR's IARP and the FSR protocol. The PE identifies patterns in the node mobility and traffic, whose data is gathered by the MRE. Based on these patterns, the SO provides efficient reachability to nodes not encompassed by the LID (similar to IZR's IERP). It attempts to minimize the number of overhead required for route discovery by computing routes *a priori* for destinations, called *Reference Nodes (RNs)*, which are expected to be part of routes to other nearby nodes. These RNs are selected based on the expected number of routes whose destination is within a *Reference Area (RA)* around them, called the node's *gain*. The gains' computations are performed based on information acquired by the PE module. Based on the routes computed by the LID and SO, the MRE selects a forwarding "mode". For instance, use a known route if available; forward messages towards an RN whose RA includes the messages' destination; or employ conventional broadcast-based reactive route discovery.

2.4.3 Discussion

Throughout this section, we discussed routing in wireless *ad hoc* networks. We started by reviewing the challenges in routing and discussed how existing solutions can be classified. Additionally, we reviewed a set of representative solutions and some frameworks.

Although routing solutions employ highly heterogeneous strategies, they share some common patterns and structures. Routing and neighborhood tables are inherent to most routing solutions, despite existing some exceptions. At the core of routing solutions, we can find a *Routing Strategy* which distributively computes routes, defining specific route provision and route computation strategies. Examples of it may be: a proactive link-state strategy, a proactive distance-vector strategy, a reactive distance-vector strategy, or a hybrid link-reversal strategy. Note that these high-level strategies can be further specialized to better fit specific routing protocols. Moreover, all proactive (and hybrid) solutions periodically disseminate control messages with static or dynamically adjusted periods, which are determined by the *Announce Period*. We split the announce period and the routing strategy to allow using distinct period strategies independently from the strategy to compute routes. The announce period is simply a number that represents the interval between periodic announcements of control messages. This value can be the result of a function in the case dynamic periods are employed. In addition, routing solutions resort to a *Cost Function* to evaluate the local links, by employing some metric, and is used to qualify each route. Examples of these functions were already previously discussed (§2.4.1). Moreover, routing solutions need to propagate control messages throughout the network to enable the actual computation of routes by means of some routing strategy. Across the literature, routing protocols employ several *Dissemination Strategies*, even within the same protocol, which can be grouped into specific communication patterns according to the nature and intended destinations of such messages into: *network-wide* [89, 94, 98] or *limited-hop broadcast* [29, 87], to inform all or a sub-set of the nodes; *bordercast* [29], to inform a specific sub-set of the nodes; or (*network-wide*) *unicast* [94, 102] to inform a single node. Finally, routing protocols are also responsible for leveraging the computed routes to forward applicational messages. To this end, routing protocols can employ different *Forwarding Strategies* that provide different trade-offs across reliability and communication overhead. Examples of these strategies are: to forward to the next-hop contained in the routing table; leverage several routes to the same destination to increase the chances of delivering messages employ coordination mechanisms to dynamically elect, from a set of candidate next-hops, the one which will proceed with the forwarding of each message; use the nodes' coordinates to base their forwarding decisions; obtain the next-hop that is carried in the messages; or rely on the explicit or implicit acknowledgment and retransmission of messages. In this regard, we can specify routing solutions according to these five components.

Some routing frameworks were already proposed; however, they fail to address all the components we identified, being not expressive enough to specify all the routing examples discussed. Although IZR allows combining practically any proactive and reactive solutions, it considers them as "black boxes" and does not attempt to decompose them into their fundamental constituents, like our framework. Nonetheless, IZR only identifies two dissemination strategies: limited broadcast for IARP and bordercast for IERP. RNS dissects the routing solutions from an evaluation standpoint, not being able to specify

them according to their internal operation, as our framework does. However, the RNS building, RNS control message propagation, and the unreliable transmission handling correspond to the dissemination strategy and the RNS maintenance is part of the routing strategy. Nonetheless, the other components are not considered. MMRP is not flexible enough to specify the majority of the existing protocols since it has a specific architectural pattern, only suitable for a very restricted set of solutions, unlike our framework which is much more generic. Nonetheless, the LID and SO correspond to two different dissemination strategies, while PE can be considered part of the routing strategy. However, cost metrics and forwarding strategies are not addressed.

In conclusion, although routing solutions are highly heterogenous, they share some architectural elements. However, we emphasize the absence of a framework in the literature that considers the five components we identified.

2.5 Summary

Throughout this chapter, we reviewed the related work on communication primitives for wireless *ad hoc* networks.

We started the chapter with an overview of the fundamental concepts and characteristics of wireless *ad hoc* networks and the wireless medium, on §2.1. We discussed the different types of wireless *ad hoc* networks and decided to focus on generic wireless *ad hoc* networks because they are more suitable for commodity devices since they do not require specialized hardware nor assume specific roles and behaviors of the nodes. We also provided a brief review of existing technologies to construct wireless *ad hoc* networks and decided to rely on WiFi since it is widely available in commodity devices and has high data-rates, which are more appealing for devising distributed applications and services without major bandwidth constraints. Next, we examined the characteristics and the inherent challenges of communicating through the wireless medium, such as: fading, collisions, contention, gray zones, unidirectional links, network partitions, and network congestion. These challenges turn the networks' topologies highly dynamic, even if the nodes do not present mobility. We further discussed several MAC protocols that were proposed to mitigate some of these challenges. However, despite the techniques employed by them, those challenges are not always overcomed, and thus, they must also be addressed by the communication primitives to enable effective communications.

Following that, we studied the existing literature on neighbor discovery, on §2.2. Neighbor discovery is the process that allows each node to detect and evaluate the links between itself and the other nearby nodes with whom it can direclty communicate, i.e., its neighbors. This process is remarkably important to several applications and protocols, especially several broadcast and most routing solutions, which leverage neighborhood information to make informed decisions. We identified five fundamental components in the existing neighbor discovery solutions: the *discovery pattern*, the *discovery period*, link *quality metrics*, link *acceptance policies*, and the *discovery context*. The discovery pattern

encodes the strategy employed to exchange hellos and hacks with the neighbors. The discovery period determines the intervals between consecutive discovery messages (if the discovery is non-passive). The link quality metrics estimate the quality of links with neighbors (in the incoming direction). The link acceptance policies filter the links with neighbors based on their properties, e.g., link qualities or age.

Next, on §2.3 we put forward the related work on broadcast. Broadcast consists of propagating a message through the network such that all the nodes deliver it. All broadcast solutions must detect the reception of copies. In addition to this, we determined four constituents in the existing broadcast solutions: the *retransmission policy*, the *retransmission delay*, the *retransmission context*, and *the retransmission phases*. The retransmission policy determines if a node will retransmit or not each new message received. The retransmission delay controls when the broadcast policy should be evaluated. The retransmission context gathers information from the environment to support the broadcast policy and delay. We highlight that one most common examples of such contexts is an external neighbor discovery protocol. The retransmission phases encode how many times each node should consider retransmitting each message.

Finaly, on §2.4 we reviewed the literature on routing. Routing is defined as the process to automatically establish distributed routes across the network between two nodes to perform the unicast of messages. The majority of routing solutions have a routing table and a neighbors table[8], which is obtained from an external discovery protocol. We highlight five elements forming the existing routing solutions: the *dissemination strategy*, the *routing strategy*, the *announce period*, the *cost metric*, and the *forwarding strategy*. The dissemination strategy is in charge of delivering control messages to other nodes. The routing strategy is responsible for computing routes from the control information gathered. The announce period stipulates when each node should disseminate a new control message in proactive and hybrid solutions. The cost metric is a function that computes the costs of each link with the neighbors, in both directions, through the link's properties. The forwarding strategy defines for each (non-control) message how to determine the next-hop and how the message should be delivered to it.

During these literature reviews, it became apparent the lack of frameworks expressive enough to specify the vast panoply of existing solutions by addressing the underlying common components of these solutions that we identified. On the basis of these components, in the next chapter, we will present the conceptual frameworks we developed to specify and capture the inherent similarities and differences among the existing solutions proposed of these three key services that support the communication primitives for wireless *ad hoc* networks.

[8] In our case, the routing and neighbor discovery frameworks do not share memory and only communicate trough the exchange of messages between the two protocols. As such, the routing framework has to maintain its own copy of the neighbors table.

Chapter 3

Frameworks for Communication Primitives

This chapter unveils the main contributions of this thesis: three conceptual frameworks that capture the behavior of neighbor discovery (§3.1), broadcast (§3.2), and routing (§3.3) solutions. The design of our frameworks is derived directly from the observations presented previously on §2. The central insight for devising these frameworks is that most instances of the three services that support the communication primitives for wireless *ad hoc* networks present similar architectural patterns. Each framework corresponds to a generic parametrizable protocol of each service for wireless *ad hoc* networks. In the following, we discuss each framework individually.

3.1 Neighbor Discovery Framework

This section presents our conceptual framework for specifying neighbor discovery solutions for wireless *ad hoc* networks. This framework abstracts the shared components of neighbor discovery solutions, such as managing a neighbors table, processing hellos and hacks, detecting losses of hellos and neighbors, detecting bidirectionality, serializing and deserializing messages, and notifying other protocols and applications regarding changes on the observed neighborhood. For each new neighbor found, the framework manages both its status and link properties by processing hellos and hacks from that neighbor. At the same time, this framework offers a set of parameters that control particular aspects of its operation that enable capturing the behavior of particular neighbor discovery solutions. These parameters correspond to the five fundamental components of neighbor discovery solutions previously identified in §2.2.3: the *Transmission Periods*; the *Discovery Pattern*; the *Discovery Context*; the *Link Quality* metric; and the *Link Admission* policy.

Next, we discuss the auxiliary conceptual data structures used by our framework (§3.1.1), followed by an overview of the operation of the framework (§3.1.2), and finishing with a set of examples of values for the parameters of the framework as well as some solutions (§3.1.3).

3.1.1 Conceptual Data Structures

Here we discuss the conceptual data structures that are part of our framework.

Neighbors Table Every discovery solution contains a table where the discovered neighbors and the properties of their links with the local node are recorded. Each neighbor has a single entry in this table which is updated by processing hellos and hacks from that neighbor. Each entry corresponds to a tuple $(ID, MAC, SEQ, T, A, P_{hello}, P_{hacks}, RX\_LQ, TX\_LQ, RX\_EXP, TX\_EXP, FOUND, LOST, NEIGHS)$, where: ID is the ID of the neighbor; MAC is the MAC address of the neighbor, to enable sending unicast messages without the need to execute ARP (Address Resolution Protocol); SEQ is the sequence number carried in the last hello received from the neighbor to enable detecting losses; T is the status of the node, which can be U for unidirectional, B for bidirectional, or L for lost (lost neighbors are kept in the table for a period upon being lost to enable informing the neighbors of the loss [86]); A encodes if the neighbor is accepted or not according to a *link acceptance policy* (later discussed); P_{hello} and P_{hacks} are the periods of the neighbor for transmitting hellos and hacks, respectively, and are used to set expiration of the link and bidirectionality, respectively; RX_LQ is the link quality metric measured for the neighbor, based on the reception and loss of hellos; TX_LQ is the link quality metric that the neighbor estimated for the local node and which is advertised in each hack (sent by the neighbor for the local node); RX_EXP and TX_EXP are timestamps for the expiration of the link and its bidirectionality, respectively; $FOUND$ is a timestamp of the moment the neighbor was found; $LOST$ is a timestamp of the moment the neighbor was considered lost; and $NEIGHS$ is the set of neighbors of the node (i.e., the two-hop neighbors of the local node), which can be constructed by eavesdropping on the hacks meant for other nodes. Each entry of this table has an associated timer with a value equal to the minimum between RX_EXP and TX_EXP, which triggers an update of the entry upon expiring, changing the status of the node (T).

Hello Each hello is a tuple (n, m, s, P_{hello}), where: n is the ID of the local node; m is the MAC address of the local node; s is a (local) monotonically increasing sequence number that is increased for each hello sent by the local node; and P_{hello} is the period of the local node for transmitting hellos (in case of a passive approach this value must also be set so that the neighbors can set their expiration timestamps). Additionally, each hello can carry additional information, such as the coordinates of the local node or its willingness to forward messages on behalf of other nodes.

Hack Each hack is a tuple (n, d, s, rx, P_{hack}), where: n is the ID of the local node; d is the ID of the neighbor for which the hack is destined; s is the sequence number carried in the last hello received from the neighbor; rx is the RX_LQ value registered in the neighbors table for the neighbor d so that each node becomes aware of the link qualities in each direction; and P_{hack} is the period of the local node for transmitting hacks (in case of a passive approach this value must also be set so that the neighbors can set their expiration timestamps). In addition, since hacks can also be used to obtain two-hop neighborhoods, each hack can be extended with tx and t, the TX_LQ and T values registered in the neighbors table for the neighbor d, respectively, so that the neighbors can know link qualities in both directions and the status of each link of the two-hop neighbors. Furthermore, an extended hack can also have the value L for the field t, which serves as a *negative hello acknowledgment (NHACK)* to inform the neighbors that the node identified by d is no longer a neighbor of the local node [86].

3.1.2 Operation

Figure 3.1 depicts a simplified flux diagram that captures the workflow of a generic wireless *ad hoc* discovery protocol that serves as the foundation of our framework. The diagram describes how the framework processes each event and how its components are interconnected. There are five ways to initiate the execution flow (represented has black circles in Figure 3.1): i) when the local node request the transmission of a given message unrelated to the discovery framework (Downstream Message); ii) when a timer encoding the transmission of a periodic hello and/or hacks expires (Hello/Hack Timer); iii) upon the reception of a hello from a neighbor; iv) upon the reception of a hack from a neighbor; or v) when a timer associated with an entry of the neighbors table expires (Neighbor(n) Timer).

In the case of (i), another protocol or application requests the transmission of a given message. In the case of (ii), a timer encoding the periodic transmission of a hello or hacks expires. Both events are captured by the discovery framework, which informs the *Discovery Pattern*. This component is responsible for deciding to transmit 0 or 1 hellos and 0 or k hacks based on the events observed or simply ignore the events, in which case the flow finishes. Then, the framework creates a new message containing the amount of hello and hacks as specified by the *Discovery Pattern*, and whose contents are retrieved from the *Neighbors Table*. Next, if a hello is present in this new message, the sequence number of the local node SEQ is incremented. Afterward, the framework resorts to the *Transmission Period* to (re)schedule the hello and hack timers in periodic and hybrid solutions. Then, the framework executes the *Discovery Context* to optionally append additional information in the hellos and hacks if intended. This feature is particularly relevant in many solutions to acquire the coordinates of the neighbors or other useful information for other protocols and applications. Finally, the message containing the ID of the local node p and the hellos and hacks is either transmitted (in the case of

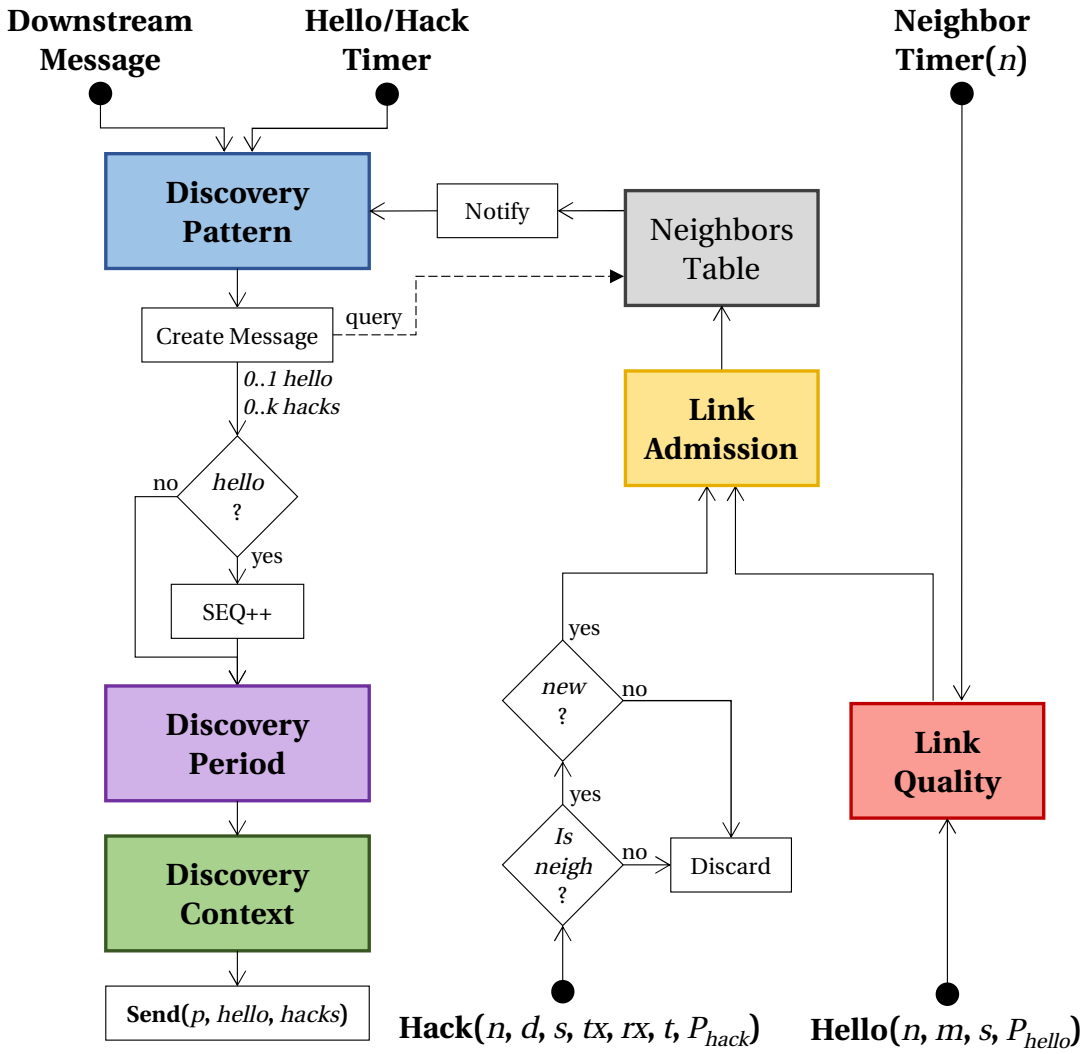


Figure 3.1: Discovery Framework Execution Flow.

(*ii*)) or piggybacked to the outgoing message (in the case of (*i*)) being both sent with a single transmission. If the message only contains a single hack, this transmission can be performed with (one-hop) unicast. Otherwise, the transmission must be performed with one-hop broadcast so that all neighbors can receive the hello and/or the hacks destined to them.

In the case of (*iii*), the framework receives a hello from a neighboring node n with a sequence number s. If this is the first hello received from the node n, then n is a new neighbor, and its link is considered unidirectional. Then, the framework executes the *Link Quality* metric to obtain a new value for the *RX_LQ* field of n's entry in the *Neighbors Table*. Next, the *Link Acceptance* policy is executed, which determines the acceptance state A of the neighbor n based on its attributes, such as link qualities, age, or bidirectionality. Rejected neighbors are not considered when sending hacks and are not reported to other applications and protocols, i.e., they are simply ignored. However, they are still kept in the *Neighbors Table* until they are either eventually flushed or accepted by the policy. Next,

the n's entry in the *Neighbors Table* is updated with the new information and the RX_EXP field is set according to P_{hello}. Then, the framework may notify the other protocols and applications of a change in the neighborhood state and informs the *Discovery Pattern*, which may continue the execution flow as previously described or simply terminate it.

In the case of (iv), the framework receives a hack from a node n destined to the node d. If d is the local node and this is the first hack received for the neighbor n, then the link with n becomes bidirectional. The diagram of the Figure 3.1 is only illustrating the reception of a single hack. Whenever multiple hacks are received at the same time, the framework processes each one individually in sequential order. Whenever hellos and hacks are received in the same message, hellos are always processed first. In case the node n is not a neighbor of the local node (i.e., is not registered in the *Neighbors Table*) or the hack is not new (verified by comparing its sequence number s with the local node's sequence number), the hack is discarded and the flow terminates. Otherwise, the execution flow continues with the evaluation of the *Link Acceptance* policy, as previously explained, and the TX_EXP field is set according to P_{hack}. On the other hand, if the hack is destined to other node than the local node, the $NEIGHS$ field of n's entry in the *Neighbors Table* is updated. This process is not illustrated in the Figure 3.1 since this is an optional behavior of some discovery solutions.

Finally, in the case of (v), the framework receives a timer encoding the expiration of a timestamp of a neighbor n. In that event, if the TX_EXP field of n's entry is expired successively more than the number of *hack misses* (another secondary parameter of the framework), the bidirectionality with the neighbor is considered lost. In case the RX_EXP field of n's entry was the one that expired, the *Link Quality* metric is executed to obtain a new value for the RX_LQ field of n's entry. Then, if the times the RX_EXP field expired successively is less than the number of *hello misses* (a secondary parameter of the framework), this field is updated according to P_{hello}. Otherwise, if the number of successive expirations is equal to the number of *hello misses*, then the neighbor is considered as lost. Finally, the timer of the neighbor is reset to the minimum between RX_EXP and TX_EXP (if one of them is expired, it is ignored).

3.1.3 Framework Parameters

Our framework is a generic (or meta) discovery protocol that may be parameterized to describe a wide range of discovery protocols with varying characteristics. To specify a discovery protocol in our framework, one only has to specify five parameters, resulting in a tuple (DP, TP, DC, LQ, LA), where DP is the selected *Discovery Pattern*; TP is the *Transmission Periods* employed; DC is the *Discovery Context* configured; LQ is the choosen *Link Quality* metric; and LA is the adopted *Link Admission* policy.

Next, we present various possible alternatives for these parameters. We note that some of them may also be dependent on their own (hyper) parameters, which configure certain aspects of their behavior, such as threshold values, static probabilities, or flags.

3.1.3.1 Discovery Patterns

PeriodicJoint(θ): Periodically send hellos and hacks to all neighbors in a single message. The parameter θ is a boolean flag that encodes whether or not unscheduled messages should also be sent upon the loss or discovery of a neighbor.

PeriodicDisjoint(θ): Periodically send explicit hellos and hacks to all neighbors in distinct messages, with different periodicities. The parameter θ is a boolean flag that encodes whether or not unscheduled messages should also be sent upon the loss or discovery of a neighbor.

Passive: Piggybacks hellos and hacks in outgoing messages of other protocols or applications. There are specialized versions of this pattern that piggyback hellos and hacks on specific messages, such as the one employed in BATMAN's discovery protocol.

Echo(μ): Sends periodic hellos, which are immediately replied with an explicit hack to the neighbor that sent the hello. The parameter μ is a boolean flag that encodes whether or not a hello should be piggybacked in the replying hack.

Traffic(t): Periodically send explicit hellos and hacks in a single message if the local node recently sent (in the last t seconds) any non-discovery traffic.

3.1.3.2 Transmission Periods

Static(P_{hello}, P_{hack}): Defines static intervals of duration P_{hello} and P_{hack} seconds between consecutive periodic transmissions of hellos and hacks, respectively.

Dynamic(w, P, T): Compares the number of new neighbors within a time window of w seconds, called the turnover (t), with the optimal turnover value (T) the nodes should be detecting. P is the initial interval value in seconds (for simplicity, we assume both hellos and hacks use the same interval). If $t < T$ the current interval is increased. If $t > T$ the current interval is decreased. The actual formula used to compute the new intervals is sugested by the authors of [106].

3.1.3.3 Discovery Contexts

$\perp$: Denotes the non-usage of a discovery context, and it is employed in solutions that do not enrich the hellos and hacks with additional information.

Coords: This context extends each hello with the coordinates of the local node in a given location system, such as GPS [45] or Galileo [46].

MPR: This context extends each hack with a boolean flag indicating if the local node selected the hack's destination as an MPR (discussed in §2.3).

3.1.3.4 Link Quality Metrics

1: Encodes the absence of a link quality metric, with the framework always assigning a default quality of 1 (perfect quality) to each link.

MA(w): Moving average of the ratio of hellos successfully received in a window of the last w hellos sent by the neighbor.

EMA(w, α): Exponetial moving average, of parameter α (smoothing factor), of the ratio of hellos successfully received in a window of the last w hellos sent by the neighbor.

3.1.3.5 Link Acceptance Policies

Accept: Always accepts every neighbor regardless of its characteristics.

Hyst(H_{reject}, H_{accept}): Accepts neighbors whose RX_LQ becomes $\geq H_{accept}$ and rejects (i.e., consider as lost) them when RX_LQ becomes $\leq H_{reject}$.

Age(λ): Only accepts neighbors that transmited more than λ hellos since they were found by the local node.

Using these parameters, we can define a large number of protocols found in the literature. Table 3.1 includes an illustrative set of discovery protocols specified trough our framework and containing exemplary values for the hyper-parameters. All time references are in seconds. PDJ and PDJ2 are simplifications of NHDP, striping the optional components (DC and LA). PDD corresponds to Babel's discovery protocol and PDD2 is a different configuration of this protocol. ED and ED2 correspond to variants BATMAN's discovery protocol when considered separately from routing.

3.2 Broadcast Framework

In this section, we present our conceptual framework for specifying a broad spectrum of broadcast solutions for wireless *ad hoc* networks. Our framework corresponds to a generic broadcast protocol for wireless *ad hoc* networks that abstracts the common elements of broadcast solutions, such as detecting and registering the reception of duplicate messages to avoid multiple deliveries of the same message, managing the retransmission process of each message, serialize and deserialize messages and related meta-data, and even implementing broadcast with a limited scope. At the same time, this framework provides a set of parameters that control certain aspects of its behavior, enabling the specification

Label	Ref	DP	TP	DC	LQ	LA
NHDP	[86]	PeriodicJoint(*True*)	Static(5, 5)	MPR	MA(10)	Hyst(0.4, 0.7)
PJD	[86]	PeriodicJoint(*True*)	Static(5, 5)	⊥	MA(10)	Accept
PJD2	[86]	PeriodicJoint(*False*)	Static(5, 5)	⊥	MA(10)	Accept
GPSR's	[93]	PeriodicJoint(*False*)	Static(5, 5)	Coords	MA(10)	Accept
PDD	[87]	PeriodicDisjoint(*True*)	Static(5, 10)	⊥	MA(10)	Accept
PDD2	[87]	PeriodicDisjoint(*False*)	Static(5, 10)	⊥	MA(10)	Accept
ED	[98]	Echo(*False*)	Static(5, 5)	⊥	MA(10)	Accept
ED2	[98]	Echo(*True*)	Static(5, 5)	⊥	MA(10)	Accept
BATMAN's	[98]	Passive	Static(5, 5)	⊥	MA(10)	Accept
TAP	[106]	PeriodicJoint(*False*)	Dynamic(10, 5, 3)	⊥	MA(10)	Accept
EBHP	[101]	Traffic(10)	Static(5,5)	⊥	MA(10)	Accept

Table 3.1: Specification of Discovery Protocols.

of particular broadcast solutions. These parameters correspond to the four fundamental components previously identified in §2.3.3: the *Retransmission Policy*; the *Retransmission Delay*; the *Retransmission Context*; and the *Retransmission Phases*.

In the following, we discuss the auxiliary conceptual data structures used by our framework (§3.2.1), followed by an overview of the framework's operation (§3.2.2), and finally presenting a set of examples of values for the framework's parameters as well as some solutions (§3.2.3).

3.2.1 Conceptual Data Structures

In this section, we will go through the conceptual data structures that are part of our framework.

Seen Messages Every broadcast solution contains a set where the IDs of recently received messages are recorded, which we called *Seen Messages*. This set is used to detect the reception of duplicates to avoid multiple deliveries and retransmissions of the same message. Each message ID has an associated expiration timestamp establishing when it can be removed from this set so that it does not grow indefinitely. The period each ID remains in this set is a secondary parameter of the framework.

Pending Messages The framework contains a set of messages and associated meta-data whose retransmission process is concurrently ongoing, which we called *Pending Messages*. For each message, its ID, payload, the identifier of the protocol that requested the broadcast, current phase, and the results of all previous evaluations of the broadcast policy are registered. This information encodes the current state of the message's retransmission process. While messages are in this set, their IDs must not be removed from the *Seen Messages*.

Copies Table Table of the copies received of each message. For each copy, it is registered the parent, the timestamp, the TTL, and the additional headers appended by the broadcast context (if any), later discussed. While the message is in *Pending Messages*, its copies cannot be removed from this table. This information is relevant for solutions that consider the reception of multiple copies to base their decisions, such as counting copies or computing the neighbors that were already covered.

3.2.2 Operation

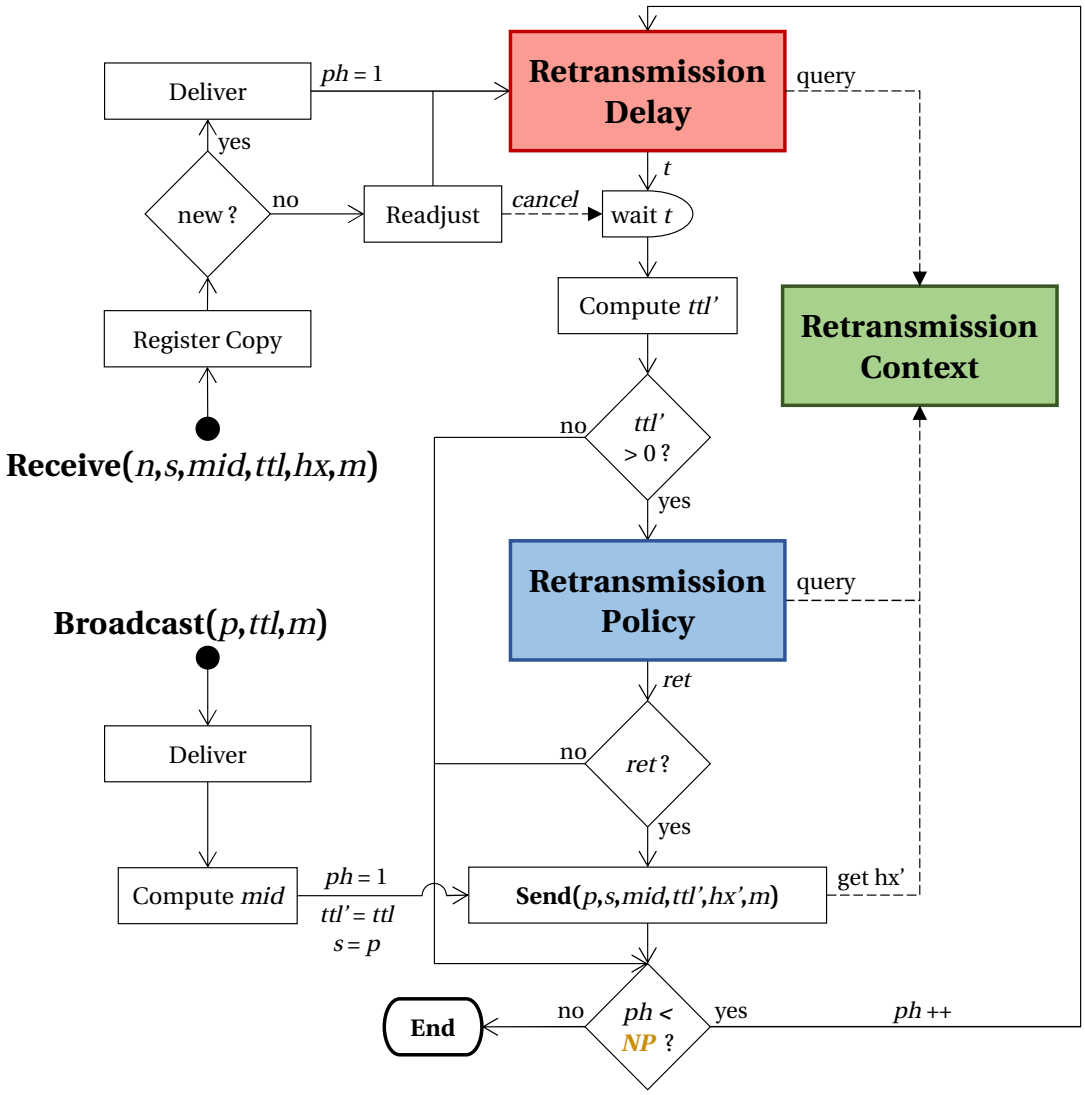


Figure 3.2: Broadcast Framework Execution Flow.

Figure 3.2 presents a simplified flux diagram that captures the workflow of a generic wireless *ad hoc* broadcast protocol which is at the basis of our framework. The diagram describes how the framework handles each message, revealing how its components are intertwined. Multiple messages may have parallel workflows operating at the same node, being that they are independent of each other. There are two ways to initiate the flow of

a given message m: either through i) the request of the broadcast operation by another protocol or application, or through ii) the reception of m from a neighboring node.

In the case of (i), another protocol or application provides the ID of the current node (p), an initial (non-zero) ttl, and the message itself (m). Then, m is immediately delivered (this self-delivery is part of the broadcast definition). Next, the framework assigns a unique identifier to m (mid), which can be, for instance, randomly generated or the concatenation of the current node's ID with a monotonically increasing sequence number. Afterward, m's transmission process begins, at phase (ph) 1. At this point, the framework inserts the mid in the sets *Seen Messages* and *Pending Messages*. This step is vital to ensure that the current node's neighbors' future retransmissions are not mistaken with a new message and do not trigger a new delivery and subsequent retransmission. Next, the TTL to be sent with the message (ttl') is set as ttl, and the source ID (s) is set as the current node's ID (p). Optionally, the broadcast context may piggyback additional headers (hx) to enable the nodes to gather additional information, such as the number of hops the message traveled or the ID of the first parent of the local node. Finally, the message is sent (with one-hop broadcast) along with all the complementary meta-data.

In the case of (ii), the framework receives the message m from a neighbor n. The first step is to register this reception in the *Copies Table*. Next, the framework consults the *Seen Messages* to verify if m is a new message or a duplicate (using the mid). If m is a new message, the framework simply delivers it and inserts an entry for m in the *Seen Messages*. Then, m's retransmission process begins, at phase (ph) 1. The first step of m's retransmission process is to obtain a jitter to wait before attempting to retransmit this message. This jitter is computed by invoking the *Retransmission Delay* parameter. Note that the function that computes the jitters has access to the current phase value (ph) to enable employing different delays in each phase if intended, and it may query the *Retransmission Context*. While waiting for this jitter to expire, the framework may receive duplicate copies of m. In this case, and if the framework is still processing m, it simply registers a new entry for the new copy in the *Copies Table*. Additionally, the *Retransmission Delay* parameter may be optionally executed again to readjust the remaining jitter. This functionality is relevant to solutions that dynamically extend the duration of the jitter after each copy is received. Upon the eventual expiration of the jitter, the framework determines the new TTL to use for the message (ttl') by selecting the maximum TTL carried in each copy received and decrementing it by one. If this new TTL is greater than zero, the framework executes the *Retransmission Policy* to determine if m should be retransmitted by the local node or not. Similar to the retransmission delay, the retransmission policy also has access to the current phase value (ph) to employ distinct strategies in each phase and can also consult the *Retransmission Context*. In addition, the policy can also consult the *Copies Table* to obtain information regarding the duplicate messages received. If the retransmission policy chooses to retransmit m, the framework retransmits it (with one-hop broadcast) along with the complementary meta-data, including the local node's ID (p), the ID of the broadcast source (s), and the new TTL (ttl'). Optionally, the broadcast context may

piggyback additional information (hx'). On the other hand, if the new TTL is equal to zero or the broadcast policy chooses not to retransmit m, the framework does not perform any transmission and simply continues the execution flow.

In either case ((i) and (ii)), the execution flow continues with the framework verifying if it reached the last retransmission phase for this message by comparing m's current phase (ph) with the *Retransmission Phase* parameter (represented as NP in Figure 3.2). If there are additional phases, m's current phase is incremented (ph++), and the framework returns to the beginning of the retransmission process, obtaining a new jitter from the *Retransmission Delay*. Otherwise, m's execution flow terminates, and its associated entries in the *Pending Messages* and the *Copies Table* can be safely discarded. However, it is important to not remove the *mid* from the *Seen Messages* for some additional time to avoid a new copy received in the future from re-initiating this execution flow.

3.2.3 Framework Parameters

Our framework represents a generic (or meta) broadcast protocol that can be parameterized to express a multitude of different algorithms with different properties and strategies. To specify a broadcast algorithm in our framework, one only has specify four parameters, resulting in a tuple (RP, RD, RC, NP), where RP stands for the *Retransmission Policy* function, RD stands for the *Retransmission Delay* function, RC stands for the *Retransmission Context* module, and finally, NP denotes the number of *Retransmission Phases* configured for the algorithm.

In the following we discuss some alternatives of possible values for these parameters. Some of these parameters can also depend on their own (hyper) parameters that configure certain aspects of their behavior, such as some threshold values or static probabilities.

3.2.4 Retransmission Contexts

$\perp$: Denotes the non-usage of a retransmission context, and it is employed in solutions that do not take into account any external environmental information to base their decisions.

Power: This context retrieves and records the RSSI of the transmission associated with each copy received [107, 129, 134, 139]. These values may be normalized to a common reference and averaged to smooth small fluctuations. However, this context requires support from the hardware to retrieve this information.

Coordinates: This context provides to each node its coordinates in a given location system, such as GPS or Galileo [90, 107, 139] Additionally, this context may also piggyback in every outbound message the location of the local node for its neighbors to become aware of its location.

Hops: Piggybacks to each outbound message the number of hops it traveled, i.e., it retrieves the number of hops of the first copy received and increments it by one [138].

ParentDelay: Appends to each outbound message the value of the initial retransmission delay computed upon the reception of the first copy of the current message [125].

Parents: This context appends to each outbound message the IDs of the first parent of the local node, informing the receivers of the message of the last two-hops [123, 129].

Routes: Piggybacks to each outbound message the route it traveled, i.e., it retrieves the route of the first copy and appends the ID of the local node [96].

Neighbors(h): This context acquires neighborhood information up to h hops [95, 96, 128, 147]. This information can be retrieved from configuration files in static settings (i.e., when nodes are stationary) or from a companion neighbor discovery protocol, such as our discovery framework (§3.1)). Each broadcast solution that uses this context can resort to different discovery protocols that offer different information, such as NHDP to obtain the MPRs. However, we employ this generic notation for this context to simplify the exposition.

Delegation: This context retrieves from an external source (e.g., companion protocol) the set of neighbors selected by the local node has its delegated neighbors. These sets can be static, i.e., independent of the messages [147], or dynamic, i.e., computed for each message [96]. Clusters [107, 122] are a particular case of static delegation where the gateways and cluster-heads are selected as delegated neighbors. In addition, this policy also retrieves whether the local node was selected as a delegated neighbor by one of its parents.

3.2.5 Retransmission Policies

Always: This simple policy always decides to retransmit each message [107].

Probability(λ): This policy decides to retransmit each message with λ probability [130].

HopsProbability(λ, k): Always retransmit if the hops traveled by the message is $\leq k$, and retransmit with λ probability otherwise (requires the Hops context) [130].

NeighsProbability(λ_1, k, λ_2, n): Always retransmit if the hops traveled by the message is $\leq k$, retransmit with λ_2 probability if the local node's number of neighbors is $\leq n$,

and retransmit with λ_1 probability otherwise (requires the Hops and Neighbors contexts) [130].

CountProbability(λ, k, m): Decides as in HopsProbability when the message's current phase is equal to 1, and decides to retransmit if the number of copies received is $< m$ when the current phase is equal to 2 (requires the Hops context) [130].

DynamicProbability: At each phase ph, if the local node has not retransmitted the current message yet, this policy decides to retransmit with probability given by the formula $1/((n+1)-(3ph/n))$, where n is the first parent's number of neighbors (requires the Neighbors context) [133].

Count(c): This policy decides to retransmit if the number of copies received (including the first) is below c [107, 125, 134].

FlowCount(c): This policy is equal to Count if the message's current phase is equal to 1 and is equal to a variant of Count that only counts some copies if the phase is equal to 2. This later variant only counts copies received from nodes that either share the same first parent or whose transmission power is higher than the one of the first copy received (requires the Parents and Power contexts) [129].

HopCount: This policy decides to retransmit if no duplicate copy was received with a number of hops higher than the first copy (requires the Hops context) [138].

Distance(D): This policy decides to retransmit if the distance to the closest parent is higher than D (requires the Power context) [107].

DynamicDistance(D_1, ..., D_n): This policy selects the distance threshold from the sequence of parameters D_i whose index i corresponds to the number of copies of the current message received, and decides to retransmit if the distance to the closest parent is higher than the selected threshold (requires the Power context) [139].

AdditionalArea(ϵ): This policy decides to retransmit if the portion of the area within the local node's transmission range that was not covered by the transmission ranges of all the parents is above ϵ (requires the Coordinates context) [107].

CoveredNeighbors: This policy decides to not retransmit if all the local node's neighbors were already covered by the transmissions of local node's parents (requires the **Neighbors(2)** context) [128].

PriorityCoveredNeighbors: This policy is a variant of the previous that further verifies if future retransmissions will cover all the neighbors that were not yet covered, based on a set of priorities (that correspond to the number of neighbors using node IDs in decreasing order to break ties). It assumes that all the nodes that receive a copy of a message from a neighbor with lower priority will retransmit it and uses this assumption recursively until either all the neighbors become covered or there are no more future transmissions to occur [95].

Delegated(ω) This policy decides to retransmit if the local node was selected as a delegated neighbor by one of its parents (requires a context that informs if the local node was selected or not) [147]. If $\omega = True$, this policy also decides to retransmit if one the parents for the current message is not know or is not bidirectional [96].

3.2.6 Retransmission Delays

0: Encodes the absence of a retransmission delay function, with the framework immediately evaluating the retransmission policy. This configuration is sometimes employed in many solutions; however, it may incur broadcast storms [107].

Random(t): Computes a uniformly distributed delay in the interval between 0 and t [107].

RandomEx(t): Computes a delay equal to the sum of an initial delay resultant of the Random(t) function and the result of t minus the initial delay computed by the first parent (requires the ParentDelay context) [125]. Furthermore, upon the reception of subsequent copies of a message, this function dynamically extends the remaining delay by $2t$.

Exp(t, μ): Computes a exponencially distributed delay with average of $\mu(t/2)$, truncated in the interval between 0 and t [158].

PowerD(t): Computes a delay inversely proportional to the distance between the local node and its first parent in the interval between 0 and t (requires the Coordinates or Power contexts to estimate the distances) [107, 134, 139].

PowerDEx(t): Computes an initial delay as in PowerD(t) and upon the reception of subsequent copies of a message, this function computes the associated delay with the new parent with the same strategy and, if this new delay is greater than the delay previously used, the new delay replaces the latter, dynamically increasing the remaining waiting period (requires the Coordinates or Power contexts to estimate the distances) [129]. In addition, if the message's current phase is equal to 2, this function simply returns the

last delay employed (i.e. the highest).

NeighsD(t): Computes a uniformly distributed delay between 0 and $t \times (1+d_m)/(1+d_n)$, where d_m is the maximum number of neighbors of the local node's neighbors and d_n is the local node's neighbors (requires Neighbors(2) context) [128]. This function causes the nodes that have more neighbors than their neighbors to have higher retransmission priority.

With these parameters, we can define a large number of algorithms found troughout the literature. Table 3.2 contains an illustrative set of examples of these algorithms with possible configurations for its hyper-parameters. All time references are in milliseconds.

Label	Ref	RP	RD	RC	NP
EagerFlooding	[107]	Always	0	$\perp$	1
Flooding	[107]	Always	Random(250)	$\perp$	1
D-Flooding	[123]	Always	PowerD(250)	Power	1
Gossip1	[130]	Probability(0.8)	Random(250)	$\perp$	1
Gossip1Hops	[130]	HopsProbability(0.8, 3)	Random(250)	Hops	1
Gossip1Exp	[158]	Probability(0.8)	Exp(250, 0.7)	$\perp$	1
Gossip2	[130]	NeighsProbability(0.8, 3, 0.95, 3)	Random(250)	Hops∪Neighbors(1)	1
Gossip3	[130]	CountProbability(0.8, 3, 3)	Random(250)	Hops	2
DAFP	[133]	DynamicProbability	Random(63)	Neighbors(1)	4
CBS	[107]	Count(2)	Random(250)	$\perp$	1
HCA-RADEx	[125]	Count(2)	RandomEx(80)	ParentDelay	1
PAMPA	[134]	Count(2)	PowerD(250)	Power	1
FABA	[129]	FlowCount(2)	PowerDEx(250)	Power∪Parents	2
HCAB	[138]	HopCount	Random(250)	Hops	1
DBS	[107]	Distance(50)	Random(250)	Power	1
DibA	[139]	DynamicDistance(10, 50, 90)	Random(250)	Power	1
LBS	[107]	AdditionalArea(0.025)	Random(250)	Coordinates	1
SBA	[128]	CoveredNeighbors	NeighsD(250)	Neighbors(2)	1
LENWB	[95]	PriorityCoveredNeighbors	Random(250)	Neighbors(3)	1
CLS	[107]	Delegated(*False*)	Random(250)	Neighbors(1)∪Delegation	1
MPR	[147]	Delegated(*False*)	Random(250)	Neighbors(2)∪Delagation	1
AHBP-EX	[96]	Delegated(*True*)	Random(250)	Neighbors(2)∪Routes∪Delagation	1

Table 3.2: Specification of Broadcast Algorithms.

3.2.7 Mingling with Routing

The broadcast framework is suited for generic use cases that consider the broadcast primitive as a black-box. However, most of the existing routing solutions have a broadcast sub-protocol embedded within them. This architecture allows the routing solution to share information with broadcast (and vice-versa) and freely affect its operation. In our case, since broadcast and routing are isolated protocols that do not share memory, it is necessary to add additional features to the framework to support these specific interactions. In this sense, we made the following additions to this framework:

Meta-data Delivery Many routing solutions require assessing the meta-data of the several copies of each broadcast message received. Examples of meta-data consulted are knowing the broadcast source, the parents' ID, the full routes traveled by the message, or the accumulated cost. Therefore, we extended the broadcast framework to also deliver the gathered meta-data along with each message.

Late Delivery If messages are immediately delivered after being received, only the first copy's meta-data is included. However, routing solutions may require information of all copies. We identified two ways to address this: i) notify the destination process after each copy received or ii) deliver the message only after its execution flow ends —*late delivery*. In the case of i), the broadcast semantics is modified since each message is delivered more than once and it requires adapting the protocols to handle the additional notifications. In the case of ii), the modification is agnostic to other protocols. As such, we chose to integrate the latter into our framework by using a secondary parameter that toggles this behavior. However, it increases the delay between the reception of a message and its delivery.

BiFlooding Flooding variant that ensures when a node receives a copy of a given message from a bidirectional neighbor, then the route taken by the message is all comprised of bidirectional links. This solution is materialized with a policy that decides to retransmit if the link with at least one of the parents is bidirectional and a context that provides neighborhood information. This solution can be parametrized with a boolean flag with the value $True$ to only retransmit if a copy was received from the selected next-hop (i.e., the neighbor with the best route) towards the broadcast source. Variants of this solution are employed in many routing solutions to propagate control messages, such as route requests in reactive solutions and OGMs in BATMAN.

RouteCostContext This context piggybacks to each outbound message the accumulated cost of the route it traveled. It retrieves the cost of each link with a neighbor from a companion routing protocol. This context is employed in several route discovery strategies.

Parallel Specifications Hybrid routing solutions may require distinct broadcast strategies. Having parallel executions of this framework incurs more memory consumption and computational overhead. Therefore, we extended this framework to support multiple broadcast solutions simultaneously. As such, we added another parameter to the broadcast request operation that specifies the algorithm to employ for that message, being this parameter piggybacked on the message when it is transmitted.

3.3 Routing Framework

This section presents our conceptual framework for specifying routing protocols for wireless *ad hoc* networks. This framework abstracts the shared components of routing solutions, such as managing a routing table and neighbors table, forward messages, and serializing and deserializing messages. Simultaneously, this framework provides a set of parameters that govern certain aspects of its operation, allowing it to capture the behavior of specific routing solutions. These parameters correspond to the five fundamental components of routing solutions previously identified in §2.4.3: the *Forwarding Strategy*; the *Announce Period*; the *Cost Function*; the *Routing Strategy*; and the *Dissemination Strategy*.

Next, we discuss the auxiliary conceptual data structures used by our framework (§3.3.1), followed by an overview of the framework's operation (§3.3.2), and finishing with a set of examples of values for the parameters of the framework as well as some solutions (§3.3.3).

3.3.1 Conceptual Data Structures

Here we discuss the conceptual data structures that are part of our framework.

Neighbors Table An essential piece in most routing solutions is a table where the discovered neighbors and the properties of their links with the local node are recorded. This table is distinct from the one contained in the discovery framework since both frameworks do not share memory. In addition, this table requires managing less information since it does not support the discovery process. Most of the information recorded in this table is obtained by the notifications from the discovery framework. Each entry of this table corresponds to a tuple $(ID, MAC, RX\_COST, TX\_COST, B, FOUND)$, where: ID is the ID of the neighbor; MAC is the MAC address of the neighbor; RX_COST is the receiving link cost metric; TX_COST is the sending link cost metric; B is a flag indicating if the link with the neighbor is bidirectional or not; and $FOUND$ is a timestamp of the moment the neighbor was found.

Routing Table Most routing solutions resort to a table where they store pre-computed routes. Each entry of this table encodes a route as a tuple $(DEST, NEXT\_HOP, COST)$, where: $DEST$ is the ID of the destination node; $NEXT\_HOP$ is the ID of the neighbor selected as next-hop for this route; and $COST$ is the total cost of the route. Multiple routes for the same destination can be stored in this table. This basic routing table is extended with additional fields in some routing solutions, such as DSR that stores the full routes and not only the next-hop.

Seen Messages Set where the IDs of recently received messages to forward are recorded. This set is used to detect the reception of duplicates to avoid multiple deliveries and/or retransmissions of the same message when retransmissions at each hop or simultaneous

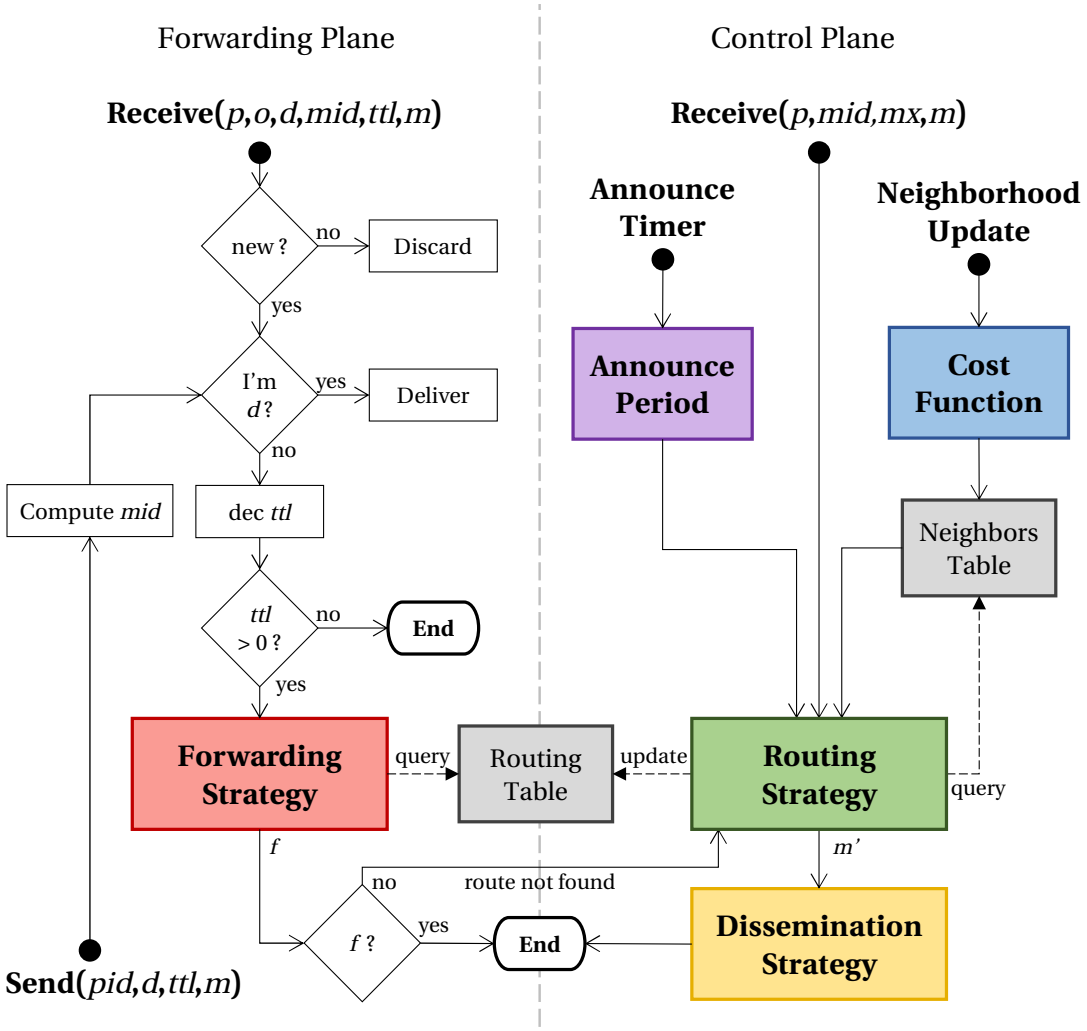


Figure 3.3: Routing Framework Execution Flow.

propagations through multiple routes are employed. Each message ID has an associated expiration timestamp establishing when it can be removed from this set so that it does not grow indefinitely. The period each ID remains in this set is a secondary parameter of the framework.

3.3.2 Operation

Figure 3.3 illustrates the execution flow captured by our framework, which is divided into two parts: the *Control Plane* (on the right), responsible for computing routes, and the *Forwarding Plane* (on the left), responsible for forwarding applicational messages.

3.3.2.1 Control Plane

The control plane is responsible for processing internal events to manage the computation of routes. There are three main entry points in this plane: *i*) a neighborhood update; *ii*) the set off of an announce timer; and *iii*) the reception of a control message.

Neighborhood updates are provided by an external discovery protocol, such as our discovery framework (§3.1). A neighborhood update must encode either the discovery, suspicion of failure, or an update to the state of a communication link of a neighboring node. This update is processed by the cost function, which assigns a cost metric to that neighbor. Next, the framework updates its internal *Neighbors Table* according to the received information.

The announce timer is a periodic timer that informs the *Routing Strategy* to disseminate a new control message, and is employed by protocols following a proactive or hybrid strategy. When the announce timer sets off, it is first processed by the announce period component which is responsible to reset it. This enables the usage of dynamic periods for announcements [196].

A control message is composed of four parts: (p, mid, mx, m), respectively, the identifier of the node that generated the control message, the message's unique identifier, meta-data obtained from the message's propagation and which is associated with a specific dissemination strategy, and the message payload that encodes data specific to the *Routing Strategy*.

These three events flow into the main component of the control plane: the *Routing Strategy*. The *Routing Strategy* evaluates these events, which may lead into an update on the routing table and/or the dissemination of a new control message. On either case, the *Routing Strategy* may query the framework's internal *Neighbors Table* to obtain cost metrics and bidirectionality information to compute or update a new or existing route. Finally, in the case a new control message is to be disseminated, the framework deliveres it to the *Dissemination Strategy* that is responsible to send the message to all intended destinations.

3.3.2.2 Forwarding Plane

The forwarding plane is responsible for handling the flow of applicational messages and applying a *Forwarding Strategy*, which can be triggered by two events: *i*) a request to send a message to an arbitrary destination or *ii*) the reception of a forwarded message from a neighboring node.

To request the forwarding of a message, the application must provide the following parameters: (p, d, ttl, m), respectively, the local node's identifier, the identifier of the destination node, a time-to-live, and the message payload. Upon receiving such request, the framework first generates a message identifier (mid) to uniquely identify the message in the network. Then, the message enters the flow of received messages in the forwarding plane.

Upon the reception of a message, the forwarding plane first checks if it has already processed it, discarding it if so. Next, if the destination of the message is the local node, the message is delivered to the application, continuing otherwise to the next processing stage, where the *ttl* is decremented and verified. If the *ttl* has expired, the forwarding of the message ends, otherwise the message is delegated to the forwarding strategy which will obtain the next-hop, potentially consulting the framework's routing table, and send the message to it. If no next-hop was found, the *Routing Strategy* in the control plane is notified, possibly requesting the dissemination of a new control message, such as a route request in reactive protocols [94, 102]. Otherwise, the execution flow for that message ends.

3.3.3 Framework Parameters

Our framework represents a generic (or meta) routing protocol that can be parameterized to express a multitude of different protocols with different properties and strategies. To specify a routing protocol in our framework, one only has specify five parameters, resulting in a tuple (FS, AP, CF, RS, DS), where FS is the *Forwarding Strategy*, AP is the *Announce Period*, CF the *Cost Function*, RS the *Routing Strategy*, and DS is the *Dissemination Strategy*.

In the following, we discuss some alternatives of possible values for these parameters, being that they can also assume a value of $\perp$ to encode not employing a specific parameter. We note that some of these parameters may also be dependent on their own (hyper) parameters, which configure certain aspects of their behavior, such as threshold values, static probabilities, or flags.

3.3.3.1 Forwarding Stategies

Simple: Simply retrieves the first next-hop contained in the routing table.

Multipath: Multiple next-hops are retrieved from the routing table and one is selected according to some criteria.

Source: The complete route is retrieved and used as an header for the message, allowing intermediary nodes to become aware of their next-hops.

Acked(s): Extends a strategy s with explicit acknowledgments and retransmissions of forwarded messages.

Opportunistic: The next-hop is dynamically selected according to a given algorithm.

Geographic: The next-hop is the neighbor geographically closer to the destination.

3.3.3.2 Announce Period

The *Announce Period* is a natural number that represents the interval between periodic announcements of control messages. It can be the result of a function in the case dynamic periods are employed. The value 0 encodes that there are no periodic announcements.

3.3.3.3 Cost Functions

HOPS: Is always 1 so that the routes' cost is their number of hops.

ETX: Estimates the expected number of retransmissions for a successful forwarding based on the link qualities [165].

ETT: Estimates the expected time for a successful forwarding based on link qualities and data-rates [166].

AGE: The time elapsed since the neighbor was detected is used to estimate the cost, with older neighbors representing better links (i.e., more stable) [159].

DIST: Uses the geographic distance between the local node and the neighbor, with higher costs representing better links (i.e., closer to the destination) [93].

MCX: The number of control messages received from different sources and neighbors is considered, with higher counts representing better links [98].

3.3.3.4 Routing Strategies

DistVec(m, π, ϕ): Disseminates across the network a vector of tuples containing a destination and associated cost and sequence number, which always includes a tuple with the local node's identity (with a cost of 0 and a local sequence number). The m parameter encodes whether the vector dissemination is proactive (Pro) or reactive (Re). The tuples of the disseminated vector are selected by the π parameter (a function) from the local node's routing table. Examples of this function can be All, for all destinations contained in the routing table, or $Single$, for only the identity of the local node. The ϕ parameter corresponds to a function that selects which tuples encode a valid route from the received distance vectors and when to increase the local node's sequence number. Examples of this function can be Inc, which increases the sequence number for every vector disseminated and only accepts tuples whose sequence number is either larger than the largest known sequence number or the cost is lower than the cost of the current route selected for that destination, or Req, which only increases the sequence number upon receiving an explicit request from another node and only accepts tuples that have either higher sequence number or have a lower cost than all the tuples advertised by the local node

for that destination [87]. From the vectors received and the information regarding the properties of the links traveled by the control messages (containing the vectors), this strategy computes the routes to all reachable destinations.

LinkState(φ): Periodically disseminates a small sub-set of the local known topology, allowing all the nodes to gather the global topology which is used to locally calculate routes to all reachable destinations. The sub-set of the known topology that each node disseminates is selected by the function φ passed as parameter and examples of it can be: *BiNeighs*, where the set of bidirectional neighbors of the local node are disseminated [190]; *MPRSlts*, where the *MPR Selectors* (discussed in §2.4.2) of the local node are disseminated [89]; *Fisheye*, where a sub-set of all the links is selected based on their distance to the local node, with closer links being disseminated more often [178]; or *SrcTree*, where part of the tree formed by all the best routes of the local node, called the *Source Tree*, is disseminated [100].

LinkReversal(m): This trategy distributively constructs a DAG for the local node. The m parameter encodes if the DAG's construction is triggered by the local node proactively (*Pro*) or reactively (*Re*). The routes towards a given destination are computed from its DAG by selecting the neighbors with whom the local node have a downstream edge (in the DAG). Cost metrics, if available, can be used to select the best route among these routes. Otherwise, a route is randomly selected.

Zone(i, o, r): A proactive routing strategy i is employed within routing zones with a limited scope of r hops, and a reactive strategy o is employed to compute routes towards nodes outside of these zones. When the parameter r is ommited, it is dynamically selected for each node based on a given selection criteria specific to each solution.

Hierarchy(l): Constructs a hierarchy on the network topology with l levels. On top of this hierarchy, it constructs a logical network, where each node is identified by a tuple $(subnet, host)$, with the subnet identifying the logical group the node belongs to, and the host uniquely identifying the node within the group.

3.3.3.5 Dissemination Strategies

Broad(h): Control messages are broadcast throughout the entire network if h is ∞, or up to limited number of hops h, otherwise. When a sequence $S_1, ..., S_n$ is passed as parameter instead of h, the radius employed in each dissemination follows that sequence in round-robin order.

Border: Control messages are disseminated with bordercast (efficient variant of multicast) to the nodes at the border of routing zones [29].

Uni: Send control messages to a single destination with (network-wide) unicast, leveraging previously computed routes.

With these parameters, we can define a large number of protocols found troughout the literature. Table 3.3 contains an illustrative set of examples of these protocols, with possible configurations for the hyper-parameters. The ETX cost metric was applied in protocols that did not define any cost metric in their specification since this metric is usually more appropriate than simply resorting to HOPS. TORA's *m* parameter was not configured as it is specific for each node. The usage of multiple dissemination strategies (DS column) in the same protocol is encoded in the table as a set of strategies, resultant from the union (∪) of several individual dissemination strategies (omitting the { and } of the usual mathematical notation for sets to not clutter the table entries). In these cases, the resulting set must be accompanied by a function that selects the proper dissemination strategy from the set to employ for each control message to disseminate (this function was omitted from the table). For instance, the hybrid ZRP protocol disseminates periodic announcements within routing zones of radius 5 with broadcast (Broad(5)), route requests with bordercast (Border), and route replies with unicast (Uni). The zone radius of 5 (the third parameter of the Zone routing strategy) was arbitrarily configured as an example.

Label	Ref	FS	AP	CF	RS	DS
OSPF	[190]	Simple	5	ETX	LinkState(*BiNeighs*)	Broad(∞)
OLSR	[89]	Simple	5	ETX	LinkState(*MPRSlts*)	Broad(∞)
FSR	[178]	Simple	5	ETX	LinkState(*Fisheye*)	Broad(1)
FSLS	[115]	Simple	5	ETX	LinkState(*BiNeighs*)	Broad(3,5,∞)
TBRPF	[100]	Simple	5	ETX	LinkState(*SrcTree*)	Broad(1)
DSDV	[160]	Simple	5	ETX	DistVec(*Pro*,*All*,*Inc*)	Broad(1)
BABEL	[87]	Simple	5	ETX	DistVec(*Pro*,*All*,*Req*)	Broad(1)∪Broad(∞)
BATMAN	[98]	Simple	5	MCX	DistVec(*Pro*,*Single*,*Inc*)	Broad(∞)
JOKER	[176]	Opportunistic	5	MCX	DistVec(*Pro*,*Single*,*Inc*)	Broad(∞)
AODV	[94]	Simple	0	ETX	DistVec(*Re*,*Single*,*Inc*)	Broad(∞)∪Uni
DSR	[102]	Source	0	ETX	DistVec(*Re*,*Single*,*Inc*)	Broad(∞)∪Uni
ABR	[159]	Simple	0	AGE	DistVec(*Re*,*Single*,*Inc*)	Broad(∞)∪Uni
ZRP	[29]	Simple	5	ETX	Zone(LinkState(*BiNeighs*),DistVec(*Re*,*Single*,*Inc*),5)	Broad(5)∪Border∪Uni
TORA	[173]	Simple	0	⊥	LinkReversal(*m*)	Broad(∞)∪Broad(1)
SHARP	[177]	Simple	5	ETX	Zone(LinkReversal(*Pro*),DistVec(*Re*,*Single*,*Inc*))	Broad(∞)∪Broad(1)∪Uni
GPSR	[93]	Geographic	0	DIST	⊥	⊥
HSR	[187]	Simple	5	ETX	Hierarchy(3)	Broad(∞)

Table 3.3: Specification of Routing Protocols.

3.4 Summary

In this chapter, we presented three generic conceptual frameworks that can be parameterized to capture the behavior of a vast amount of neighbor discovery, broadcast, and routing solutions. Each framework corresponds to a generic protocol that abstracts the solutions' common elements, and that can be parameterized to capture the behavior of

particular instances. For each one of these frameworks, we introduced their components and explained how these components fit within their respective framework.

The neighbor discovery framework can specify discovery solutions through five parameters: (DP, TP, DC, LQ, LA), where DP is the selected *Discovery Pattern*; TP is the *Transmission Periods* employed; DC is the *Discovery Context* configured; LQ is the choosen *Link Quality* metric; and LA is the adopted *Link Admission* policy.

In turn, the broadcast framework can specify broadcast solutions through four parameters: (RP, RD, RC, NP), where RP is the *Retransmission Policy* function, RD is the *Retransmission Delay* function, RC is the *Retransmission Context* module, and finally, NP denotes the number of *Retransmission Phases* configured.

At the same time, the routing framework can specify routing solutions through five parameters: (FS, AP, CF, RS, DS), where FS is the *Forwarding Strategy*, AP is the *Announce Period*, CF the *Cost Function*, RS the *Routing Strategy*, and DS is the *Dissemination Strategy*.

For each one of these frameworks, we presented examples of possible values for their parameters and how to specify a set of representative solutions found in the literature. Leveraging prototypes of these frameworks, we conducted a preliminary experimental evaluation that will be discussed in detail in the next chapter.

Chapter 4

Experimental Evaluation

This chapter presents the methodology and results of our experimental evaluation of a set of representative solutions found in the literature, that were implemented leveraging prototypes of our proposed frameworks. We start by describing the experimental setting and configurations in §4.1, followed by presenting and discussing the experimental results of the Neighbor Discovery (§4.2), Broadcast (§4.3), and Routing (§4.4) solutions.

4.1 Experimental Setting

We implemented prototypes of the three frameworks presented in the previous chapter and all its components in the C programming language resorting to the Yggdrasil framework [197]. Each framework was implemented as a separate protocol, which does not share memory with the other components of the system (i.e., with the applications and the other frameworks). The frameworks can asynchronously communicate with each other through the exchange of messages (within the same node), in a cross-layer architecture. As mentioned in §2.1.3, the frameworks operate leveraging the WiFi technology (802.11b/g/n standard at 2.4 GHz), and no modifications were performed to the network stack of the devices (i.e., no modifications on the MAC and PHY layers).

The experimental evaluation was conducted in a wireless *ad hoc* network composed of 17[1] *Raspberry Pi 3 - model B* devices, that were dispersed through the rooms and hallways (with approximately 30 meters) of our department building across two floors, as schematically illustrated in Figure 4.1. Notice that in the Figure 4.1, the node 4 appears represented in both floors since it was placed at the middle of the stairs between the two floors.

[1] We intended to use 24 devices; however, some of them became unoperational during the period over which we conducted our experimental evaluation.

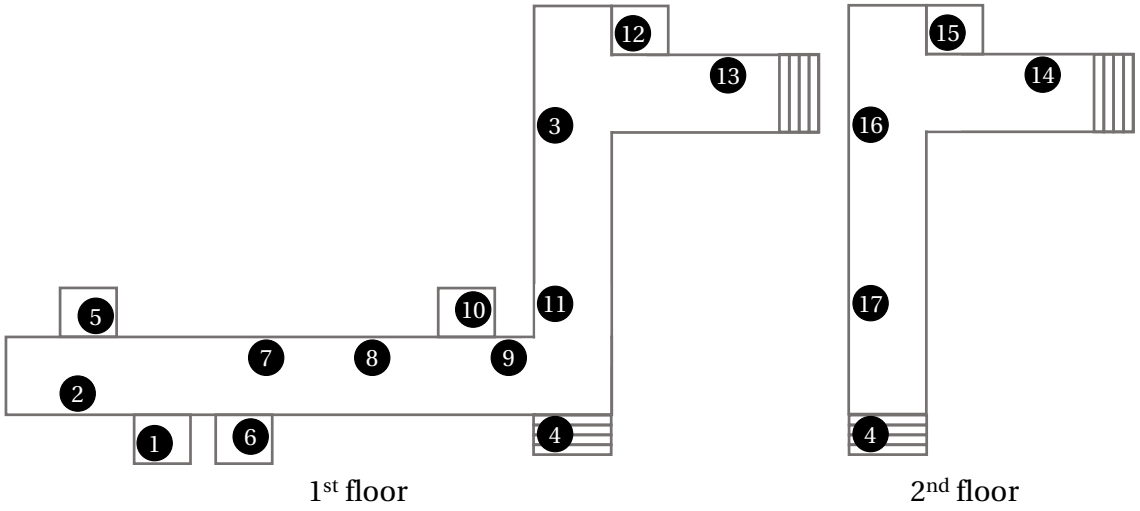


Figure 4.1: Network Deployment.

The prototypes of our frameworks were leveraged to evaluate a set of representative solutions of the three key services. Each experiment consists of the execution of a single solution in a given scenario. Each experiment was executed three times, in random order, with the results showing the average of all runs. Next, we discuss the details of each experiment performed for each service, presenting the elected solutions to evaluate, how these solutions were evaluated, and the metrics that were measured.

4.1.1 Neighbor Discovery

We selected to evaluate six representative neighbor discovery protocols: PJD [86], PJD2 [86], PDD [87], PDD2 [87], ED [98], and ED2 [98], which were configured as indicated in the previous chapter in Table 3.1. The vaue of 5 seconds for the periodic hellos and 10 seconds for the periodic hacks were selected to not cause too much contention and collisions in the wireless medium. The hack periodicity is the double of the hello's as sugested in [87] to attempt to decrease the overall number of transmissions performed by the protocol. These protocols were selected since they comprehend the most employed strategies and are not dependent on the traffic patterns of other protocols and applications. These protocols were alredy discussed in detail on §2.2.

For each neighbor discovery protocol, we measured its *cost*, as the average number of transmissions performed by each node; and its *latency*, as the average time each pair of neighboring nodes required to both nodes consider their link as bidirectional.

We evaluated each discovery protocol in a single scenario with no node faults. In each experiment, each node executed the discovery framework, parameterized as the selected protocol, for a period of 5 minutes. Each node was configured to start the neighbor discovery protocol at a random period up to 60 seconds after the begining of the experiment to avoid biasing the measurements with the synchronization of the nodes.

4.1.2 Broadcast

Among the algorithms presented in the previous chapter in Table 3.2, some require specific hardware features that are not available in commodity hardware, such as accessing the RSSI of transmissions and the coordinates of the devices. In addition, it is unmanageable to evaluate all the remaining algorithms in a real network. As such, from that set, we selected ten representative broadcast algorithms to evaluate: Flooding [107], Gossip1 [130], Gossip1Hops [130], Gossip2 [130], Gossip3 [130], CBS [107], HCAB [138], SBA [128], MPR [147], and AHBP [96]; which were configured as indicated in Table 3.2. These protocols were already discussed in detail on §2.3.

At every 2 seconds, the application decides to request with 0.5 probability the broadcast of a small message to the framework, with an initial TTL equal to ∞. This configuration was selected so that each node requests on average a broadcast every 4 seconds to not cause too much contention and collisions in the wireless medium.

For each broadcast algorithm, we measured its *reliability*, as the average ratio of nodes that delivered each message; its *cost*, as the average number of transmissions incurred to disseminate each message; and its *latency*, as the average maximum accumulated retransmission delays of each message before being delivered. A reliability of 100% implies that all messages were delivered to all correct nodes in the network. Notice that the cost is relative to the total number of nodes. Solutions where each node performs a single transmission per message result in a cost of 17 (as our experimental setup is composed of 17 nodes).

We evaluated each algorithm in three scenarios: one without node faults and two with deterministic node faults of the first five nodes and nine nodes from the sequence 3, 12, 7, 9, 11, 2, 5, 10, 14. In the experiments with faults, these were introduced simultaneously at the middle point of the experiment (2.5 minutes). Preliminary results for this framework were previously presented in [47].

Each node executed the broadcast framework, parameterized as the selected algorithm, a companion neighbor discovery protocol in algorithms that employed the Neighbors context, and a simple application for a period of 5 minutes, including grace periods of 1 minute at the start. The neighbor discovery protocol employed for all neighbor-aware algorithms was PJD (previously summarized in Table 3.1) since this is the most common discovery protocol in the literature and for the discovery process being identical in all cases to not bias the performance measurements. In the evaluation of the MPR broadcast algorithm, the discovery protocol was additionally configured with the MPR discovery context, to allow the nodes to obtain their MPRs. The companion discovery protocol was configured to issue messages every 5 seconds to minimize the collisions and contention in the wireless medium.

4.1.3 Routing

We choose to evaluate five representative routing protocols: OLSR [89], BABEL [87], BATMAN [98], AODV [94], and DSR [102], which were configured as indicated in Table 3.3. The value of 5 seconds for the periodic announcements was selected to not cause too much contention and collisions in the wireless medium. The first three protocols are proactive, and employ different routing strategies, and the other two are reactive, and employ different forwarding strategies. These protocols were selected since they encompass a broad diversity of strategies, they don't require special hardware features, and they are the most discussed in the literature. These protocols were alredy discussed in detail on §2.4.

Each node executed one of the routing protocols, a companion discovery protocol, a companion broadcast protocol, and a simple ping application for a period of 10 minutes, including grace periods of 2 minutes at the beginning and end. The ping application, at every second, requests to the routing protocol to send a message to a randomly selected destination (other than the local node), which upon the reception replies with the same message to the source node. This behavior allows to evaluate the selected routes in both directions.

For each routing protocol, we measured the number of *routing loops*, as the number of applicational messages that were forwarded in a loop, and the average and maximum *route lengths* of the computed routes. In addition, we also measured the protocol's *reliability*, as the ratio of messages that were successfully received back; its *latency*, as the average round-trip-time (RTT) of each message; and its *communication overhead*, as the total number of transmissions incurred by the dissemination of control messages by the routing protocol and all companion protocols.

The reactive routing protocols do not perform buffering of messages while route computation is taking place, being the messages discarded since it would bias the route latency measurments of the application and providing buffering is an optional behavior of these protocols. The neighbor discovery and broadcast companion protocols corresponded to the ones described in the original specification of the routing protocols, with the exception of AODV and DSR which employed a periodic discovery since their original discovery procedure is heavly coupled within the forwarding of messages and relies on the transmission of acknoledgemnts (which were not used to measure the "raw" performance of the routes selected by each routing protocol). In short, OLSR employed PJD2 (with the MPR discovery context) and MPR, BABEL employed PDD and FLOODING, BATMAN employed BATMAN's and BIFLOODING(*True*), AODV employed PJD2 and BIFLOODING(*False*), and DSR employed PJD and BIFLOODING(*False*), as the companion neighbor discovery and broadcast solutions, respectively (all these algorithms and protocols were previously presented and discussed).

We evaluated each protocol in four scenarios: one without node faults and three with deterministic node faults of the first two nodes, five nodes, and nine nodes from the

sequence 3, 12, 7, 9, 11, 2, 5, 10, 14. In the experiments with faults, these were introduced simultaneously at the middle point of the experiment (5 minutes). Furthermore, the nodes configured to fail were never selected to be the destination of messages as to not affect reliability measurements.

4.2 Neighbor Discovery Experimental Results

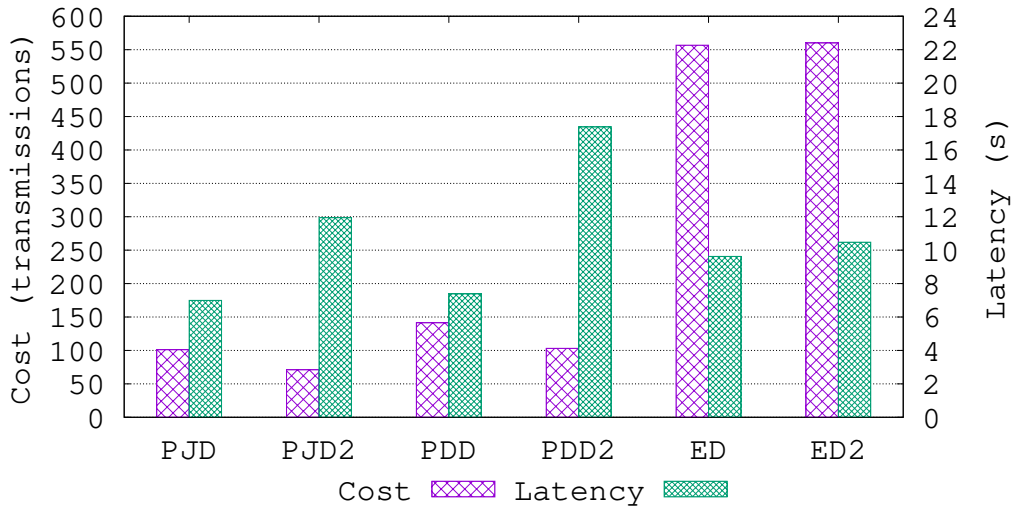


Figure 4.2: Cost and Latency of Neighbor Discovery Protocols.

Figure 4.2 presents the results for the cost, on the left axis, and the latency, on the right axis, for each neighbor discovery protocol. The insight of these plots is that the better the protocol, the lower its latency and cost.

Regarding the cost, PJD2 achieved the best results with 71.1 transmissions per node on average. These results are explained since this protocol combines the transmission of hellos and hacks into a single message and does not perform unscheduled transmissions upon changes in the neighborhood are detected. PJD and PDD2 achieved the second-best results with 101.1 transmissions per node on average. As expected, PJD presented a higher cost than PJD2 due to performing approximately 30 unscheduled transmissions due to topology changes, which indicate that the discovered neighborhoods were highly unstable. PDD2 presented a higher cost than PJD2 due to separating the transmissions of hellos and hacks. PDD achieved the third-best cost of 141.5 transmissions per node on average. These results are due to this protocol separating the transmissions of hellos and hacks and performing unscheduled transmissions. Finally, ED and ED2 achieved the worst cost with approximately 588.2 transmissions per node on average. These results are explained by every node replying with a hack upon the reception of a hello on both protocols, with an average of 6.9 hacks transmitted per hello.

Regarding the latency, PJD and PDD achieved the best results with approximately 7.1 seconds. These results are due to these two protocols having lower costs, which

create less interference in the wireless medium, allied with performing unscheduled transmissions upon changes in the neighborhood. ED and ED2 achieved the second best latency of approximatly 10.0 seconds. We expected these two protocols to achieve the best latency results due to immediate hack transmissions; however, this was not the case. We suspect this behavior was mainly caused by the high amounts of additional interference in the medium caused by the high cost of these two protocols, leading to many messages (i.e., hellos and hacks) not being received by the nodes. In this case, piggybacking hellos on the hacks (in ED2) presented no significant advantage. PJD2 presented the third-best latency of 12.0 seconds, higher than PJD as expected since the first does not perform unscheduled transmissions. PDD2 presented the worst latency of 17.4 seconds, much higher than PDD; this is expected since the first does not perform unscheduled transmissions, and much higher than PJD2 since the transmission period of hacks is the double on PDD2.

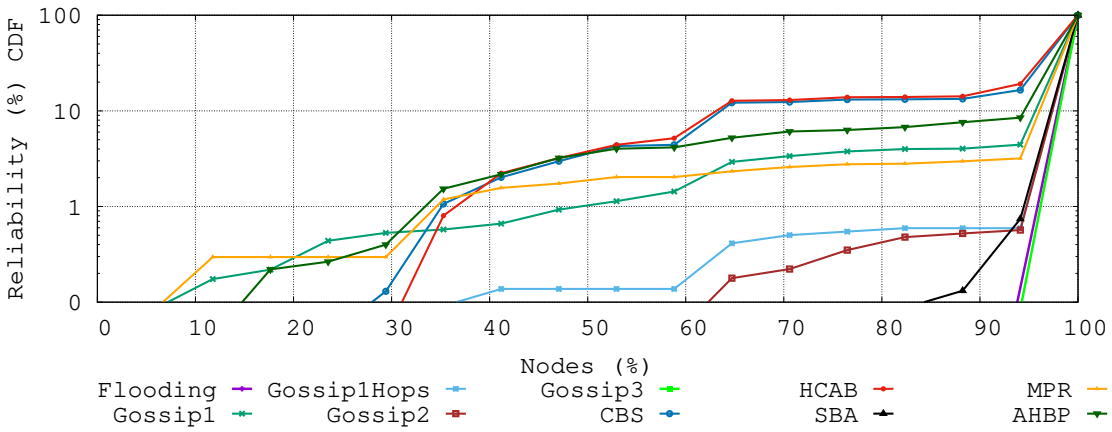
4.3 Broadcast Experimental Results

In each experiment, approximately and on average 762.3, 696.9, and 636.4 messages were broadcast in the scenarios with no faults, five faults, and nine faults, respectively.

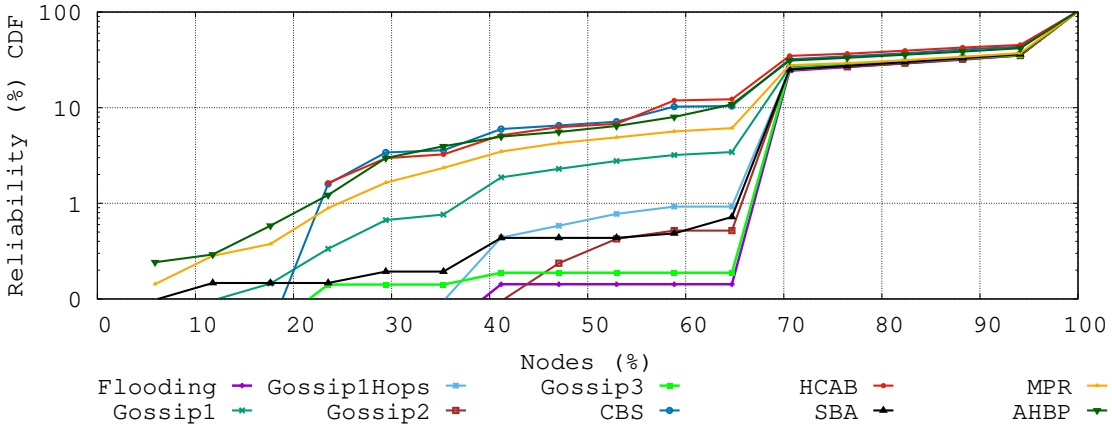
Figure 4.3 presents in each plot the results for the reliability in the three scenarios, with the ratio of messages delivered, represented in the y axis as a Cumulative Distribution Function (CDF) in logarithmic scale, for each ratio of nodes that delivered them, represented in the x axis. The intuition of these plots is that the better the algorithm the lower the number of messages delivered (y axis) for a lower number of nodes (x axis), i.e., the lower the area under the plot.

We note that in the scenarios with faults (Figures 4.3b and 4.3c), due to the introduction of node failures at the middle of the experiments, the total number of nodes that received each message decreases from that point onward. Therefore, in these scenarios, around 50% of all messages should have been ideally received by 100% of all nodes and the other 50% been received by the correct nodes (i.e., that did not fail), i.e., 70% and 47% of the nodes in the scenario with five and nine faults, respectively.

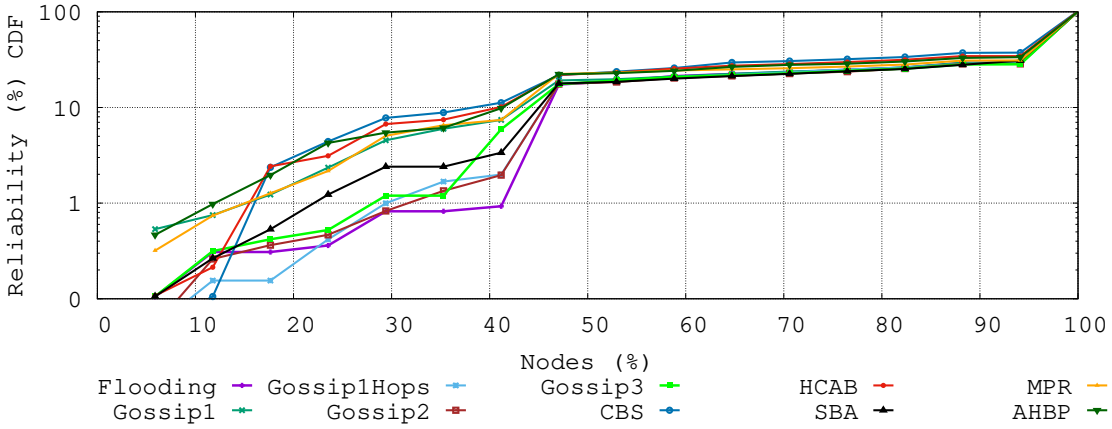
Each algorithm achieved comparable results in the three scenarios, with Flooding consistently presenting the best reliability distributions with approximately all the correct nodes receiving all the messages. Gossip3 presented similar results, having almost the same distribution in all scenarios. Gossip1Hops, Gossip2, and SBA had slightly worst results, with less than 1%, in the first two scenarios, and 10%, in the last, of all the messages having been received by few nodes. The good reliability exhibited by these gossip variants are due to them behaving like Flooding in the majority of the network (i.e., they presented similar cost Figure 4.4a) since the average number of hops of the shortest paths between all the nodes in the network is relatively small. SBA results are explained due to it transmitting if all the neighbors were not already covered, which is a very robust policy, and thus allowing each node to receive the messages. Surprisingly,



(a) No Faults



(b) Five Faults



(c) Nine Faults

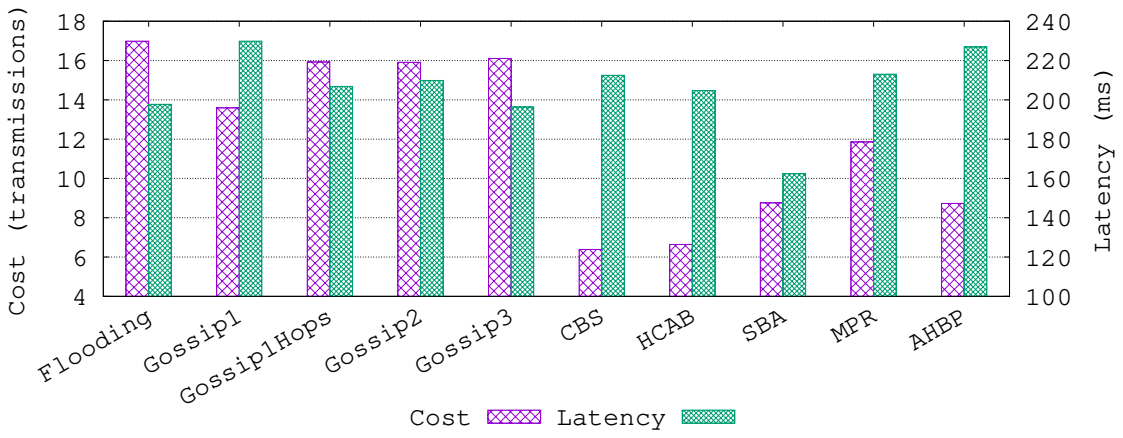
Figure 4.3: Average Reliability of Broadcast Algorithms.

Gossip1, MPR, AHBP, CBS, and HCAB achieved the worst reliability distributions, with many messages (approximately 13%) being only delivered by a few correct nodes. These results were expected from Gossip1 since each node does not take into account the topology of the network and blindly retransmit each message with a given probability. We suspect the behavior of MPR and AHBP was caused due to highly unstable neighborhood relations, which led to the selected delegated neighbors including unstable neighbors that not always received the messages which they were delegated to transmit. The results of CBS and HCAB are explained due the network having regions where the nodes have many neighbors in common, which cause the reception of many copies by some nodes with different hop counts, which makes their policies (based on counting copies and counting hops) to decide not to retransmit in many occasions.

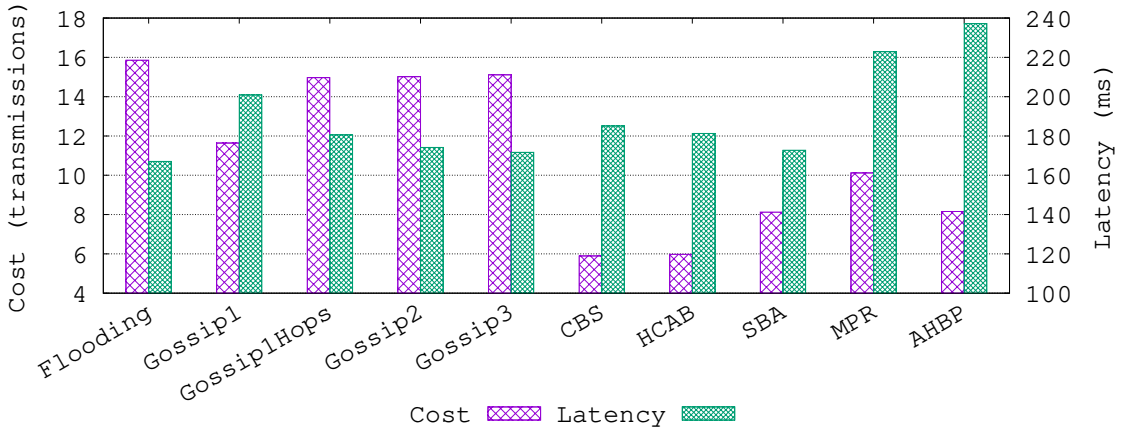
Figure 4.4 presents in each plot the results for the cost, represented in the left y axis as the average number of transmissions performed to disseminate each message, and the latency, represented in the right y axis as the average accumulated retransmission delays before the last delivery of each message, for each algorithm, represented in the x axis. The intuition of these plots is that the better the algorithm the lower the cost and latency.

Regarding the cost, CBS and HCAB exhibit the lowest values in all scenarios for the reasons previously explained, hence their low reliability. SBA and AHBP presented the second lowest cost in all scenarios. These results show that the policy employed by SBA is effective in achieving high reliability with a low cost. The cost results of AHBP are explained due to its policy selecting a small sub-set of neighbors to retransmit allied with many nodes not receiving the messages (low reliability). MPR presented a cost slightly above AHBP since it selects a static sub-set of neighbors to retransmit each message while the latter selects a sub-set per message, which takes into account the routes traveled by the message. Gossip1 presented a cost of 13.6 transmissions per broadcast message in the scenario with no faults (Figure 4.4a). This value corresponds to approximately 80% of the number of nodes, as expected due to Gossip1's simple policy of retransmitting with 80% probability. This protocol exhibited slightly lower cost in the scenarios with faults (Figures 4.4b and 4.4c) since the number of correct nodes decreased. Gossip1Hops, Gossip2, and Gossip3 had the second highest cost in all scenarios, since their policies behaved similarly to Flooding in the majority of the nodes. Finally, Flodding achieved the highest cost with all the nodes retransmitting each message, as expected.

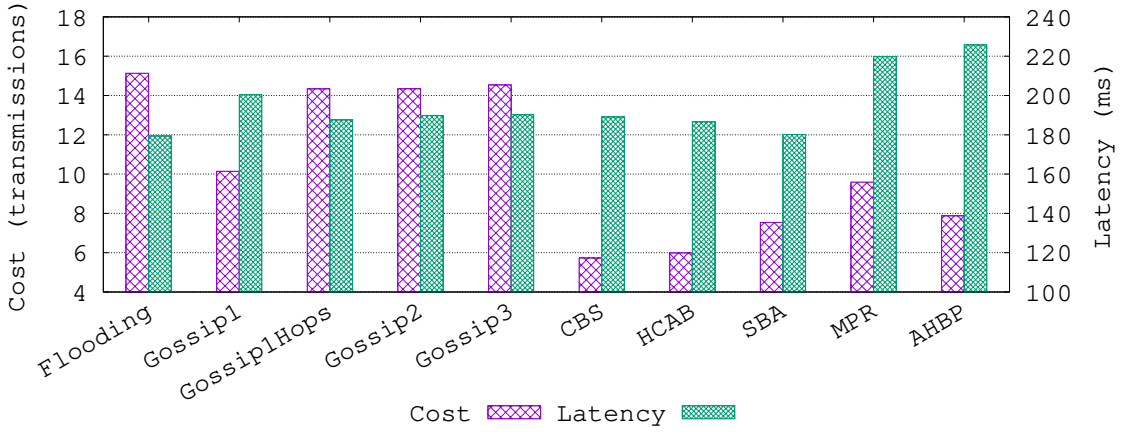
All algorithms except SBA employed the Random retransmission delay. Hence, their latencies are related to the average number of hops each message travels before being delivered by each node, which is dictated by the policy employed. In this sense, Flooding and Gossip3 achieved the lowest latency among these algorithms in the scenario with no faults (Figure 4.4a), and Flooding, Gossip1Hops, Gossip2, Gossip3, CBS, and HCAB achieved the lowest latency among these algorithms in the scenarios with faults (Figures 4.4b and 4.4c). AHBP obtained the worst latency across the three scenarios, with Gossip1 having the same results in the scenario with no faults (Figure 4.4a). MPR



(a) No Faults



(b) Five Faults



(c) Nine Faults

Figure 4.4: Average Cost and Latency of Broadcast Algorithms.

presented a latency slightly below AHBP since some messages traveled through paths with lower hop-count due to a slightly higher number of transmissions performed by its policy. Finally, SBA attained the lowest latency in all scenarios due to its retransmission delay giving retransmission priority to nodes with more neighbors than their neighbors, which enabled the messages to reach all the nodes faster. In the scenario with nine faults, SBA's latency is close to other algorithms due to the network being much smaller after the failures.

Overall, SBA was the best algorithm across the three scenarios, having presented one of the highest reliabilities, one of the lowest costs, and the lowest latency.

4.4 Routing Experimental Results

Scenario	Protocol	Loops	Avg. Route Length	Max Route Length
No Faults	OLSR	4.67	2.22	6.33
	BABEL	0	2.09	5.67
	BATMAN	28.67	2.18	7.33
	AODV	0.33	2.05	6.00
	DSR	0	2.17	6.00
2 Faults	OLSR	2.33	2.11	6.00
	BABEL	0	1.94	5.33
	BATMAN	69.00	2.23	7.33
	AODV	0	2.02	6.00
	DSR	0	2.13	6.00
5 Faults	OLSR	8.00	2.23	6.00
	BABEL	0	2.07	6.00
	BATMAN	43.33	2.17	6.33
	AODV	0	2.06	6.33
	DSR	0	2.23	6.67
9 Faults	OLSR	1.67	2.11	6.33
	BABEL	0	2.05	5.00
	BATMAN	30.00	2.17	6.67
	AODV	0	2.04	5.67
	DSR	0	2.11	5.67

Table 4.1: Route Statistics of Routing Protocols.

Table 4.1 presents the number of routing loops, the average route length, and the maximum route length that were observed on average for each protocol across all scenarios.

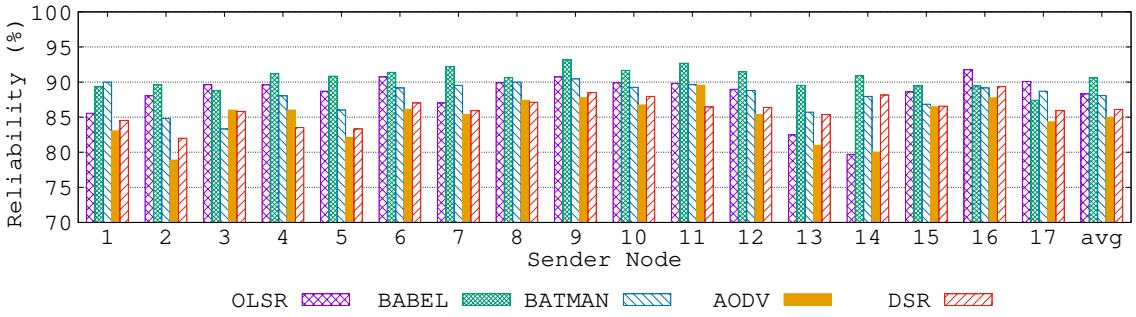
Regarding the routing loops, BATMAN displayed the highest amount in every scenario. These loops emerge since BATMAN's routing strategy has no loop prevention mechanism, and the combination of BATMAN's cost function and dissemination strategy allied with unstable neighborhoods cause the nodes to frequently change their selected next-hops. OLSR presented a very small amount of routing loops. These loops arise during the propagation of updated routing information (i.e., the MPR Selectors),

which leads to some nodes having updated routes while others still having older routes, which, in turn, can cause short-lived loops to form. As expected, the protocols which employed a distance-vector routing strategy did not form any loop (due to the usage of sequence numbers). The exception was AODV that formed one loop in one of the executions in the scenario with no faults due to the asynchronous communication between the broadcast and routing frameworks. One node successfully retransmitted a broadcast message (containing a routing control message) to a neighbor. However, the neighbor was faster to deliver that message, and thus it updated its routing table faster than the first node. Then, the first node used an older route to forward an applicational message towards its neighbor already employing the new route, whose next-hop was the first node, thus creating a short-lived loop that was immediately resolved after the first node delivered the broadcast message. Therefore, our communication model between the frameworks infringed the AODV's loop-freedom guarantees. Addressing this issue was left as future work.

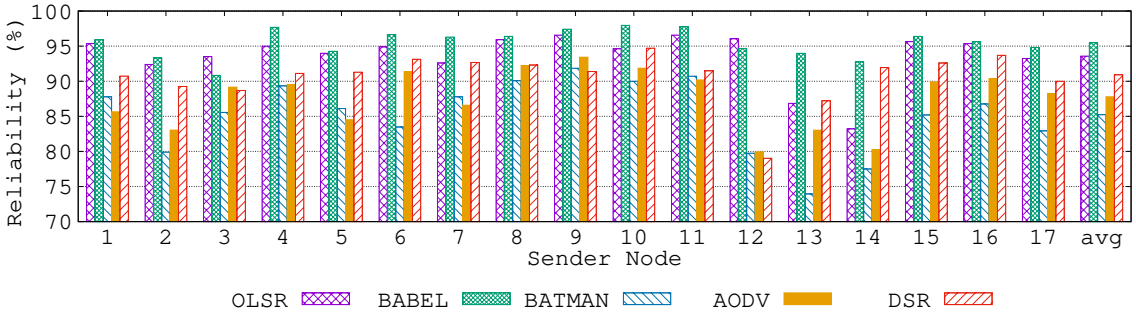
Regarding the length of the computed routes, all protocols converged roughly to the same routes being selected, with approximately 2.1 hops on average across all scenarios, since almost all protocols used the same cost metric (except BATMAN) and the diversity of available routes to compute in our network deployment was small. BABEL consistently computed the shortest max length routes, with an average of 5.5 hops across all scenarios. On the other hand, BATMAN computed the longest max length routes, with an average of 6.9 hops overall.

Figure 4.5 shows in each plot the results for the reliability, represented in the y axis, discriminated by node and on average (the last set of columns) represented in the x axis. Overall, all protocols achieved a reliability above 80% in all scenarios, with the proactive protocols (OLSR, BABEL, and BATMAN) achieving higher reliability than the reactive ones (AODV and DSR). This is explained by the nodes dropping requested messages while route computation is being performed and routes being constantly broken and re-computed due to unstable neighbors, whose impact is mitigated in proactive solutions since routes are continuously updated. In more stable deployments, allied with employing message buffering in the reactive protocols, it is expected that both types of protocols achieve similiar reliability results.

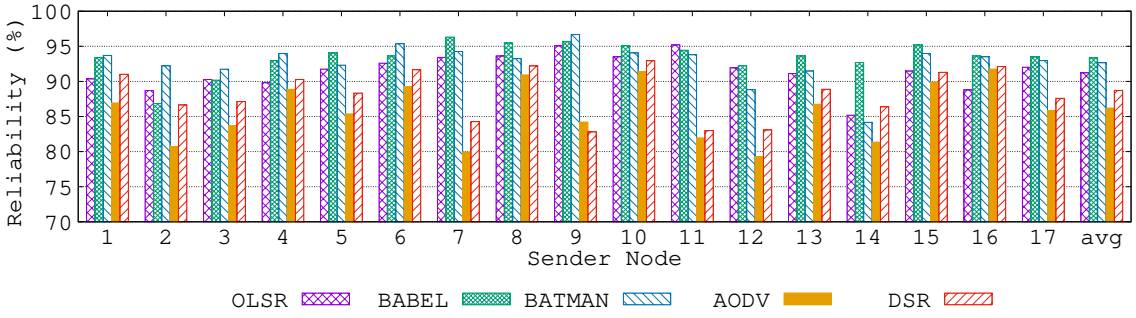
We note that, as the number of failures increases so does the reliability of the protocols. This is due to the fact that the resulting network after the faults had more stable paths, had less unstable redundant paths, and less interference between the nodes. Furthermore, in the scenario with two faults (Figure 4.5b), we note that BATMAN had significantly lower reliability when compared to the other scenarios (even worst than the reactive protocols). This was caused by the emergence of a high number of short-lived routing loops (46, 56, and 109 total loops in each execution in this scenario), as previously discussed. In addition, in the scenario with nine faults (Figure 4.5d), OLSR attained slightly inferior reliability results (90.9%) than AODV (91.2%) and DSR (91.9%). However, such a slight difference (below 1%) is most likely due to noise in the experimental environment



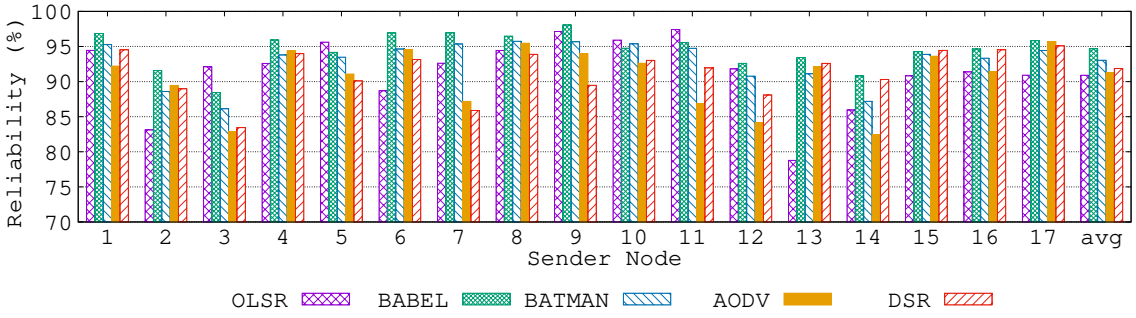
(a) No Faults



(b) Two Faults



(c) Five Faults



(d) Nine Faults

Figure 4.5: Reliability of Routing Protocols.

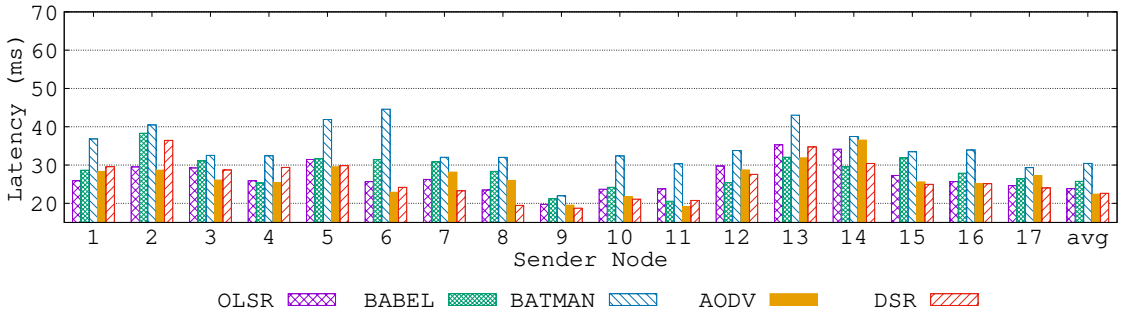
and cannot be directly attributed to OLSR's performance being worst than the reactive protocols, especially since the results are the average of only three executions and the resulting network after the node faults becomes extremely small.

BABEL was the protocol that achieved the highest reliability on average, in all scenarios. The main reason for this behavior is that, among the proactive protocols, BABEL was the one with the lowest overhead (discussed further ahead) and, as such, this lead to less interferences in the wireless medium causing less message losses.

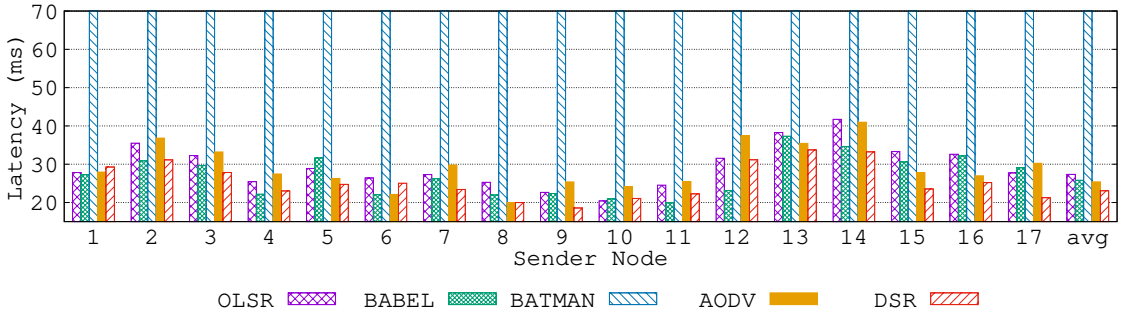
Among the reactive protocols, DSR achieved a slightly higher reliability than AODV in all scenarios. We suspect this behavior was caused by unstable neighborhood relations that induced the routes to break, leading the intermediary nodes in AODV to remove the routes from their routing tables. This instability impacted DSR less since the full routes are carried within the messages.

Figure 4.6 presents in each plot the average latency in milliseconds (ms) in the y axis, across all nodes and on average (the last set of columns), represented in the x axis. Overall, all protocols achieved a latency below 35 ms in all scenarios, with approximately the same average latency per scenario. The reason behind these results is that all the protocols converged to approximately the same routes being selected, as previously discussed. The exception was BATMAN, consistently being the protocol with the highest latency overall in all scenarios (except in the one with five faults), due to the formation of short-lived routing loops that were observed during the experiment across all scenarios. In addition, DSR attained the highest average latency in the scenario with five faults (Figure 4.6c) due to one of the executions presenting abnormal latency (54 ms), which biased the average of the three executions, being the reason for this behavior on that execution unknown (in that execution there were no loops; the average route length was 2.1 hops, similar to the other protocols; this was the only execution where this happened to this protocol, and thus it cannot be assigned to higher processing overhead to retrieve the next-hops from the message headers instead of consulting the routing table; and the communication overhead was similar in the three executions and was slightly lower than AODV, so the contention and collisions should have been similar for the two protocols or even slightly better to DSR). Understanding this particular behavior is delegated to future work.

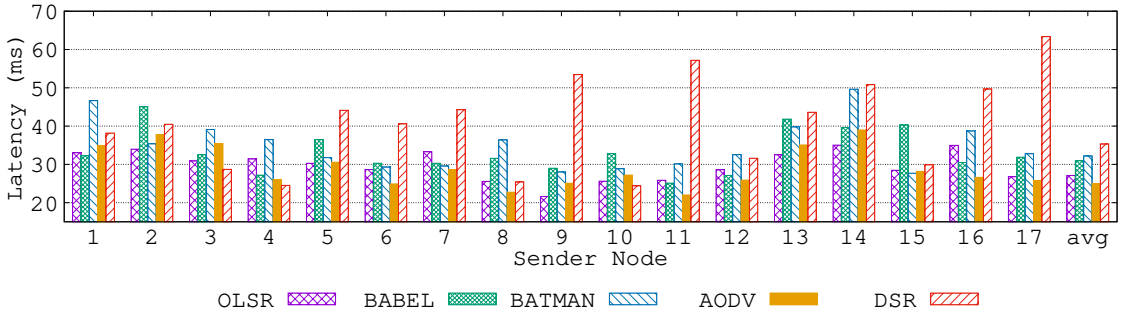
Figure 4.7 presents in each plot the results of the total communication overhead, represented in the y axis, for each protocol, represented in the x axis. The overhead is discriminated into three types: the *discovery overhead*, as being the number of transmissions incurred by the discovery protocol, the *broadcast overhead*, as the number of transmissions incurred by the broadcast protocol, and the *routing overhead*, as the number of transmissions incurred to disseminate control messages with unicast. We begin to note that, as the number of failures increases, the overhead decreases as fewer nodes disseminate control messages. Overall, OLSR presented the highest overhead since its routing strategy triggered the dissemination of unscheduled control messages whenever the selected sub-set of the topology to disseminate (with broadcast) changed, which frequently happened due to unstable neighborhood relations.



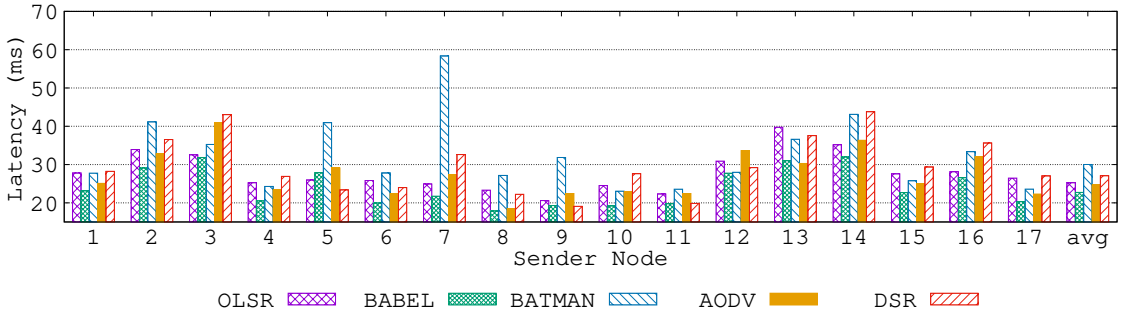
(a) No Faults



(b) Two Faults

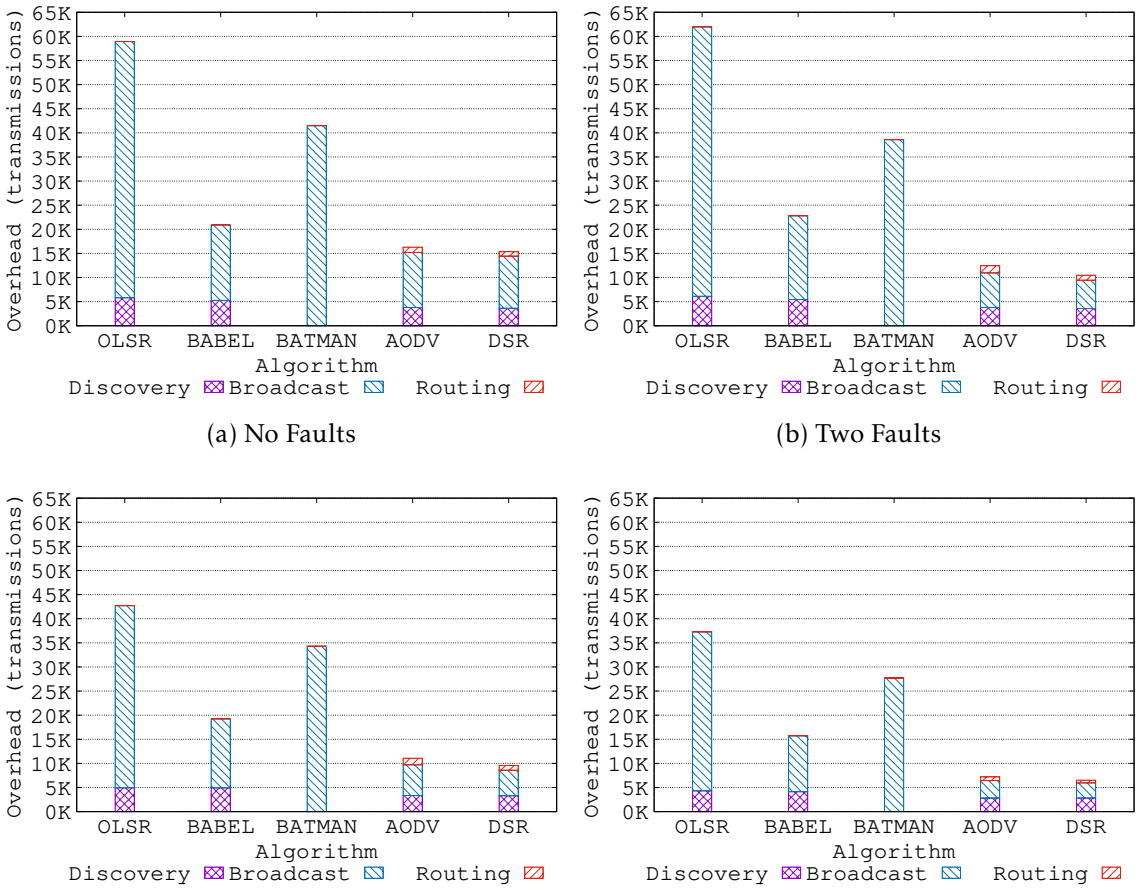


(c) Five Faults



(d) Nine Faults

Figure 4.6: Average Latency of Routing Protocols.



(a) No Faults

(b) Two Faults

(c) Five Faults

(d) Nine Faults

Figure 4.7: Total Communication Overhead per Routing Protocol.

The reactive routing protocols, AODV and DSR, presented the lowest overhead across all scenarios. This is the result of caching eavesdropped routes destined to other nodes which allows less route requests to be disseminated.

The BATMAN protocol presented the second highest overhead, which is fundamentally caused by the constant periodic broadcasting of a node's identity. In addition, BATMAN's neighbor discovery process is merged with the dissemination of such control messages, allowing to have no additional discovery overhead.

4.5 Summary

This chapter discussed the methodology and the results of a preliminary experimental evaluation of some representative solutions of neighbor discovery, broadcast, and routing. These solutions were implemented leveraging prototypes of our proposed frameworks and were evaluated in a real wireless *ad hoc* network formed by commodity devices.

Among all experiments, the neighborhoods detected where highly unstable and there were many message that were lost, which heavly affected the results. Furthermore, the

results obtained cannot be extrapolated to other topologies and configurations. Therefore, for these two reasons, more exhaustive evaluations should be carried out.

The next chapter contains the final remarks regarding the work conducted in this thesis and introduces directions for future work.

CHAPTER 5

FINAL REMARKS

This chapter contains the final remarks regarding the conclusions derived from the work presented in this thesis (§5.1) and introduces several directions for future work (§5.2).

5.1 Conclusions

In this thesis, we developed three conceptual frameworks for capturing the behavior of three essential services to support communication primitives for wireless *ad hoc* networks. These services correspond to neighbor discovery, broadcast, and routing. These frameworks have the objective to better compare and relate the particular strategies employed by each solution of these services. Each framework corresponds to a generic protocol that abstracts, for a broad range of solutions for each of these services, their main common elements and that can be parameterized to capture the behavior of particular instances of these different solutions.

Leveraging prototypes of our frameworks, we conducted a preliminary experimental evaluation of a set of representative solutions on a real wireless *ad hoc* network formed by commodity devices. This process showed the validity of our approach, while the results revealed intriguing insights that had not before been explored and addressed in the context of communication primitives wireless *ad hoc* networks.

Regarding neighbor discovery, as far as we know, this was the first experimental evaluation on the performance of different discovery protocols. The results revealed that the classic PDJ presented the best overall performance and that employing unscheduled transmissions greatly increases the cost of discovery solutions since the network topology tends to be highly dynamic even when there is no node mobility. The results also revealed that separating the transmission of hacks and hellos, either promptly replying or with different periodicities, does not significantly decrease the latency.

Regarding broadcast, our results showed that SBA was the best algorithm overall, despite unstable neighborhood relations, which highly influenced negatively MPR and AHBP, other neighbor-aware algorithms which employed policies that strived for higher cost reductions than SBA (which they could not achieve). Therefore, stabilizing neighbor discovery is remarkably important for the performance of neighbor-aware algorithms in real networks. Furthermore, probabilistic-based and count-based approaches did not perform well due to either behaving similarly to flooding in the majority of the nodes or their policies not taking into account the network's topology.

Regarding routing, the results showed that BATMAN, which is considered in the literature as one of the best routing protocols [35, 40–42, 44], was not only never the best regarding the reliability in any scenario but also was the worst regarding the latency in all scenarios due to the formation of short-lived routing loops. Furthermore, reactive protocols presented similar reliability to proactive ones despite the unstable neighborhoods and node faults, due to the usage of route caching, while having much lower overhead.

Unfortunately, across all experiments, the neighborhoods detected where highly unstable and there were many message losses, which heavly affected the results. In addition, these results cannot be extrapolated to other topologies and configurations, and thus, for these two reasons, more exhaustive evaluations should be carried out in the future.

Finally, the main outcome of this thesis is the creation of the three conceptual frameworks that showed to be effective in capturing the behavior of a wide range of solutions and enabling the quick implementation, prototyping, and evaluation of existing and novel solutions of the three key services that support communication primitives for wireless *ad hoc* networks, in the future.

5.2 Future Work

As future work, a more extensive experimental evaluation of the solutions presented in this thesis should be conducted across more complex and realistic scenarios, considering node mobility, a higher number of nodes, and more routing solutions.

In addition, it would also be appealing to evaluate the performance of routing protocols configured with several neighbor discovery and broadcast alternatives other than the ones contained in their specification to assess if better solutions can be derived.

Furthermore, another interesting direction for future work is to leverage our frameworks to dynamically select, individually for each node, the most suitable solutions and configurations according to its local network conditions. This process requires leveraging the results of a more exhaustive experimental evaluation under the most diverse conditions and determining which solutions are the most suitable for each situation. However, it is necessary to ensure that our frameworks are able to support the interoperation between several distinct solutions running on different nodes.

Bibliography

[1] A. Al-Fuqaha et al. "Internet of things: A survey on enabling technologies, protocols, and applications". In: *IEEE communications surveys & tutorials* 17.4 (2015), pp. 2347–2376.

[2] D. G. Reina et al. "The Role of Ad Hoc Networks in the Internet of Things: A Case Scenario for Smart Environments". In: *Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence*. Ed. by N. Bessis et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 89–113. ISBN: 978-3-642-34952-2. DOI: 10.1007/978-3-642-34952-2_4.

[3] Cisco. *Cisco Annual Internet Report (2018–2023)*. 2020. URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[4] L. D. Xu, W. He, and S. Li. "Internet of Things in Industries: A Survey". In: *IEEE Transactions on Industrial Informatics* 10.4 (2014), pp. 2233–2243. DOI: 10.1109/TII.2014.2300753.

[5] IEEE. *IEEE 802.11-2020 - IEEE Approved Draft Standard for Information Technology – Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks – Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. 2020.

[6] K. Pahlavan and P. Krishnamurthy. *Principles of wireless networks: A unified approach*. Prentice Hall PTR, 2011, pp. 2–4, 11–12, 40–46, 50–51, 58–61, 160, 179–201, 416–520.

[7] I. F. Akyildiz, D. M. Gutierrez-Estevez, and E. C. Reyes. "The evolution to 4G cellular systems: LTE-Advanced". In: *Physical Communication* 3.4 (2010), pp. 217–244. ISSN: 1874-4907. DOI: 10.1016/j.phycom.2010.08.001.

[8] M. Shafi et al. "5G: A Tutorial Overview of Standards, Trials, Challenges, Deployment, and Practice". In: *IEEE Journal on Selected Areas in Communications* 35.6 (2017), pp. 1201–1221. DOI: 10.1109/JSAC.2017.2692307.

[9] A. D. JoSEP et al. "A view of cloud computing". In: *Communications of the ACM* 53.4 (2010).

[10] T. Dillon, C. Wu, and E. Chang. "Cloud Computing: Issues and Challenges". In: *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. 2010, pp. 27–33. DOI: 10.1109/AINA.2010.187.

[11] J. Leitão et al. *Towards Enabling Novel Edge-Enabled Applications*. Tech. rep. Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2018. arXiv: 1805.06989. URL: https://dblp.org/rec/bib/journals/corr/abs-1805-06989.

[12] W. Shi et al. "Edge computing: Vision and challenges". In: *IEEE Internet of Things Journal* 3.5 (2016), pp. 637–646.

[13] L. M. Vaquero and L. Rodero-Merino. "Finding your way in the fog: Towards a comprehensive definition of fog computing". In: *ACM SIGCOMM Computer Communication Review* 44.5 (2014), pp. 27–32.

[14] R. Mahesa. *How cloud, fog, and mist computing can work together*. IBM. 2018. URL: https://developer.ibm.com/dwblog/2018/cloud-fog-mist-edge-computing-iot/ (visited on 12/02/2019).

[15] H. Miranda. "Gossip-based data distribution in mobile ad hoc networks". PhD thesis. Doctoral Thesis. Report number: DI-FCUL/TR-07-33., University of Lisbon, 2007.

[16] İ. Bekmezci, O. K. Sahingoz, and Ş. Temel. "Flying Ad-Hoc Networks (FANETs): A survey". In: *Ad Hoc Networks* 11.3 (2013), pp. 1254–1270. ISSN: 1570-8705. DOI: 10.1016/j.adhoc.2012.12.004.

[17] P. Á. Costa. "Practical Aggregation in the Edge". MA thesis. Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2018.

[18] P. Ruiz and P. Bouvry. "Survey on Broadcast Algorithms for Mobile Ad Hoc Networks". In: *ACM Comput. Surv.* 48.8 (July 2015). ISSN: 0360-0300. DOI: 10.1145/2786005.

[19] A. Boukerche et al. "Routing protocols in ad hoc networks: A survey". In: *Computer Networks* 55.13 (2011), pp. 3032–3080. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2011.05.010.

[20] D. N. Patel et al. "A survey of reactive routing protocols in MANET". In: *International Conference on Information Communication and Embedded Systems (ICICES2014)*. 2014, pp. 1–6. DOI: 10.1109/ICICES.2014.7033833.

[21] R. Guerraoui and L. Rodrigues. *Introduction to Reliable Distributed Programming*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 3540288457.

[22] J. L. Martins. *Fundamentos de Redes de Computadores – Ilustrados com base na Internet e nos Protocolos TCP/IP*. Portugal: NOVA.FCT Editorial, Jan. 2018, pp. 612–698. ISBN: 978-989-99528-5-0. URL: https://legatheaux.eu/cnfbook.html.

[23] A. Medina and S. Bohacek. "Performance modeling of neighbor discovery in proactive routing protocols". In: *Journal of Advanced Research* 2.3 (2011). Special issue on Mobile Ad-Hoc Wireless Networks, pp. 227–239. ISSN: 2090-1232. DOI: 10.1016/j.jare.2011.04.007.

[24] Z. Gu et al. "A Practical Neighbor Discovery Framework for Wireless Sensor Networks". In: *Sensors* 19.8 (2019). ISSN: 1424-8220. DOI: 10.3390/s19081887.

[25] K.-H. Kim and K. G. Shin. "On Accurate Measurement of Link Quality in Multi-Hop Wireless Mesh Networks". In: *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*. MobiCom '06. Los Angeles, CA, USA: Association for Computing Machinery, 2006, pp. 38–49. ISBN: 1595932860. DOI: 10.1145/1161089.1161095.

[26] J. Wu and F. Dai. "Broadcasting in ad hoc networks based on self-pruning". In: *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*. Vol. 3. Mar. 2003, 2240–2250 vol.3. DOI: 10.1109/INFCOM.2003.1209244.

[27] J. Wu and F. Dai. "A generic distributed broadcast scheme in ad hoc wireless networks". In: *IEEE Transactions on Computers* 53.10 (Oct. 2004), pp. 1343–1354. ISSN: 0018-9340. DOI: 10.1109/TC.2004.69.

[28] I. Stojmenovic. "A General Framework for Broadcasting in Static to Highly Mobile Wireless Ad hoc, Sensor, Robot and Vehicular Networks". In: *2012 IEEE 18th International Conference on Parallel and Distributed Systems*. 2012, pp. 8–13. DOI: 10.1109/ICPADS.2012.12.

[29] Z. J. Haas. "A new routing protocol for the reconfigurable wireless networks". In: *Proceedings of ICUPC 97 - 6th International Conference on Universal Personal Communications*. Vol. 2. Oct. 1997, 562–566 vol.2. DOI: 10.1109/ICUPC.1997.627227.

[30] Tao Lin, S. F. Midkiff, and J. S. Park. "A framework for wireless ad hoc routing protocols". In: *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.* Vol. 2. 2003, 1162–1167 vol.2. DOI: 10.1109/WCNC.2003.1200535.

[31] B. Williams and T. Camp. "Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks". In: *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing*. MobiHoc '02. Lausanne, Switzerland: Association for Computing Machinery, 2002, pp. 194–205. ISBN: 1581135017. DOI: 10.1145/513800.513825.

[32] S. R. Das et al. "Comparative performance evaluation of routing protocols for mobile, ad hoc networks". In: *Proceedings 7th International Conference on Computer Communications and Networks (Cat. No.98EX226)*. 1998, pp. 153–161. DOI: 10.1109/ICCCN.1998.998772.

[33] S. Baraković and J. Baraković. "Comparative performance evaluation of Mobile Ad Hoc routing protocols". In: *The 33rd International Convention MIPRO*. 2010, pp. 518–523.

[34] S. R. Das, R. Castañeda, and J. Yan. "Simulation-based performance evaluation of routing protocols for mobile ad hoc networks". In: *Mobile Networks and Applications* 5.3 (Sept. 2000), pp. 179–189. ISSN: 1572-8153. DOI: 10.1023/A:1019108612308.

[35] M. Reineri, C. Casetti, and C. Chiasserini. "Routing protocols for mesh networks with mobility support". In: *2009 6th International Symposium on Wireless Communication Systems*. Sept. 2009, pp. 71–75. DOI: 10.1109/ISWCS.2009.5285344.

[36] D. Cavin, Y. Sasson, and A. Schiper. "On the Accuracy of MANET Simulators". In: *Proceedings of the Second ACM International Workshop on Principles of Mobile Computing*. POMC '02. Toulouse, France: Association for Computing Machinery, 2002, pp. 38–43. ISBN: 1581135114. DOI: 10.1145/584490.584499.

[37] T. R. Andel and A. Yasinsac. "On the credibility of manet simulations". In: *Computer* 39.7 (2006), pp. 48–54. DOI: 10.1109/MC.2006.242.

[38] W. Kiess and M. Mauve. "A survey on real-world implementations of mobile ad-hoc networks". In: *Ad Hoc Networks* 5.3 (2007), pp. 324–339. ISSN: 1570-8705. DOI: 10.1016/j.adhoc.2005.12.003.

[39] E. Kulla et al. "Evaluation of a MANET Testbed in Outdoor Bridge Environment Using BATMAN Routing Protocol". In: *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*. 2012, pp. 384–390. DOI: 10.1109/AINA.2012.54.

[40] E. Kulla et al. "Impact of Source and Destination Movement on MANET Performance Considering BATMAN and AODV Protocols". In: *2010 International Conference on Broadband, Wireless Computing, Communication and Applications*. 2010, pp. 94–101. DOI: 10.1109/BWCCA.2010.54.

[41] D. Seither, A. König, and M. Hollick. "Routing performance of Wireless Mesh Networks: A practical evaluation of BATMAN advanced". In: *2011 IEEE 36th Conference on Local Computer Networks*. 2011, pp. 897–904. DOI: 10.1109/LCN.2011.6115569.

[42] M. S. Singh and V. Talasila. "A practical evaluation for routing performance of BATMAN-ADV and HWMN in a Wireless Mesh Network test-bed". In: *2015 International Conference on Smart Sensors and Systems (IC-SSS)*. 2015, pp. 1–6. DOI: 10.1109/SMARTSENS.2015.7873617.

[43] F. Zeiger et al. "Mobile Robot Teleoperation via Wireless Multihop Networks - Parameter Tuning of Protocols and Real World Application Scenarios". In: *Informatics in Control, Automation and Robotics: Selcted Papers from the International Conference on Informatics in Control, Automation and Robotics 2008*. Ed. by J. A. Cetto, J.-L. Ferrier, and J. Filipe. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 139–152. ISBN: 978-3-642-00271-7. DOI: 10.1007/978-3-642-00271-7_10.

[44] M. Abolhasan, B. Hagelstein, and J. C.-P. Wang. "Real-world performance of current proactive multi-hop mesh protocols". In: *2009 15th Asia-Pacific Conference on Communications*. 2009, pp. 44–47. DOI: 10.1109/APCC.2009.5375690.

[45] *GPS: The Global Positioning System*. Dec. 2, 2019. URL: https://www.gps.gov/.

[46] *Galileo*. Dec. 2, 2019. URL: https://www.gsa.europa.eu/european-gnss/galileo/galileo-european-global-satellite-based-navigation-system.

[47] A. Rosa, P. A. Costa, and J. Leitão. "Revisiting Broadcast Algorithms for Wireless Edge Networks". In: *38th IEEE International Symposium on Reliable Distributed Systems (SRDS 2019)*. Lyon, France, Oct. 2019.

[48] I. F. Akyildiz and Xudong Wang. "A survey on wireless mesh networks". In: *IEEE Communications Magazine* 43.9 (Sept. 2005), S23–S30. ISSN: 1558-1896. DOI: 10.1109/MCOM.2005.1509968.

[49] G. Araniti, I. Bisio, and M. De Sanctis. "Interplanetary Networks: Architectural Analysis, Technical Challenges and Solutions Overview". In: *2010 IEEE International Conference on Communications*. 2010, pp. 1–5. DOI: 10.1109/ICC.2010.5502491.

[50] I. Akyildiz et al. "Wireless sensor networks: a survey". In: *Computer Networks* 38.4 (2002), pp. 393–422. ISSN: 1389-1286. DOI: 10.1016/S1389-1286(01)00302-4.

[51] I. F. Akyildiz et al. "A survey on sensor networks". In: *IEEE Communications Magazine* 40.8 (Aug. 2002), pp. 102–114. ISSN: 1558-1896. DOI: 10.1109/MCOM.2002.1024422.

[52] J. Partan, J. Kurose, and B. N. Levine. "A Survey of Practical Issues in Underwater Networks". In: *SIGMOBILE Mob. Comput. Commun. Rev.* 11.4 (Oct. 2007), pp. 23–33. ISSN: 1559-1662. DOI: 10.1145/1347364.1347372.

[53] F. Li and Y. Wang. "Routing in vehicular ad hoc networks: A survey". In: *IEEE Vehicular Technology Magazine* 2.2 (June 2007), pp. 12–22. ISSN: 1556-6080. DOI: 10.1109/MVT.2007.912927.

[54] Y. Cao and Z. Sun. "Routing in Delay/Disruption Tolerant Networks: A Taxonomy, Survey and Challenges". In: *IEEE Communications Surveys Tutorials* 15.2 (Feb. 2013), pp. 654–677. ISSN: 2373-745X. DOI: 10.1109/SURV.2012.042512.00053.

[55] I. F. Akyildiz, W.-Y. Lee, and K. R. Chowdhury. "CRAHNs: Cognitive radio ad hoc networks". In: *Ad Hoc Networks* 7.5 (2009), pp. 810–836. ISSN: 1570-8705. DOI: https://doi.org/10.1016/j.adhoc.2009.01.001. URL: https://www.sciencedirect.com/science/article/pii/S157087050900002X.

[56] S. M. Trenkwalder et al. "SwarmCom: an infra-red-based mobile ad-hoc network for severely constrained robots". In: *Autonomous Robots* 44.1 (Jan. 2020), pp. 93–114. ISSN: 1573-7527. DOI: 10.1007/s10514-019-09873-0.

[57] D. Tsonev, S. Videv, and H. Haas. "Light fidelity (Li-Fi): towards all-optical networking". In: *Broadband Access Communication Technologies VIII*. Ed. by B. B. Dingel and K. Tsukamoto. Vol. 9007. International Society for Optics and Photonics. SPIE, 2014, pp. 1–10. DOI: 10.1117/12.2044649.

[58] M. Torrent-Moreno et al. "IEEE 802.11-based One-hop Broadcast Communications: Understanding Transmission Success and Failure Under Different Radio Propagation Environments". In: *Proceedings of the 9th ACM International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems*. MSWiM '06. Terromolinos, Spain: ACM, 2006, pp. 68–77. ISBN: 1-59593-477-4. DOI: 10.1145/1164717.1164731.

[59] Yihong Zhou and S. M. Nettles. "Balancing the hidden and exposed node problems with power control in CSMA/CA-based wireless networks". In: *IEEE Wireless Communications and Networking Conference, 2005*. Vol. 2. Mar. 2005, 683–688 Vol. 2. DOI: 10.1109/WCNC.2005.1424590.

[60] P. Karn et al. "MACA-a new channel access method for packet radio". In: *ARRL/CRRL Amateur radio 9th computer networking conference*. Vol. 140. London, Canada. 1990, pp. 134–140.

[61] H. Lundgren, E. Nordstrom, and C. Tschudin. "The Gray Zone Problem in IEEE 802.11b Based Ad Hoc Networks". In: *SIGMOBILE Mob. Comput. Commun. Rev.* 6.3 (June 2002), pp. 104–105. ISSN: 1559-1662. DOI: 10.1145/581291.581311.

[62] R. Aguirre et al. "The grey area in wireless communications: A multiplatform experimental study". In: *2014 IEEE Latin-America Conference on Communications (LATINCOM)*. 2014, pp. 1–6. DOI: 10.1109/LATINCOM.2014.7041867.

[63] I. D. Chakeres and E. M. Belding-Royer. "The utility of hello messages for determining link connectivity". In: *The 5th International Symposium on Wireless Personal Multimedia Communications*. Vol. 2. 2002, 504–508 vol.2. DOI: 10.1109/WPMC.2002.1088225.

[64] T. Senthil Kumaran and V. Sankaranarayanan. "Early congestion detection and adaptive routing in MANET". In: *Egyptian Informatics Journal* 12.3 (2011), pp. 165–175. ISSN: 1110-8665. DOI: 10.1016/j.eij.2011.09.001.

[65] I. Demirkol, C. Ersoy, and F. Alagoz. "MAC protocols for wireless sensor networks: a survey". In: *IEEE Communications Magazine* 44.4 (2006), pp. 115–121.

[66] H. Menouar, F. Filali, and M. Lenardi. "A survey and qualitative analysis of MAC protocols for vehicular ad hoc networks". In: *IEEE wireless communications* 13.5 (2006), pp. 30–35.

[67] R. Jurdak, C. V. Lopes, and P. Baldi. "A survey, classification and comparative analysis of medium access control protocols for ad hoc networks". In: *IEEE Communications Surveys & Tutorials* 6.1 (2004), pp. 2–16.

[68] J. Yeo, H. Lee, and S. Kim. "An efficient broadcast scheduling algorithm for TDMA ad-hoc networks". In: *Computers & operations research* 29.13 (2002), pp. 1793–1806.

[69] I. Rhee et al. "DRAND: Distributed Randomized TDMA Scheduling for Wireless Ad Hoc Networks". In: *IEEE Transactions on Mobile Computing* 8.10 (Oct. 2009), pp. 1384–1396. ISSN: 2161-9875. DOI: 10.1109/TMC.2009.59.

[70] C. Zhu and M. S. Corson. "A five-phase reservation protocol (FPRP) for mobile ad hoc networks". In: *Wireless networks* 7.4 (2001), pp. 371–384.

[71] W. Crowther et al. "A system for broadcast communication: Reservation-ALOHA". In: *Proc. 6th Hawaii Int. Conf. Syst. Sci.* 1973, pp. 596–603.

[72] F. Borgonovo et al. "ADHOC MAC: New MAC Architecture for Ad Hoc Networks Providing Efficient and Reliable Point-to-Point and Broadcast Services". In: *Wireless Networks* 10.4 (July 2004), pp. 359–366. ISSN: 1572-8196. DOI: 10.1023/B:WINE.0000028540.96160.8a.

[73] X. Wang and K. Kar. "Throughput modelling and fairness issues in CSMA/CA based ad-hoc networks". In: *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.* Vol. 1. IEEE. 2005, pp. 23–34.

[74] K. Xu, M. Gerla, and S. Bae. "Effectiveness of RTS/CTS handshake in IEEE 802.11 based ad hoc networks". In: *Ad Hoc Networks* 1.1 (2003), pp. 107–123. ISSN: 1570-8705. DOI: 10.1016/S1570-8705(03)00015-5.

[75] K. Tang and M. Gerla. "MAC reliable broadcast in ad hoc networks". In: *2001 MILCOM Proceedings Communications for Network-Centric Operations: Creating the Information Force (Cat. No.01CH37277)*. Vol. 2. Oct. 2001, pp. 1008–1013. DOI: 10.1109/MILCOM.2001.985991.

[76] S. Li, L. Da Xu, and S. Zhao. "The internet of things: a survey". In: *Information Systems Frontiers* 17.2 (2015), pp. 243–259.

[77] G. R. Hiertz et al. "The IEEE 802.11 universe". In: *IEEE Communications Magazine* 48.1 (Jan. 2010), pp. 62–70. ISSN: 1558-1896. DOI: 10.1109/MCOM.2010.5394032.

[78] G. Hiertz et al. "IEEE 802.11s: the WLAN mesh standard". In: *Wireless Communications, IEEE* 17 (Mar. 2010), pp. 104–111. DOI: 10.1109/MWC.2010.5416357.

[79] M. Collotta et al. "Bluetooth 5: A Concrete Step Forward toward the IoT". In: *IEEE Communications Magazine* 56.7 (July 2018), pp. 125–131. ISSN: 1558-1896. DOI: 10.1109/MCOM.2018.1700053.

[80] M. Collotta et al. "Bluetooth 5: A Concrete Step Forward toward the IoT". In: *IEEE Communications Magazine* 56.7 (2018), pp. 125–131. DOI: 10.1109/MCOM.2018.1700053.

[81] M. Petrova et al. "Performance study of IEEE 802.15.4 using measurements and simulations". In: *IEEE Wireless Communications and Networking Conference, 2006. WCNC 2006.* Vol. 1. 2006, pp. 487–492. DOI: 10.1109/WCNC.2006.1683512.

[82] S. Safaric and K. Malaric. "ZigBee wireless standard". In: *Proceedings ELMAR 2006*. June 2006, pp. 259–262. DOI: 10.1109/ELMAR.2006.329562.

[83] G. Mulligan. "The 6LoWPAN Architecture". In: *Proceedings of the 4th Workshop on Embedded Networked Sensors*. EmNets '07. Cork, Ireland: Association for Computing Machinery, 2007, pp. 78–82. ISBN: 9781595936943. DOI: 10.1145/1278972.1278992.

[84] I. Tinnirello et al. "Wireless MAC processors: Programming MAC protocols on commodity Hardware". In: *2012 Proceedings IEEE INFOCOM*. 2012, pp. 1269–1277. DOI: 10.1109/INFCOM.2012.6195488.

[85] N. Ahmed et al. "A Survey of COVID-19 Contact Tracing Apps". In: *IEEE Access* 8 (2020), pp. 134577–134601. DOI: 10.1109/ACCESS.2020.3010226.

[86] T. H. Clausen, C. Dearlove, and J. Dean. *Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)*. RFC 6130. Apr. 2011. DOI: 10.17487/RFC6130.

[87] J. Chroboczek and D. Schinazi. *The Babel Routing Protocol*. RFC 8966. Jan. 2021. DOI: 10.17487/RFC8966.

[88] E. B. Hamida, G. Chelius, and E. Fleury. "Revisiting Neighbor Discovery with Interferences Consideration". In: *Proceedings of the 3rd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor and Ubiquitous Networks*. PE-WASUN '06. Terromolinos, Spain: Association for Computing Machinery, 2006, pp. 74–81. ISBN: 1595934871. DOI: 10.1145/1163610.1163623.

[89] T. H. Clausen et al. *The Optimized Link State Routing Protocol Version 2*. RFC 7181. Apr. 2014. DOI: 10.17487/RFC7181.

[90] Min-Te Sun, Wuchi Feng, and Ten-Hwang Lai. "Location aided broadcast in wireless ad hoc networks". In: *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No.01CH37270)*. Vol. 5. 2001, 2842–2846 vol.5. DOI: 10.1109/GLOCOM.2001.965948.

[91] V. C. Giruka and M. Singhal. "Hello protocols for ad-hoc networks: overhead and accuracy tradeoffs". In: *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*. 2005, pp. 354–361. DOI: 10.1109/WOWMOM.2005.50.

[92] V. Ramasubramanian, R. Chandra, and D. Mosse. "Providing a bidirectional abstraction for unidirectional ad hoc networks". In: *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 3. 2002, 1258–1267 vol.3. DOI: 10.1109/INFCOM.2002.1019376.

[93] B. Karp and H. T. Kung. "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks". In: *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*. MobiCom '00. Boston, Massachusetts, USA: Association for Computing Machinery, 2000, pp. 243–254. ISBN: 1581131976. DOI: 10.1145/345910.345953.

[94] C. E. Perkins et al. *Ad Hoc On-demand Distance Vector Version 2 (AODVv2) Routing*. Internet-Draft draft-perkins-manet-aodvv2-03. Work in Progress. Internet Engineering Task Force, Feb. 2019. 90 pp.

[95] J. Sucec and I. Marsic. *An efficient distributed network-wide broadcast algorithm for mobile ad hoc networks*. Tech. rep. Citeseer, 2000.

[96] W. Peng and X. Lu. "AHBP: An efficient broadcast protocol for mobile Ad hoc networks". In: *Journal of Computer Science and Technology* 16.2 (Mar. 2001), pp. 114–125. ISSN: 1860-4749. DOI: 10.1007/BF02950416.

[97] K.-W. Chin et al. "Implementation Experience with MANET Routing Protocols". In: *SIGCOMM Comput. Commun. Rev.* 32.5 (Nov. 2002), pp. 49–59. ISSN: 0146-4833. DOI: 10.1145/774749.774758.

[98] A. Neumann et al. *Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.)* Internet-Draft draft-wunderlich-openmesh-manet-routing-00. Work in Progress. Internet Engineering Task Force, Apr. 2008. 24 pp.

[99] A. Woo, T. Tong, and D. Culler. "Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks". In: *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*. SenSys '03. Los Angeles, California, USA: Association for Computing Machinery, 2003, pp. 14–27. ISBN: 1581137079. DOI: 10.1145/958491.958494.

[100] F. L. Templin, R. Ogier, and M. S. Lewis. *Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)*. RFC 3684. Feb. 2004. DOI: 10.17487/RFC3684.

[101] V. C. Giruka and M. Singhal. "Hello protocols for ad-hoc networks: overhead and accuracy tradeoffs". In: *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*. 2005, pp. 354–361. DOI: 10.1109/WOWMOM.2005.50.

[102] Y.-C. Hu, D. A. Maltz, and D. B. Johnson. *The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4*. RFC 4728. Feb. 2007. DOI: 10.17487/RFC4728. URL: https://rfc-editor.org/rfc/rfc4728.txt.

[103] T. H. Clausen, C. Dearlove, and B. Adamson. *Jitter Considerations in Mobile Ad Hoc Networks (MANETs)*. RFC 5148. Feb. 2008. DOI: 10.17487/RFC5148.

[104] W. Sun et al. "Energy-Efficient Neighbor Discovery in Mobile Ad Hoc and Wireless Sensor Networks: A Survey". In: *IEEE Communications Surveys Tutorials* 16.3 (2014), pp. 1448–1459. DOI: 10.1109/SURV.2013.012414.00164.

[105] O. A. Saraereh, I. Khan, and B. M. Lee. "An Efficient Neighbor Discovery Scheme for Mobile WSN". In: *IEEE Access* 7 (2019), pp. 4843–4855. DOI: 10.1109/ACCESS.2018.2886779.

[106] F. Ingelrest, N. Mitton, and D. Simplot-Ryl. "A Turnover based Adaptive HELLO Protocol for Mobile Ad Hoc and Sensor Networks". In: *2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. 2007, pp. 9–14. DOI: 10.1109/MASCOTS.2007.5.

[107] Y.-C. Tseng et al. "The broadcast storm problem in a mobile ad hoc network". In: *Wireless networks* 8.2-3 (2002), pp. 153–167.

[108] B. Williams and T. Camp. "Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks". In: *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing*. MobiHoc '02. Lausanne, Switzerland: Association for Computing Machinery, 2002, pp. 194–205. ISBN: 1581135017. DOI: 10.1145/513800.513825.

[109] K. Obraczka, K. Viswanath, and G. Tsudik. "Flooding for Reliable Multicast in Multi-Hop Ad Hoc Networks". In: *Wireless Networks* 7.6 (Nov. 2001), pp. 627–634. ISSN: 1572-8196. DOI: 10.1023/A:1012323519059.

[110] Y.-B. Ko and N. H. Vaidya. "Flooding-Based Geocasting Protocols for Mobile Ad Hoc Networks". In: *Mobile Networks and Applications* 7.6 (Dec. 2002), pp. 471–480. ISSN: 1572-8153. DOI: 10.1023/A:1020712802004.

[111] V. Dimakopoulos and E. Pitoura. "On the Performance of Flooding-Based Resource Discovery". In: *IEEE Transactions on Parallel and Distributed Systems* 17.11 (2006), pp. 1242–1252. DOI: 10.1109/TPDS.2006.161.

[112] H. Miranda et al. "An Algorithm for Dissemination and Retrieval of Information in Wireless Ad Hoc Networks". In: *Euro-Par 2007 Parallel Processing*. Ed. by A.-M. Kermarrec, L. Bougé, and T. Priol. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 891–900. ISBN: 978-3-540-74466-5.

[113] P. Verissimo and L. Rodrigues. *Distributed Systems for System Architects*. Norwell, MA, USA: Kluwer Academic Publishers, 2001. ISBN: 0792372662.

[114] P. Samar, M. R. Pearlman, and Z. J. Haas. "Independent zone routing: an adaptive hybrid routing framework for ad hoc wireless networks". In: *IEEE/ACM Transactions on Networking* 12.4 (2004), pp. 595–608. DOI: 10.1109/TNET.2004.833153.

[115] C. A. Santiváñez, R. Ramanathan, and I. Stavrakakis. "Making Link-State Routing Scale for Ad Hoc Networks". In: *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing*. MobiHoc '01. Long Beach, CA, USA: Association for Computing Machinery, 2001, pp. 22–32. ISBN: 1581134282. DOI: 10.1145/501417.501420.

[116] I. Park, J. Kim, and I. Pu. "Blocking Expanding Ring Search Algorithm for Efficient Energy Consumption in Mobile Ad Hoc Networks". In: *WONS 2006 : Third Annual Conference on Wireless On-demand Network Systems and Services*. http://citi.insa-lyon.fr/wons2006/index.html. INRIA, INSA Lyon, IFIP, Alcatel. Les Ménuires (France), Jan. 2006, pp. 191–195.

[117] M. A. Al-Rodhaan, L. Mackenzie, and M. Ould-Khaoua. "Improvement to blocking expanding ring search for manets". In: *Dept of Computing Science, University of Glasgow, Glasgow, UK* 286.1 (2008), pp. 1–13.

[118] I. M. Pu and Y. Shen. "Enhanced Blocking Expanding Ring Search in Mobile Ad Hoc Networks". In: *2009 3rd International Conference on New Technologies, Mobility and Security*. Dec. 2009, pp. 1–5. DOI: 10.1109/NTMS.2009.5384676.

[119] A. H. Shintre and S. Sondur. "Improved Blocking Expanding Ring Search (I-BERS) protocol for energy efficient routing in MANET". In: *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*. May 2014, pp. 1–6. DOI: 10.1109/ICRAIE.2014.6909257.

[120] I. M. Pu, D. Stamate, and Y. Shen. "Improving time-efficiency in blocking expanding ring search for mobile ad hoc networks". In: *Journal of Discrete Algorithms* 24 (2014). LSD/LAW 2012, pp. 59–67. ISSN: 1570-8667. DOI: 10.1016/j.jda.2013.03.006.

[121] R. Lima, C. Baquero, and H. Miranda. "Broadcast Cancellation in Search Mechanisms". In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. SAC '13. Coimbra, Portugal: Association for Computing Machinery, 2013, pp. 548–553. ISBN: 9781450316569. DOI: 10.1145/2480362.2480467.

[122] T. J. Kwon and M. Gerla. "Efficient Flooding with Passive Clustering (PC) in Ad Hoc Networks". In: *SIGCOMM Comput. Commun. Rev.* 32.1 (Jan. 2002), pp. 44–56. ISSN: 0146-4833. DOI: 10.1145/510726.510730.

[123] C. Ellis, H. Miranda, and F. Taïani. "Count on Me: Lightweight Ad-Hoc Broadcasting in Heterogeneous Topologies". In: *Proceedings of the International Workshop on Middleware for Pervasive Mobile and Embedded Computing*. M-PAC '09. Urbana Champaign, Illinois: Association for Computing Machinery, 2009. ISBN: 9781605588490. DOI: 10.1145/1657127.1657129.

[124] J. Leitão. "Gossip-Based Broadcast Protocols". MA thesis. Faculdade de Ciências da Universidade de Lisboa, May 2007.

[125] Shintaro Izumi et al. "Hop count aware broadcast algorithm with Random Assessment Delay extension for wireless sensor networks". In: *2008 7th Asia-Pacific Symposium on Information and Telecommunication Technologies*. 2008, pp. 207–212. DOI: 10.1109/APSITT.2008.4653566.

[126] M. D. Colagrosso. "Intelligent Broadcasting in Mobile Ad Hoc Networks: Three Classes of Adaptive Protocols". In: *EURASIP Journal on Wireless Communications and Networking* 2007.1 (Nov. 2006), p. 010216. ISSN: 1687-1499. DOI: 10.1155/2007/10216.

[127] A. Rahman, P. Gburzynski, and B. Kaminska. "Enhanced Dominant Pruning-based Broadcasting in Untrusted Ad-hoc Wireless Networks". In: *2007 IEEE International Conference on Communications*. June 2007, pp. 3389–3394. DOI: 10.1109/ICC.2007.561.

[128] W. Peng and X.-C. Lu. "On the Reduction of Broadcast Redundancy in Mobile Ad Hoc Networks". In: *Proceedings of the 1st ACM International Symposium on Mobile Ad Hoc Networking & Computing*. MobiHoc '00. Boston, Massachusetts: IEEE Press, 2000, pp. 129–130. ISBN: 0780365348.

[129] D. Lima and H. Miranda. "Flow-Aware Broadcasting Algorithm". In: *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*. June 2012, pp. 1601–1608. DOI: 10.1109/TrustCom.2012.162.

[130] Z. J. Haas, J. Y. Halpern, and L. Li. "Gossip-based ad hoc routing". In: *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 3. June 2002, 1707–1716 vol.3. DOI: 10.1109/INFCOM.2002.1019424.

[131] Y. Sasson, D. Cavin, and A. Schiper. "Probabilistic broadcast for flooding in wireless mobile ad hoc networks". In: *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.* Vol. 2. Mar. 2003, 1124–1130 vol.2. DOI: 10.1109/WCNC.2003.1200529.

[132] Qi Zhang and D. P. Agrawal. "Dynamic probabilistic broadcasting in mobile ad hoc networks". In: *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No.03CH37484)*. Vol. 5. Oct. 2003, 2860–2864 Vol.5. DOI: 10.1109/VETECF.2003.1286132.

[133] F. Nourazar and M. Sabaei. "DAPF: An Efficient Flooding Algorithm for Mobile Ad-hoc Networks". In: *2009 International Conference on Signal Processing Systems*. 2009, pp. 594–598. DOI: 10.1109/ICSPS.2009.112.

[134] H. Miranda et al. "A Power-Aware Broadcasting Algorithm". In: *2006 IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications*. Sept. 2006, pp. 1–5. DOI: 10.1109/PIMRC.2006.254191.

[135] C. Winstanley et al. "PAMPA in the wild: a real-life evaluation of a lightweight ad-hoc broadcasting family". In: *Journal of Internet Services and Applications* 5.1 (Apr. 2014), p. 5. ISSN: 1869-0238. DOI: 10.1186/1869-0238-5-5.

[136] M. B. Yassein, S. F. Nimer, and A. Y. Al-Dubai. "A new dynamic counter-based broadcasting scheme for Mobile Ad hoc Networks". In: *Simulation Modelling Practice and Theory* 19.1 (2011). Modeling and Performance Analysis of Networking and Collaborative Systems, pp. 553–563. ISSN: 1569-190X. DOI: 10.1016/j.simpat.2010.08.011.

[137] Shintaro Izumi et al. "Hop count aware broadcast algorithm with Random Assessment Delay extension for wireless sensor networks". In: *2008 7th Asia-Pacific Symposium on Information and Telecommunication Technologies*. Apr. 2008, pp. 207–212. DOI: 10.1109/APSITT.2008.4653566.

[138] Q. Huang, Y. Bai, and L. Chen. "Efficient Lightweight Broadcasting Protocols for Multi-Hop Ad Hoc Networks". In: *2006 IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications*. Sept. 2006, pp. 1–5. DOI: 10.1109/PIMRC.2006.254169.

[139] D. Liarokapis, A. Shahrabi, and A. Komninos. "DibA: An Adaptive Broadcasting Scheme in Mobile Ad Hoc Networks". In: *2009 Seventh Annual Communication Networks and Services Research Conference*. 2009, pp. 224–231. DOI: 10.1109/CNSR.2009.42.

[140] J. Cartigny, D. Simplot, and I. Stojmenovic. "Localized minimum-energy broadcasting in ad-hoc networks". In: *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*. Vol. 3. Mar. 2003, 2210–2217 vol.3. DOI: 10.1109/INFCOM.2003.1209241.

[141] M.-T. Sun, W. Feng, and T.-H. Lai. "Location aided broadcast in wireless ad hoc networks". In: *Global Telecommunications Conference, 2001. GLOBECOM'01. IEEE*. Vol. 5. IEEE. 2001, pp. 2842–2846.

[142] Yu-Chee Tseng, Sze-Yao Ni, and En-Yu Shih. "Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network". In: *IEEE Transactions on Computers* 52.5 (May 2003), pp. 545–557. ISSN: 2326-3814. DOI: 10.1109/TC.2003.1197122.

[143] S. Singh, C. Raghavendra, and J. Stepanek. "Power-aware broadcasting in mobile ad hoc networks". In: *Proceedings of IEEE PIMRC*. Vol. 99. 1999, pp. 22–31.

[144] Maggie Xiaoyan Cheng et al. "Energy-efficient broadcast and multicast routing in ad hoc wireless networks". In: *Conference Proceedings of the 2003 IEEE International Performance, Computing, and Communications Conference, 2003.* Apr. 2003, pp. 87–94. DOI: 10.1109/PCCC.2003.1203687.

[145] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. "Algorithms for Energy-Efficient Multicasting in Static Ad Hoc Wireless Networks". In: *Mobile Networks and Applications* 6.3 (2001), pp. 251–263. ISSN: 1572-8153. DOI: 10.1023/A:1011478717164.

[146] A. Banerjee and S. Shosh. "Energy-Efficient Broadcasting of Route-Request Packets (E2BR2) in Ad Hoc Networks". In: *Smart Network Inspired Paradigm and Approaches in IoT Applications*. Ed. by M. Elhoseny and A. K. Singh. Singapore: Springer Singapore, 2019, pp. 25–45. DOI: 10.1007/978-981-13-8614-5_3.

[147] A. Qayyum, L. Viennot, and A. Laouiti. "Multipoint relaying for flooding broadcast messages in mobile wireless networks". In: *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*. Jan. 2002, pp. 3866–3875. DOI: 10.1109/HICSS.2002.994521.

[148] H. Lim and C. Kim. "Multicast Tree Construction and Flooding in Wireless Ad Hoc Networks". In: *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. MSWIM '00. Boston, Massachusetts, USA: Association for Computing Machinery, 2000, pp. 61–68. ISBN: 1581133049. DOI: 10.1145/346855.346865.

[149] Wei Lou and Jie Wu. "On reducing broadcast redundancy in ad hoc wireless networks". In: *IEEE Transactions on Mobile Computing* 1.2 (2002), pp. 111–122. DOI: 10.1109/TMC.2002.1038347.

[150] G. Calinescu et al. "Selecting Forwarding Neighbors in Wireless Ad Hoc Networks". In: *Mobile Networks and Applications* 9.2 (2004), pp. 101–111. ISSN: 1572-8153. DOI: 10.1023/B:MONE.0000013622.63511.57.

[151] M. Akter, A. Islam, and A. Rahman. "Fault tolerant optimized broadcast for wireless Ad-Hoc networks". In: *2016 International Conference on Networking Systems and Security (NSysS)*. Jan. 2016, pp. 1–9. DOI: 10.1109/NSysS.2016.7400690.

[152] Wei Lou and Jie Wu. "Double-covered broadcast (DCB): a simple reliable broadcast algorithm in MANETs". In: *IEEE INFOCOM 2004*. Vol. 3. Mar. 2004, 2084–2095 vol.3. DOI: 10.1109/INFCOM.2004.1354616.

[153] H. Chizari, M. Hosseini, and S. Abd Razak. "MultiPoint Relay selection using GA". In: *2009 IEEE Symposium on Industrial Electronics Applications*. Vol. 2. Oct. 2009, pp. 957–962. DOI: 10.1109/ISIEA.2009.5356301.

[154] A. Singh and W. N. Bhukya. "An Evolutionary Approach to Multi-point Relays Selection in Mobile Ad Hoc Networks". In: *Pattern Recognition and Machine Intelligence*. Ed. by B. Deka et al. Cham: Springer International Publishing, 2019, pp. 375–384. ISBN: 978-3-030-34869-4.

[155] H. Chizari et al. "EF-MPR, a new energy eFficient multi-point relay selection algorithm for MANET". In: *The Journal of Supercomputing* 59.2 (2012), pp. 744–761. ISSN: 1573-0484. DOI: 10.1007/s11227-010-0470-7.

[156] A. Mchergui et al. "A survey and comparative study of QoS aware broadcasting techniques in VANET". In: *Telecommunication Systems* 66.2 (Oct. 2017), pp. 253–281. ISSN: 1572-9451. DOI: 10.1007/s11235-017-0280-9.

[157] R. Carvajal Gomez et al. "Emergent Overlays for Adaptive MANET Broadcast". In: *38th IEEE International Symposium on Reliable Distributed Systems (SRDS 2019)*. Lyon, France, Oct. 2019.

[158] F. Forero, N. M. Peña, and N. L. S. da Fonseca. "Latency Reduction in Probabilistic Broadcast Protocols for Ad Hoc Networks". In: *IEEE Wireless Communications Letters* 8.4 (2019), pp. 1268–1271. DOI: 10.1109/LWC.2019.2915077.

[159] C.-K. Toh. *Long-lived Ad Hoc Routing based on the Concept of Associativity*. Internet-Draft draft-ietf-manet-longlived-adhoc-routing-00. Work in Progress. Internet Engineering Task Force, Mar. 1999. 18 pp. URL: https://datatracker.ietf.org/doc/html/draft-ietf-manet-longlived-adhoc-routing-00.

[160] G. He. "Destination-Sequenced Distance Vector (DSDV) Protocol". In: *Networking Laboratory, Helsinki University of Technology* (2002), pp. 1–9.

[161] R. Prakash. "Unidirectional Links Prove Costly in Wireless Ad Hoc Networks". In: *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*. DIALM '99. Seattle, Washington, USA: Association for Computing Machinery, 1999, pp. 15–22. ISBN: 1581131747. DOI: 10.1145/313239.313263. URL: https://doi.org/10.1145/313239.313263.

[162] S. Nesargi and R. Prakash. "A tunneling approach to routing with unidirectional links in mobile ad-hoc networks". In: *Proceedings Ninth International Conference on Computer Communications and Networks (Cat.No.00EX440)*. 2000, pp. 522–527. DOI: 10.1109/ICCCN.2000.885539.

[163] V. Ramasubramanian, R. Chandra, and D. Mosse. "Providing a bidirectional abstraction for unidirectional ad hoc networks". In: *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 3. 2002, 1258–1267 vol.3. DOI: 10.1109/INFCOM.2002.1019376.

[164] V. Ramasubramanian and D. Mosse. "BRA: A Bidirectional Routing Abstraction for Asymmetric Mobile Ad Hoc Networks". In: *IEEE/ACM Transactions on Networking* 16.1 (2008), pp. 116–129. DOI: 10.1109/TNET.2007.912064.

[165] N. Javaid et al. "Performance study of ETX based wireless routing metrics". In: *2009 2nd International Conference on Computer, Control and Communication*. 2009, pp. 1–7. DOI: 10.1109/IC4.2009.4909163.

[166] R. Draves, J. Padhye, and B. Zill. "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks". In: *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*. MobiCom '04. Philadelphia, PA, USA: Association for Computing Machinery, 2004, pp. 114–128. ISBN: 1581138687. DOI: 10.1145/1023720.1023732.

[167] S. .-. M. Woo and S. Singh. "Longest life routing protocol (LLRP) for ad hoc networks with highly mobile nodes". In: *2000 IEEE Wireless Communications and Networking Conference. Conference Record (Cat. No.00TH8540)*. Vol. 3. 2000, 1306–1310 vol.3. DOI: 10.1109/WCNC.2000.904821.

[168] R. Dube et al. "Signal stability-based adaptive routing (SSA) for ad hoc mobile networks". In: *IEEE Personal Communications* 4.1 (1997), pp. 36–45. DOI: 10.1109/98.575990.

[169] S. .-. Lee and M. Gerla. "Dynamic load-aware routing in ad hoc networks". In: *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*. Vol. 10. 2001, 3206–3210 vol.10. DOI: 10.1109/ICC.2001.937263.

[170] R. C. Shah and J. M. Rabaey. "Energy aware routing for low energy ad hoc sensor networks". In: *2002 IEEE Wireless Communications and Networking Conference Record. WCNC 2002 (Cat. No.02TH8609)*. Vol. 1. 2002, 350–355 vol.1. DOI: 10.1109/WCNC.2002.993520.

[171] S. Mueller, R. P. Tsang, and D. Ghosal. "Multipath Routing in Mobile Ad Hoc Networks: Issues and Challenges". In: *Performance Tools and Applications to Networked Systems*. Ed. by M. C. Calzarossa and E. Gelenbe. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 209–234. ISBN: 978-3-540-24663-3.

[172] Y. Faheem and J. L. Rougier. "Loop avoidance for Fish-Eye OLSR in sparse wireless mesh networks". In: *2009 Sixth International Conference on Wireless On-Demand Network Systems and Services*. 2009, pp. 231–234. DOI: 10.1109/WONS.2009.4801855.

[173] V. D. Park and D. S. M. Corson. *Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification*. Internet-Draft draft-ietf-manet-tora-spec-04. Work in Progress. Internet Engineering Task Force, July 2001. 23 pp.

[174] M. Duke et al. *A Roadmap for Transmission Control Protocol (TCP) Specification Documents*. RFC 7414. Feb. 2015. DOI: 10.17487/RFC7414. URL: https://rfc-editor.org/rfc/rfc7414.txt.

[175] A. Boukerche and A. Darehshoorzadeh. "Opportunistic Routing in Wireless Networks: Models, Algorithms, and Classifications". In: *ACM Comput. Surv.* 47.2 (Nov. 2014). ISSN: 0360-0300. DOI: 10.1145/2635675.

[176] R. Sanchez-Iborra and M. Cano. "JOKER: A Novel Opportunistic Routing Protocol". In: *IEEE Journal on Selected Areas in Communications* 34.5 (May 2016), pp. 1690–1703. ISSN: 1558-0008. DOI: 10.1109/JSAC.2016.2545439.

[177] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer. "SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks". In: *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*. MobiHoc '03. Annapolis, Maryland, USA: Association for Computing Machinery, 2003, pp. 303–314. ISBN: 1581136846. DOI: 10.1145/778415.778450.

[178] Guangyu Pei, M. Gerla, and Tsu-Wei Chen. "Fisheye state routing: a routing scheme for ad hoc wireless networks". In: *2000 IEEE International Conference on Communications. ICC 2000. Global Convergence Through Communications. Conference Record*. Vol. 1. June 2000, 70–74 vol.1. DOI: 10.1109/ICC.2000.853066.

[179] A. Iwata et al. "Scalable routing strategies for ad hoc wireless networks". In: *IEEE Journal on Selected Areas in Communications* 17.8 (Aug. 1999), pp. 1369–1379. ISSN: 1558-0008. DOI: 10.1109/49.779920.

[180] T. Fuhrmann et al. *Pushing chord into the underlay: Scalable routing for hybrid manets*. Tech. rep. Accessed: 21-01-2020. Germany: Technical University of Karlsruhe (TH), June 2006.

[181] Rendong Bai and M. Singhal. "DOA: DSR over AODV Routing for Mobile Ad Hoc Networks". In: *IEEE Transactions on Mobile Computing* 5.10 (Oct. 2006), pp. 1403–1416. ISSN: 2161-9875. DOI: 10.1109/TMC.2006.150.

[182] C.-C. Chiang et al. "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel". In: *Proc. IEEE Singapore Int'l Conf. on Networks (SICON), 1997*. 1997, pp. 197–211.

[183] Tsu-Wei Chen and M. Gerla. "Global state routing: a new routing scheme for ad-hoc wireless networks". In: *ICC '98. 1998 IEEE International Conference on Communications. Conference Record. Affiliated with SUPERCOMM'98 (Cat. No.98CH36220)*. Vol. 1. June 1998, 171–175 vol.1. DOI: 10.1109/ICC.1998.682615.

[184] G. Pei, M. Gerla, and X. Hong. "LANMAR: Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility". In: *Proceedings of the 1st ACM International Symposium on Mobile Ad Hoc Networking & Computing*. MobiHoc '00. Boston, Massachusetts: IEEE Press, 2000, pp. 11–18. ISBN: 0780365348.

[185] K. N. Sridhar and M. C. Chan. "Stability and hop-count based approach for route computation in MANET". In: *Proceedings. 14th International Conference on Computer Communications and Networks, 2005. ICCCN 2005.* Oct. 2005, pp. 25–31. DOI: 10.1109/ICCCN.2005.1523800.

[186] E. Gafni and D. Bertsekas. "Distributed Algorithms for Generating Loop-Free Routes in Networks with Frequently Changing Topology". In: *IEEE Transactions on Communications* 29.1 (1981), pp. 11–18. DOI: 10.1109/TCOM.1981.1094876.

[187] G. Pei et al. "A wireless hierarchical routing protocol with group mobility". In: *WCNC. 1999 IEEE Wireless Communications and Networking Conference (Cat. No.99TH8466)*. Vol. 3. 1999, 1538–1542 vol.3. DOI: 10.1109/WCNC.1999.796996.

[188] S. Ruehrup. "Theory and practice of geographic routing". In: *Ad hoc and sensor wireless networks: architectures, algorithms and protocols*. Ed. by X. C. Hai Liu Yiu-Wing Leung. Vol. 69. Bentham Science, 2009. Chap. 5, pp. 69–88. ISBN: 978-1-60805-636-1.

[189] H. H. Saleh and S. T. Hasson. "A Survey of Routing Algorithms in Vehicular Networks". In: *2019 International Conference on Advanced Science and Engineering (ICOASE)*. 2019, pp. 159–164. DOI: 10.1109/ICOASE.2019.8723812.

[190] J. Moy. *OSPF Version 2*. RFC 2328. Apr. 1998. DOI: 10.17487/RFC2328.

[191] *OSI IS-IS Intra-domain Routing Protocol*. RFC 1142. Feb. 1990. DOI: 10.17487/RFC1142. URL: https://rfc-editor.org/rfc/rfc1142.txt.

[192] P. M. Gerla. *Fisheye State Routing Protocol (FSR) for Ad Hoc Networks*. Internet-Draft draft-ietf-manet-fsr-03. Work in Progress. Internet Engineering Task Force, June 2002. 17 pp. URL: https://datatracker.ietf.org/doc/html/draft-ietf-manet-fsr-03.

[193] Z. Haas, M. R. Pearlman, and P. Samar. *The Zone Routing Protocol (ZRP) for Ad Hoc Networks*. Internet-Draft draft-ietf-manet-zone-zrp-04. Work in Progress. Internet Engineering Task Force, Aug. 2002. 11 pp.

[194] V. D. Park and M. S. Corson. "A highly adaptive distributed routing algorithm for mobile wireless networks". In: *Proceedings of INFOCOM '97*. Vol. 3. Apr. 1997, 1405–1413 vol.3. DOI: 10.1109/INFCOM.1997.631180.

[195] C. A. Santivanez and I. Stavrakakis. "Towards Adaptable Ad Hoc Networks: The Routing Experience". In: *Autonomic Communication*. Ed. by M. Smirnov. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 229–244. ISBN: 978-3-540-32009-8. DOI: 10.1007/11520184_18.

[196] P. Samar and Z. Haas. "Strategies for broadcasting updates by proactive routing protocols in mobile ad hoc networks". In: *MILCOM 2002. Proceedings*. Vol. 2. 2002, 873–878 vol.2.

[197] P. Á. Costa, A. Rosa, and J. Leitão. "Enabling Wireless Ad Hoc Edge Systems with Yggdrasil". In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. SAC '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 2129–2136. ISBN: 9781450368667.