Efficient Opportunistic Routing in Dense Mobile Networks

Walla Al-Eidarous

Supervisor: Dr George Parisis

Department of Informatics

University of Sussex

This thesis is submitted for the degree of

Doctor of Philosophy

September 2019

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this thesis are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This thesis is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Walla Al-Eidarous

September 2019

Acknowledgements

After a journey of research and effort which crowned by the completion of this thesis, I thank God Almighty for his guidance and grace.

First and Foremost I would like to express my sincere gratitude to my first supervisor, Dr George Parisis for his invaluable support, guidance and constructive criticism during my studies at Sussex university over the last four years. His knowledge and support always encouraged and inspired me to follow my aspiration. Without his supervision, this research would have never been valuable.

Special thanks to my second supervisor, Prof. Ian Wakeman for his great support, guidance and supervision. He was the first person to engage with me in fruitful discussions, providing the foundation of the current research concept. I will never forget his endless support during my studies in Sussex university.

Besides my advisor, I would like to thank my examiners Dr. Bogdan Ghita and Dr.Ian Mackie for their encouragement, insightful comments during the examination procedure. Their suggestions and constructive criticism would be a courage for my future research path.

Moreover, I would like to extend my thanks to the members of the “Foundations of Software Systems” research group, members of the Department of Informatics and

to members of my lab (especially Mohammed Al-Asmar and Aydin Rajaei) for their outstanding support and friendship throughout my studies. I also would like to thank Dr. Patrick Holroyd for providing us access to computational resources to run large-scale simulations. And not to forget to thank University of Sussex for providing me with all needs for research

My acknowledgement would be incomplete without thanking the biggest source of my strength, my family and friends. Especially my father (Dr. Mustafa Al-Eidarous) whose been silent about his pain while anticipating my return. My mother (Dr. Ibtesam Abu Sulyman) whose dreams for me have resulted in this achievement. Had it not been for my mother's unflinching insistence and support, my dreams of excelling in education would have remained mere dreams.

At the end I thank my University Umm Al-Qura university who paid my scholarship especially the dean of computer science college Dr. Fahad Al-Dosari for his kind support who made it easy for communicating back to the college for all needs for scholarship.

الإهداء

بسم الله الرحمن الرحيم

إلهي لك الحمد على ما أنعمت به علينا من نعم ومنها نعمة العلم فلك الحمد ولك الشكر

وصلى الله على أشرف الخلق والمرسلين سيدنا محمد وعلى آله وصحبه أجمعين

إلى من أحمل أسمه بكل افتخار .. أرجو من الله أن يمد في عمرك لترى ثماراً قد حان قطافها بعد طول انتظار..

والدي العزيز

إلى ملاكي في الحياة .. إلى من كان دعائها سر نجاحي وحنانها بلسم جراحي

إلى : أمي الحبيبة

إلى من سار معي الدرب خطوة بخطوة.. إلى من تطلعت لنجاحي بنظرات الأمل

أخي : وحيد

إلى رفيقات دربي في هذه الحياة، إلى من ترتسم السعادة بضحكتيهما

أخواتي : وئام ورواء

إلى من تركت أهلها وأحبتها لتقضي الأعياد برفقتي

زوجة أخي: أبرار

إلى رفيقتي في الغربة .. إلى من أعانتني على تخطي فراق الأهل

إلى أختي التي لم تلدها أمي : فجرة قدارة

إلى من سكنو قلبي وأضفو لحياتي ألوانها .. إلى من شاطروني حلو الحياة ومرها ..

إلى رموز التفاني والعطاء: خالاتي، خيلاني وبنات خالاتي

إلى من تمنيت أن ترى فرحة التخرج في عيني.. إلى من غمرت الجميع بحبها وحنانها ودعواتها.. إلى من لم تمهلها الحياة لترى ثمار جهدي..

ستي : سارة

إلى من غمرتني بدعواتها الدافئة .. إلى ما تبقى من رائحة الحبيبة ستي

خالتي : فطيمة

إلى كل من ساندني ودعا لي بظهر الغيب

فلكم مني جزيل الشكر والعرفان

Abstract

The usage of smartphones is nowadays ubiquitous. Their simultaneous support for long- and short-range communication has enabled the deployment of opportunistic, device-to-device networks, which exploit human mobility to enable and facilitate communication and content exchange among peer devices. Devices connect to each other without human intervention, potentially with the assistance of the cellular network provider. The underlying network topology constantly changes, depending on the mobility patterns of the participating mobile devices. Mobile devices support various technologies for discovering their location; GPS is very accurate but it works only outdoors and is power-hungry, whereas location discovery based on nearby announced SSIDs and/or the current cell ID is less accurate but power-friendly. Indoor localisation is much more challenging; approaches that are based on inertial sensors and dead reckoning, along with deployed beacons and pre-calculated signal strength maps have been proposed.

In this thesis, we develop GeoHawk, a routing protocol for dense mobile networks that support opportunistic communication and content dissemination among mobile devices in crowded events.

The driving use case has been the Grand Mosque, the largest mosque in the world located at the heart of the city of Makkah in Saudi Arabia. During the Ramadan and Hajj,

the Grand Mosque can get extremely crowded, with anticipated number of visitors close to 2.5 million, after the current expansion work is completed.

The proposed protocol incorporates a novel distributed localisation technique that can be used in conjunction with the protocol, when GPS is not available. GeoHawk deals with the very high density of users/devices by heavily aggregating routing information using Bloom filters. Identifiers of mobile devices that reside within specific geographical regions are disseminated in the network in the form of Bloom filters. Said geographical regions are dynamically created and destroyed; their size evolves to reflect the uncertainty in the topology, due to mobility and potential inaccuracies of the underlying location estimation mechanism. Bloom filters are also decayed to reflect information ageing. Devices exchange routing information with their neighbours and announce aggregated information (i.e. Bloom filters) in messages that propagate towards specific directions and reach distant areas of the opportunistic network. Data is then disseminated (and replicated through a simple but efficient ticketing mechanism) towards directions where the information about the existence of the destination node is stronger. Upon reaching the best-known region for the destination node, a message is either flooded, if the belief that the node resides in the region is strong (as indicated by a belief threshold), or, in the opposite case, redirected to a randomly selected region. The distributed localisation algorithm is a novel synthesis of existing techniques, including Pedestrian Dead Reckoning, estimated location sharing and particle filtering. Our approach can provide reasonable errors in the estimation, which allow the routing protocol to effectively deliver messages to destination nodes.

We evaluate GeoHawk using extensive experimentation in the ONE simulator. We have developed mobility models that approximate the user behaviour in the targeted use

cases and communication environments. We have experimented with a large variety of configuration parameters that affect the behaviour of the proposed protocol and recorded its performance in terms of message delivery ratio and latency as well as induced network overhead. We show that the GeoHawk's performance is superior to baseline protocols, namely Epidemic, PRoPHET and WSR.

Contents

List of Figures **xiv**

List of Tables **xix**

1 Introduction **1**

1.1 Problem Statement 3

1.2 Challenges 9

1.3 Contribution 12

1.4 Thesis Structure 13

2 Literature Review **16**

2.1 Mobile Network Architectures 16

2.1.1 Mobile Ad Hoc Networks (MANETs) 16

2.1.2 Opportunistic, Delay Tolerant Networks (DTN) 17

2.1.3 Opportunistic Routing 20

2.2 Unicast Routing Protocols 21

2.2.1 Topology-based routing 21

2.2.2 Position-based routing 26

2.3 Localisation 45

2.3.1 Dead Reckoning 45

2.3.2 Proximity-based Localisation 46

2.3.3 Lateration 47

2.3.4 Triangulation 49

2.3.5 Fingerprinting 50

2.3.6 Scene analysis 51

2.3.7 Collaborative Localisation 51

3 Routing in Dense Opportunistic Networks 57

3.1 Geographical Regions and Weak Bloom Filters 60

3.2 Building Routing Tables 61

3.2.1 Aggregating Routing State 62

3.2.2 Direct Location Information Exchange 65

3.2.3 Routing and Processing Announcements 66

3.3 Routing Messages 74

3.3.1 Routing a Message to its Destination Region 75

3.3.2 Routing a Message within the Destination Region 79

3.3.3 Ticketing and Replication 81

3.4 Collaborative Localisation in Dense Opportunistic Networks 84

3.4.1 Pedestrian Dead Reckoning (PDR) 85

3.4.2 Proximity and Collaboration 85

3.4.3 Particle Filtering 88

4 Experimental Evaluation and Analysis 96

4.1 Key Performance Indicators (KPIs) 97

4.2 Protocol Parameters and Mobility Patterns 99

4.2.1 Geographical Regions and Routing Tables 99

4.2.2 Announcements 101

4.2.3 Data Messages 101

4.2.4 Mobility Patterns 102

4.3 Exploring GeoHawk's Behaviour 103

4.3.1 Static Network Topology 103

4.3.2 Single-Event Network with Mobility 120

4.4 Discussion 137

4.5 Comparing GeoHawk with the State-of-the-Art 140

4.5.1 Performance Comparison in a Static Network Topology 141

4.5.2 Performance Comparison in the Presence of Mobility 145

4.5.3 GeoHawk's Performance Under Location Uncertainty 156

5 Conclusion and Future Directions 161

Bibliography **164**

List of Figures

1.1 Crowds at the Grand Mosque. 3

1.2 The Architecture of the Grand Mosque. 4

1.3 EID prayer at the end of the Ramadan. 5

1.4 Protests in Hong Kong. 7

1.5 Protests in Seoul. 8

1.6 Crowds cheering for the royal couple at Buckingham Palace. 8

1.7 Glastonbury tent city and the crowds. 9

2.1 DTN protocol stack. 18

2.2 Unicast, Multicast and Anycast routing. 21

2.3 Flooding in opportunistic networks. 23

2.4 Illustration of PRoPHETs' transitive communication. 25

2.5 Illustration of perimeter mode. 35

2.6 WSR protocol. 40

2.7 Dead Reckoning example. 46

2.8 Illustration of lateration in a 2D plane. 47

2.9 Example of triangulation in a 2D plane. 50

3.1 Aggregating incoming state with existing regions. 65

3.2 Announcing and aggregating regions - constraints are satisfied. 73

3.3 Announcing and aggregating regions - per-hop radius increment constraint violation. 74

3.4 Routing to *zero strength* regions. 76

3.5 Routing to the temporally stronger region. 77

3.6 Biasing the route of a message. 78

3.7 Redirecting a message to a different region when r_m is less than the bias threshold. 80

3.8 GeoHawk replication. 83

3.9 GeoHawk replication - preventing duplicate flooding. 84

3.10 Collaborative localisation when devices have different confidences about their location. 88

3.11 Collaborative localisation when devices have similar confidence about their location. 88

3.12 Illustration of the initialisation phase. 91

3.13 Illustration of error growth in the prediction phase. 91

3.15 Illustration of re-weighting phase. 94

4.1 Performance analysis - varying the Bloom filter size (static scenario). . . . 107

4.2 Preventing spurious flooding - varying the Bloom filter size (static scenario). 108

4.3 Performance analysis - varying the region TTL (static scenario). 109

4.4 Table size and region lifetime - varying the region TTL (static scenario). . 110

4.5 Performance analysis - varying the announcement TTL (static scenario). . 111

4.6 Table size and region lifetime - varying the announcement TTL (static scenario). 112

4.7 Preventing spurious flooding - varying the announcement TTL (static scenario). 112

4.8 Performance analysis - varying the announcement frequency (static scenario). 113

4.9 Table size and region lifetime - varying the announcement frequency (static scenario). 114

4.10 Preventing spurious flooding - varying the announcement frequency (static scenario). 114

4.11 Performance analysis - varying the path ratio threshold (static scenario). . 116

4.12 Preventing spurious flooding - varying the path ratio threshold (static scenario). 117

4.13 Performance analysis - varying the membership threshold (static scenario). 118

4.14 Performance analysis - varying the message TTL (static scenario). 119

4.15 Performance analysis - varying the number of tickets (static scenario). . . 120

4.16 Performance analysis - varying the Bloom filter size (mobility). 122

4.17 Preventing spurious flooding - varying the Bloom filter size (mobility). . . 123

4.18 Performance analysis - varying the region TTL (mobility). 124

4.19 Table size and region lifetime - varying the region TTL (mobility). 124

4.20 Performance Analysis - varying the spatial decay rate (mobility). 126

4.21 Flooded region radius and table size - varying the spatial decay rate (mobility). 127

4.22 Performance analysis - varying the temporal decay rate (mobility). 128

4.23 Table size and region lifetime - varying the temporal decay rate (mobility). 128

4.24 Performance analysis - varying the routing table size (mobility). 129

4.25 Table size and region lifetime - varying the routing table size (mobility). . 130

4.26 Performance analysis - varying the announcement frequency (mobility). . 131

4.27 Table size and region lifetime - varying the announcement frequency (mobility). 132

4.28 Performance analysis - varying the path ratio threshold (mobility). 133

4.29 Preventing spurious flooding - varying the path ratio threshold (mobility). 133

4.30 Performance analysis - varying the membership threshold (mobility). . . . 134

4.31 Performance analysis - varying the message TTL (mobility). 135

4.32 Region statistics - varying the message TTL (mobility). 136

4.33 Performance analysis - varying the number of tickets (mobility). 137

4.34 Performance comparison - varying network density (static scenario). . . . 144

4.35 Performance comparison - varying network density (mobility - 1 events). . 146

4.36 Performance comparison - varying network density (mobility - 2 events). . 149

4.37 Performance comparison - varying network density (mobility - 4 events). . 150

4.38 Performance comparison - varying the message TTL (mobility - 1 event). 151

4.39 Performance comparison - varying the message TTL (mobility - 2 events). 152

4.40 Performance comparison - varying the message TTL (mobility - 4 events). 153

4.41 Performance comparison - varying the size of the device buffer (mobility - 1 event). 154

4.42 Performance comparison - varying the size of the device buffer (mobility - 2 event). 155

4.43 Performance comparison - varying the size of the device buffer (mobility - 4 event). 155

4.44 Performance analysis - varying simulated error (mobility - 1 event). . . . 157

4.45 Performance comparison - collaborative localisation (mobility - 1 events, message TTL 250s). 159

4.46 Average location error. 160

List of Tables

2.1 Position-based protocols summery (part1) [118]. 42

2.2 Position-based protocols summery (part2) [118]. 43

2.3 Position-based protocols summery (part3) [118]. 44

2.4 Localisation systems summery. [118]. 56

4.1 Default values for static scenario. 104

4.2 Default values for scenarios with movement. 121

4.3 Default values for performance comparison (static scenario). 143

4.4 Default values for performance comparison (mobility). 147

Chapter 1

Introduction

The usage of smartphones is nowadays ubiquitous. Their support for long- and short-range communication has enabled the deployment of opportunistic, device-to-device networks [25][26]. Mobile phones support simultaneous connectivity to the Internet, through cellular or Wi-Fi communication, and to other devices in their proximity, through Wi-Fi Direct and Bluetooth. As a result, they are able to form wireless networks with other nearby devices opportunistically, while being connected to (and accessing the Internet through) Wi-Fi access points or cellular providers [108][115].

Opportunistic, device-to-device networks exploit human mobility to enable and facilitate communication and content exchange [21][82][66] among peer devices, which comprise the opportunistic network. Devices connect to each other without human intervention, potentially with the assistance of the cellular network provider [32][120][87]. The underlying topology of the formed network is subject to constant (and often rapid) changes, depending on the mobility patterns of the participating mobile devices.

Opportunistic connectivity can be used by mobile devices to form meshed network topologies in order to increase the aggregate bandwidth present in the network [31][114], which, in turn, can be exploited by exchanging messages through different paths [79]. Collaborative downloading, where mobile devices download parts of files and exchange them with other neighbouring devices is also enabled [37]. Internet service providers may also benefit by offloading network traffic [32][63] or caching popular content [87] to an opportunistically formed device-to-device, edge network.

Equally important, device-to-device, mobile edge networks can support network communication and content dissemination when the network provider is unable to offer Internet access [115][24][98]. This may be the case for very crowded events, such as football matches, large music festivals, public demonstrations or religious gatherings. In such scenarios, it is very common that only a limited number of mobile devices can access the Internet, therefore the formation of such opportunistic networks is crucial to support collaborative communication.

Moreover, device-to-device networks can be crucial in scenarios where the network infrastructure is non-existent, such as in rural areas in developing countries, or inaccessible due to natural disasters (e.g. earthquakes, tsunamis etc.) or terrorist attacks. Such opportunistically formed networks may be the only means for communication for citizens that may be in danger or require evacuation instructions.

Finally, mobile users may want to form collaborative, device-to-device networks that operate independently of the Internet providers' networks, for preserving their privacy, due to government censorship or when the underlying network infrastructure is not trusted. FireChat[1] has been used for device-to-device communication in protests in Taiwan, Iran,

[1] http://tinyurl.com/ogsz75o

Iraq, and, in very large scale, in Hong Kong, where more than 100,000 new sign-ups were recorded in under 24 hours and 800,000 chat sessions since then.

1.1 Problem Statement

In this thesis, I focus on scenarios where opportunistic networking is required to support communication and content dissemination among mobile devices in crowded events. The driving use case has been the Grand Mosque, the largest mosque in the world located at the heart of the city of Makkah in Saudi Arabia. During the Ramadan and Hajj the Grand Mosque can get extremely crowded due to night prayers or because it is the last visited site during Hajj [27, 69, 109, 93]. Figure 1.1 illustrates the density of the crowds in various areas within the mosque.

Figure 1.1 Crowds at the Grand Mosque [5].

As a result of the mosque's capacity and the density of the crowds, supporting network connectivity between devices or to the Internet is extremely challenging [69, 124]. Such network connectivity is crucial for obvious security and safety reasons. Access to cellular

networks is by definition very problematic given the number and density of mobile users. On the other hand, the deployment of Wi-Fi access points, which is undergoing, would only provide partial and potentially intermittent connectivity.

The architecture of the Grand Mosque is shown in Figure 1.2. Its doors are open all day long throughout the whole year. Visitors can engage in different activities and rituals that involve movement [4]. Currently, during peak days, the mosque hosts more than 40,000 people per hour (6-8 individuals per square meter), exceeding the initial allocated capacity [109, 93]. To deal with the large number of crowds, providing a secure and safe experience to all visitors, expansion work that will increase the total designated area (including the outdoor arena) to 400,000 m^2 and overall capacity to 2.5 million visitors is currently being undertaken [123, 27]. All existing structures in the wider area are constructed of metal and concrete which has known ramifications regarding the deployment of Wi-Fi access points.

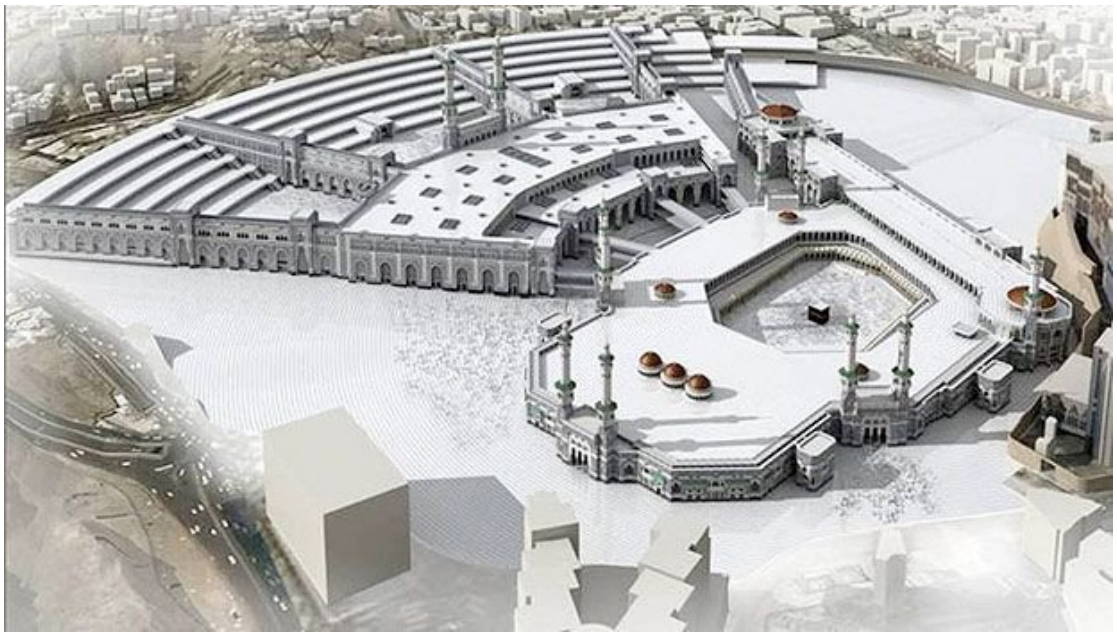


Figure 1.2 The Architecture of the Grand Mosque.

Mobility patterns. Visitors are generally free to roam within the mosque, although they are expected to observe the protocol when it comes to specific religious rituals, such as

prayers. During Ramadan these activities become more intense and there are additional prayers, each one lasting approximately two hours. Visitors usually come at an early hour to find a good spot; some arrive at the time of the sunset prayer, others arrive later in the night, while some people in the last ten days even stay in the mosque devoting themselves to worship. Eid comes at the end of Ramadan and are extremely crowded (see Figure 1.3).

Figure 1.3 EID prayer at the end of the Ramadan.

Communication environment and constrained resources. The studied network environment has a unique set of characteristics which call for a novel approach for supporting efficient communication among devices. First, the physical space is extremely crowded with mobile users that freely move within a geographically confined area. Mobility is generally low; users move slowly, often in groups and they can be stationary for long periods (e.g. praying or resting). Access to the Internet is considered to be very intermittent. The existence of Wi-Fi access points could provide specific parts of the network with Internet access (and coarse-grained location information) but connectivity would be problematic given the large open areas and the concrete walls. Location services are also considered to be intermittent given that users may reside indoors or in covered areas for long periods.

Some kind of indoor positioning system may be in place, but this could only be assumed to be providing location services to mobile users at specific points in the network.

Mobile users may be in the area (and therefore part of the network) for periods that range from some hours to a few days, therefore battery is a precious resource. Any networking substrate for supporting opportunistic, device-to-device communication should be designed with energy consumption minimisation as a primary objective, along with sensible requirements for CPU and memory. The network overhead for successfully delivering messages to recipients is a very important factor, too; given that the network is extremely crowded, one should be very careful when it comes to extensively replicating or, worse, flooding messages in large areas, as this could result in a massive amount of messages and the inevitable collapse of the network.

Wider applicability of the proposed research. Although the main ideas explored in the context of this thesis were inspired by the above-described use case, the undertaken research is directly applicable to a range of communication scenarios that share similar characteristics with the one presented above; namely, very crowded, geographically confined areas, low user mobility and intermittent access to the Internet and location services. Such scenarios include large public demonstrations, music festivals and sports events. During the past decade we have witnessed a large number of massive protests against governments. For example, massive crowds (reported to be up to a million) were gathered in several occasions in Hong Kong (Figure 1.4) and South Korea (Figure 1.5). Other notable crowded events include the million man march (1995) [122], the promise keepers march (1997) [122] and the women's March in Washington DC (2017) [107]. Large crowds were also gathered for the royal wedding of Prince William and Catherine Middleton. Approximately 1 million people were gathered at two locations in London for

the occasion (Figure 1.6)) [65]. Large music festivals are also extremely crowded events. The Glastonbury festival is nowadays attended by around 175,000 people that stay in a large but confined area for five days cellular (Figure. 1.7) [35]. The World Culture Festival, one of the largest festivals in India, was designed to host 3.5 million people in an area as large as six football fields[76].

Figure 1.4 Example of an application scenario: protests in Hong Kong [75].

Figure 1.5 Example of an application scenario: protests in Seoul [71].

Figure 1.6 Example of an application scenario: crowds cheering for the royal couple at Buckingham Palace [65].

The presented use cases share communication characteristics and challenges of a dense slow mobile opportunistic network (as discussed in the next section). Our research aims at tackling these challenges, therefore being applicable for all these use cases.

Figure 1.7 Example of an application scenario: Glastonbury tent city and the crowds [35].

1.2 Challenges

The problem space into which we lay our research presents a number of significant challenges, which we briefly describe below.

Application requirements and mobility. Message delivery cannot be generally guaranteed given the changing network topology and lack of stable end-to-end paths (and relevant state) (see *network characteristics* below). However, it may be crucial in many application scenarios (e.g. natural disasters, terrorist attacks or sudden evacuation) to ensure that messages do make it to their destination; a destination being another network device or groups of devices residing in the same broader region. The time it takes to deliver a message to its destination is also important for said application scenarios.

Users in the targeted communication scenarios move, but mobility is generally low (see Section 1.1). Nevertheless, the underlying network topology constantly changes, therefore there is a need for mechanisms that propagate changes, refresh the local state of each

node and make older information obsolete. Additionally, location discovery is required to both support the underlying routing algorithm but also to provide users with location information which may be crucial for navigating large and crowded areas, especially in emergency scenarios.

Device resources. Modern mobile phones are very capable devices in terms of processing, memory and storage resources. They support various long- and short-range communication technologies, which, in turn, provide different capabilities in terms of transmission rate and range. Battery life is a major concern; intense processing and memory access and frequent exchange of messages could result in rapid battery depletion. In our use cases, users may have to be in the same confined geographical area for a prolonged time-period with limited access to charging facilities. Moreover, given the very large number of user devices, it would be problematic to store per-device state for at a global level for routing purposes, even for today's mobile devices; instead aggregation would be required.

Modern devices support a range of technologies for location discovery, GPS being the only feasible option for the targeted scenarios, given the required accuracy levels. GPS is particularly energy hungry (if constantly on) and network deployment could also include indoors areas, therefore a collaborative localisation approach would be ideal. GPS could be periodically used to provide an upper bound in the location estimation error; indoors beacons could be used for the same reason. Exchanging location control messages with neighbouring devices and exploiting energy efficient processing for dead reckoning and internal state filtering would replace frequent GPS sampling and indoors mobility.

Although the underlying network technologies support high data rates, user mobility puts a limit on how much data two devices can exchange. As a result, control messages for both localisation and routing algorithms should be kept to a minimum so that regular

messages can be exchanged. This, in turn, limits the routing and localisation state that can be exchanged between two devices.

Network characteristics. In our communication scenarios, the network topology consists of a very large number of nodes that move. No node in the network has or could possibly have a complete view of the whole network topology. The size of the network calls for aggregation, which is reminiscent of IP *Classless Inter-Domain Routing (CIDR)* (see Section 3.2.3). Additionally, the topology changes due to user mobility, therefore it is essential to support mechanisms that periodically refresh routing state in devices' memory, End-to-end connectivity between any two devices is in most cases possible due to the large number of devices, and, therefore, available network paths, although establishing per-session state to mobile devices would be extremely challenging (see also device resources above), Long-range network connectivity (celullar or WiFi) and Internet access are intermittent or non-existent, which also severely affects devices' location discovery. Network bandwidth for device-to-device communication, in combination with low mobility (see *user mobility* above), do not pose any significant constraints on the size or frequency of exchanging control messages. However, energy consumption is a crucial aspect that influences the design of localisation and routing mechanisms in that respect (see *device resources* above).

Privacy. Privacy is an important aspect of any protocol and algorithm design that cannot be overlooked. User location and mobility patterns are sensitive information that should be anonymised or obfuscated. This way users should be able to locate and send messages to other devices if they know their identifier or an (algorithmic) way to get their identifier. Routing information should also be based on identifiers that do not reveal any sensitive information.

1.3 Contribution

The core contribution of this PhD thesis is *GeoHawk*, an opportunistic routing protocol for very dense, mobile networks. GeoHawk is efficient and effective in comparison to the state of the art protocols in terms of message delivery ratio, latency and induced network overhead. This is achieved by a unique and novel synthesis of mechanisms discussed in Chapter 3. Geohawk can operate under uncertainty with respect to the location of mobile devices, enabling deployments where GPS is not always available or battery life is very important. We extensively discuss the limitations of existing routing protocols, and identify a number of significant challenges present within our problem space. Moreover, we contribute a distributed, collaborative localisation mechanism that is based on pedestrian dead reckoning, location exchange and particle filtering (as described in Section 3.4).

We have developed simulation models for GeoHawk and the accompanying collaborative localisation mechanism in the ONE simulation [48]. Additionally, we designed and developed mobility models that enable us to evaluate the performance of GeoHawk (in comparison with the state-of-the-art) in realistic scenarios. More specifically, since we have observed that our case study and a range of communication scenarios (including large public demonstrations, music festivals and sports events) share similar characteristics in terms of density, mobility, network size and accessibility, we decided to adopt a generalized mobility model, which supports a festival-like movement where mobile devices gather at different (pre-specified) areas for a specific amount of time. Various events may be taking place simultaneously and a number of devices can be moving from event to event within the broader area (for more information see section 4.2.4).

We have extensively evaluated GeoHawk in a number of network scenarios and compared its performance against the Epidemic [112], PRoPHET [60] and WSR [2] protocols. We have investigated the baseline performance of all protocols in a mobility-free scenario in order to understand their behaviour and the various parameters that affect their performance. We have then evaluated their performance when mobility is present in dense deployments that mimic large-scale festivals in open areas with a variable number of events. We have investigated an extensive set of parameters that affect the behaviour and performance of GeoHawk, including the network density and associated number of mobile devices, frequency of exchanging routing information, size of routing tables and region characteristics (lifetime, size, and decaying). We have investigated how replication through the proposed ticketing mechanism and message redirections affect the message delivery ratio and induced network overhead. Finally, we evaluated how localisation errors affect the performance of GeoHawk when the collaborative localisation mechanism is in operation.

1.4 Thesis Structure

In Chapter 2 we present relevant research to the proposed protocols and mechanisms and highlight the novelty of our proposals in contrast to the presented related work. We focus on relevant work on geographical routing in opportunistic mobile networks and identify limitations with respect to the scale of the network and assumptions about underlying localisation services. Past work on localisation of mobile devices using inertial sensors and dead reckoning is also presented.

In Chapter 3 we present GeoHawk with its two distinct routing phases. In the first phase, a message takes a (replicated) random walk that is biased towards the geographical area that the destination is believed to be residing (see Section 3.3.1). Biasing takes place when a device holds more accurate routing information than the one carried along with the message. A fixed number of tickets is used to replicate a message whenever the walk of a message needs to be biased towards a geographical region that is in a different direction to the current one (as presented in Section 3.3.3); this significantly increases the geographical coverage of routing without inducing any significant overhead. Upon reaching said geographical area, a message is flooded if this belief is strong enough, as defined by a configurable threshold (see Section 3.3.2). In the opposite case (where the system believes that the destination doesn't resides within an area) a new random walk is attempted. Messages are discarded according to a Time-To-Live field that they carry. We also describe how devices build and maintain their routing tables by directly exchanging routing information with their neighbours and disseminating routing advertisements that contain aggregated routing information towards randomly selected directions in the network (Section 3.2). Then we describe the proposed collaborative localisation mechanism which can be directly integrated to GeoHawk; the proposed mechanism integrates particle filtering, dead reckoning and exchange of location information to provide sufficiently accurate information for dense opportunistic networks. The proposed mechanism can be used on its own in scenarios where users need to report their location to each other in crowded events or to rescue/security services in natural disaster or terrorist attack scenarios. When integrated with GeoHawk, the location uncertainty, as calculated by the localisation mechanism, is expressed as part of the uncertainty of the exchanged routing information, which uses regions instead of points when defining the local knowledge of a device about

its neighbouring nodes. This, in turn, results in weaker routing information when biasing random walks.

In Chapter 4, we discuss the Key Performance Indicators for GeoHawk, the different parameters that influence its behaviour as well as extensive experimental evaluation, which includes performance comparison with the state-of-the-art protocols, namely Epidemic [112], PRoPHET [60] and WSR [2]. In our evaluation we also study the impact of network density and mobility models on the effectiveness and efficiency of our research proposals. We evaluate a simple model (where we emulate the error) as well as our collaborative localisation approach and explore the effect of localisation inaccuracies in routing messages within the opportunistic network (Section 4.5.3). We also look at how the size of the local state kept at each device affects the performance of both mechanisms. Finally, we conduct an analysis for all major configurable parameters in our proposed mechanisms.

Chapter 5 concludes this dissertation by discussing the key findings of our research and suggesting ways forward in opportunistic routing in densely populated mobile networks.

Chapter 2

Literature Review

2.1 Mobile Network Architectures

In this section we discuss routing protocols for mobile, device-to-device networks and identify their limitations with respect to the studied network environment. Before doing so, we briefly discuss relevant network architectures along with their main characteristics.

2.1.1 Mobile Ad Hoc Networks (MANETs)

MANETs are infrastructure-less, wireless networks where devices can move, therefore the network topology constantly evolves. Devices act as mobile routers, collectively managing and updating routing tables, and are responsible for forwarding information to other devices. Mobility is assumed to be low and nodes are within a few hops of each other, therefore mobile devices are able to maintain routing information for routable, end-to-end paths in the network.

MANET protocols can be categorised as proactive and reactive [20]. In proactive protocols, such as OLSR (Optimized Link State Routing) [36] nodes maintain up-to-date routing information that reflects the network topology. This is achieved by frequently exchanging routing information, which comes at the cost of high network overhead. Unlike proactive routing protocols, reactive protocols such as DSR (Dynamic Source Routing) [39] and AODV (Ad hoc On-Demand Distance Vector routing) [83] instantiate the route discovery process only on demand, resulting in a higher latency but lower network overhead. All these protocols will only operate successfully in environments where the existence of end-to-end paths is guaranteed. In the opposite case (i.e. proactive routing), routing fails and packets are excessively dropped until a path can be discovered and established. Given the context of this thesis, we will not be discussing MANETs in more detail.

2.1.2 Opportunistic, Delay Tolerant Networks (DTN)

Opportunistic, Delay Tolerant Networks are designed to withstand extensive delays and frequent connectivity intermissions. The earliest research is dated back to 1998 studying the development of Interplanetary Networks (IPNs) [1], which aimed at establishing deep space communications. It was quickly discovered that other network environments share similar characteristics to IPNs (i.e. long delays, unavailability of end-to-end connectivity and unknown network topology), therefore opportunistic networks have become a prominent research area.

Opportunistic, mobile networks [25][16][26] can be severely constrained in terms of the availability of computing, memory and energy resources at mobile devices. The potentially high mobility results in intermittent connectivity and constantly evolving network topology.

End-to-end paths cannot be assumed to exist among all devices at all times. These requires buffering of packets and may result in potentially long delays in delivering messages.

The DTN architecture [25][16][26] suggests the implementation of a bundle layer within the IP Stack of a DTN device, as shown in figure 2.1. This layer provides features (e.g. custody transfer and store- carry-forward scheme) that address the issue of intermittent connectivity, while supporting the interoperability between different networks. The bundle protocol embedded within this layer deals with message bundles rather than individual packets (i.e. it groups small data packets into a large bundle before transferring is initiated). Each bundle is provided with an Endpoint Identifier (EID), which identifies a receiver. Given the lack of a stationary network topology, the process of binding an EID to a specific bundle can happen not only at the source but at any time during the bundles' lifetime (late binding). As a result, each node compares its own ID against the EID of a received bundle and decides whether they both match. A matching address implies that the current node is the final recipient of this bundle. The main characteristics of an opportunistic network architecture are briefly described below.

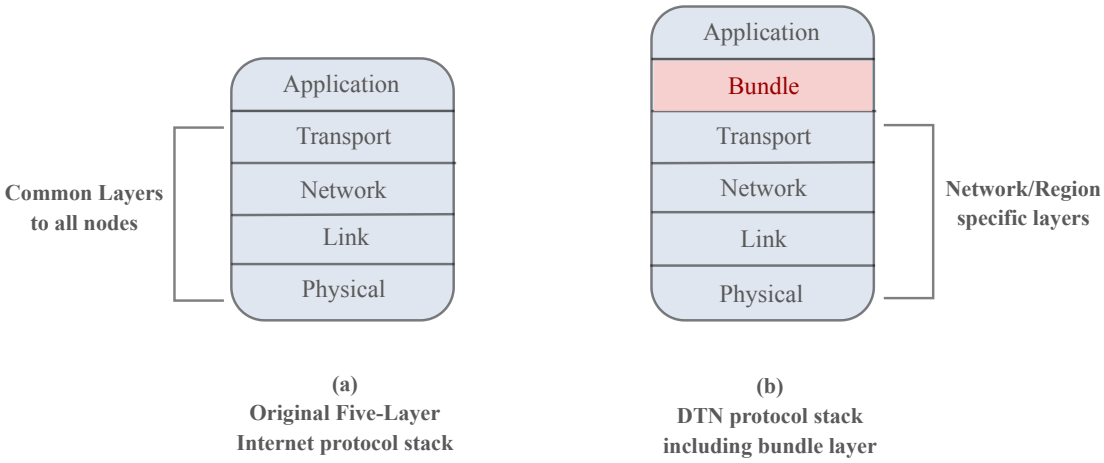


Figure 2.1 DTN protocol stack [49].

Opportunistic networks support a flexible naming scheme; each node within the network is allocated with a unique identifier. Addresses enable the dissemination of information to intended recipients. As mentioned earlier, mobile devices use these addresses to identify the recipient of bundles. Opportunistic networks route and deliver bundles that are larger in size compared to packets in regular networks and include additional information in their headers in an attempt to improve their delivery ratio; i.e. additional information can enhance routing and assist in both scheduling and buffer management. Both IP and bundle protocol headers (e.g. custodian EID, timestamp, lifetime) are included within each bundle. These fields enable various features such as priority routing and packet disposal (i.e. packets are disposed of when the value of the respective lifetime field expires, preventing unnecessary queuing and buffer depletion). Furthermore, extension blocks such as security-specific ones may be added/removed during transit to enable/disable security features, such as authentication, message integrity and confidentiality [97].

The nature of opportunistic environments suggests the importance of sustaining high message delivery ratios within the network. This is achieved through custody transferring, which allows delegating the delivery responsibility from one node to another once a consensual agreement takes place[25][16][26]. After receiving the delegated packet, the custodian node then persistently stores the message in its buffer until it is able to transfer the packets' custody to a suitable candidate. In situations where the possibility of encountering other nodes is low, the custodian node takes on the responsibility of delivering the packet to its destination [25][16][26].

2.1.3 Opportunistic Routing

Routing is a fundamental component of any network infrastructure, the objective of which is to exchange and maintain routing information that enables nodes to forward packets to their destinations efficiently and effectively. As discussed above, mobile routing protocols that assume the existence of end-to-end paths (e.g. AODV [83] and DSR [39]) would inevitably fail in an opportunistic network environment. Instead, opportunistic networks follow a "store-carry-forward" routing scheme, where nodes may store data packets upon receiving them, before subsequently forwarding them to other nodes[25][16][26]. This inevitably induces extra latency in delivering messages to their destination(s). Network routing in the context of mobile opportunistic networks can be categorised as deterministic or dynamic[100].

Deterministic routing. Deterministic protocols adhere to the expectation that a networks' topology should be known and available to network nodes. Reliable or predictable information about the network state (i.e. links' availability and nodes' future mobility patterns) is exchanged in order to discover and establish optimal paths. Protocols classified under this category (oracle, link state space time, and tree based protocols) can work efficiently in environments with predictable movement patterns [100].

Dynamic (Stochastic) routing. In many opportunistic network environments, mobility patterns cannot be known or predicted. Dynamic networks assume no knowledge of the network topology, movement or connectivity models[100]. Instead, protocols rely on replication and flooding to relay messages within the network. Much research has been done over the last decade [10][49][126][127] exploring various dynamic routing mechanisms (e.g. -of driven categorise from [10] are- epidemic, history, model, coding

and flooding based protocols). Dynamic routing is also the main focus of this thesis. In the overview below, we follow the taxonomy introduced in [10].

Opportunistic routing protocols could be divided in unicast, multicast, anycast and geocast protocols. Below we discuss the most prominent protocols for opportunistic networks, focussing on unicast protocols. We identify and examine their limitations that our research aims to tackle.

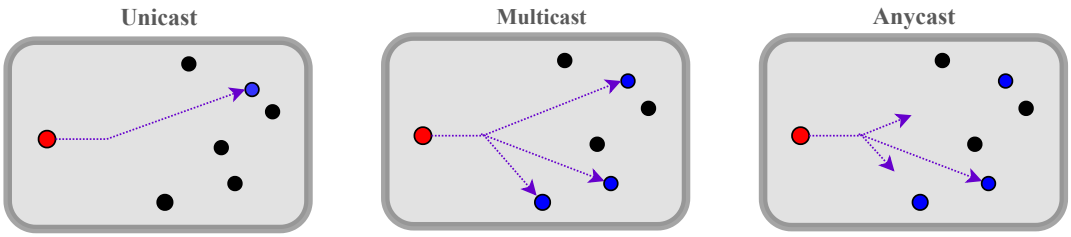


Figure 2.2 Unicast, Multicast and Anycast routing.

2.2 Unicast Routing Protocols

Unicast routing protocols in mobile, opportunistic networks enable point-to-point communication between mobile devices using unique endpoint identifiers for mobile devices (see figure 2.2). Unicast routing protocols can be classified in topology- and position-based protocols[67].

2.2.1 Topology-based routing

Topology-based routing protocols [10] exploit a node's environment, collecting the necessary metrics to select a suitable candidate for delivering a message to its destination. The selection decision relies either on local or global information. Exchanging and maintaining information locally promotes scalability, while maintaining global information provides

an overall view of the network, which potentially improves the protocol's performance. The first option is more suitable for an opportunistic network, as processing information globally will introduce undesirable latency[1]. Research in this area can be categorised as follows.

Epidemic/Flooding Routing. Epidemic routing relies on flooding to deliver messages to their recipients. Its simplicity means that mobile devices do not need to store any information about their local connectivity, network topology or devices' mobility patterns. Flooding aims to increase message delivery ratio at the cost of (potentially very) high network overhead. The earliest work on epidemic routing is the one by Vahdat and Becker [112]. In this approach, a node replicates all messages stored in its buffer, and distributes them to all its neighbours residing within communication range. Messages eventually arrive at their destination (see figure 2.3). However, replication can cause serious problems (e.g buffer overflow and battery drainage) in resource-constrained opportunistic network environments. As a result, methods on restricting replication gained a lot of research traction. The work in [104] is based on the assumption that the current location of the destination node is likely to be around the area of its last known position. Thus, the protocol sprays message replicas only towards the direction of the estimated destination location. Spray and wait [101] is another protocol that also controls replication. As the name suggests, this process consist of two stages; in the first phase, a source node generates a limited number of message copies (e.g. 3 to 5). During the wait phase, nodes wait for direct encounters before distributing those copies [10]. The protocols proposed in [73][99] also belong to the epidemic/spray category.

[1]Intermittent connectivity, limited resources and mobility in an opportunistic network result in latency. Maintaining a global system will only contribute more to this issue.

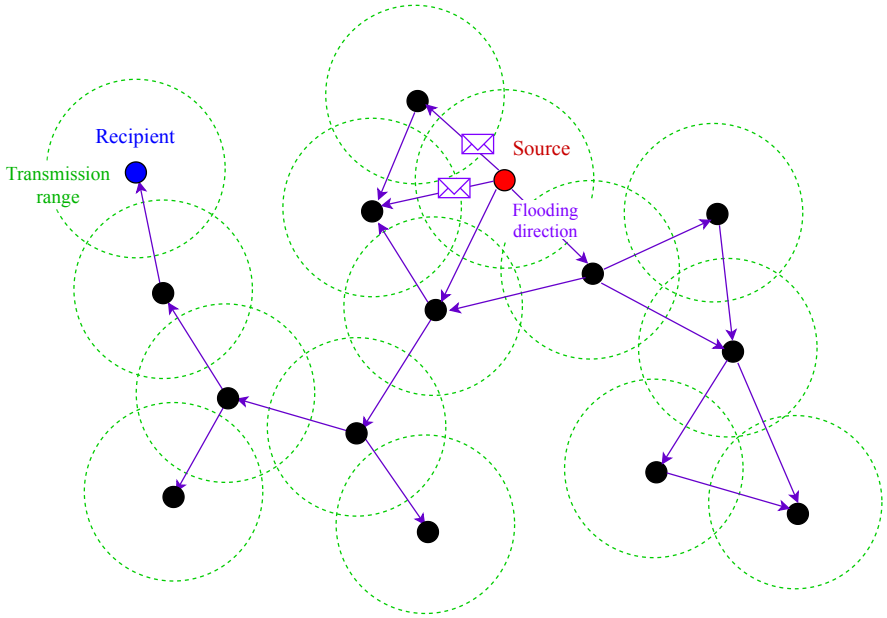


Figure 2.3 Flooding in opportunistic networks.

Estimation-based Routing. Instead of naively flooding the network and wasting potentially scarce resources, estimation-based routing protocols estimate the delivery probability for each of the neighbours of a mobile device. A message is then forwarded to the device that maximises this probability. When no suitable candidates are available, the device stores the packet until a future opportunity arises or it encounters the destination.

Estimation-based Routing is usually deployed in environments where mobile devices are expected to encounter other devices frequently. The main assumption is then that a node that was previously encountered is more likely to be encountered again in the future[112]). This assumption implies that probabilistic routing will have the advantage over flooding protocols in such circumstances.

PRoPHET (Probabilistic Routing Protocol using History of Encounters and Transitivity) [60] is one of the protocols in this category. PRoPHET estimates the delivery probability of each neighbour in range, and appoints the one with the highest delivery probability to be the new custodian of the packet. Messages are kept in the buffer of the origin node if

there is space in its buffer; i.e. effectively replicating the message. PRoPHET performs better compared to epidemic-based protocols under the assumption mentioned above [112]. In particular, assuming predictable and non-random mobility environment, history of encounters is used to estimate the delivery rate of a destination via an intermediate node. These sets of probabilities are stored and maintained in a local routing table, in the form of $p<M, D>$, where p is probability, M is a relay node and D the destination. The tables along with the neighbouring nodes' information are exchanged upon encounter. An adaptive algorithm running at each mobile device updates these sets of probabilities. Specifically, when the received information is not found in a node's table, its mapped probability is set to zero; i.e. probability of first-time encounters is set to 0 by default. Contrary, if an entry exists, then its probability will increase, while other entries are aged. Subsequently, the node uses the received table along with the transitive property to recalculate probabilities for each destination, assuming that an intermediate node E is likely to meet M and D (i.e. $P(M,E) * P(E,D) = P(M,D)$).

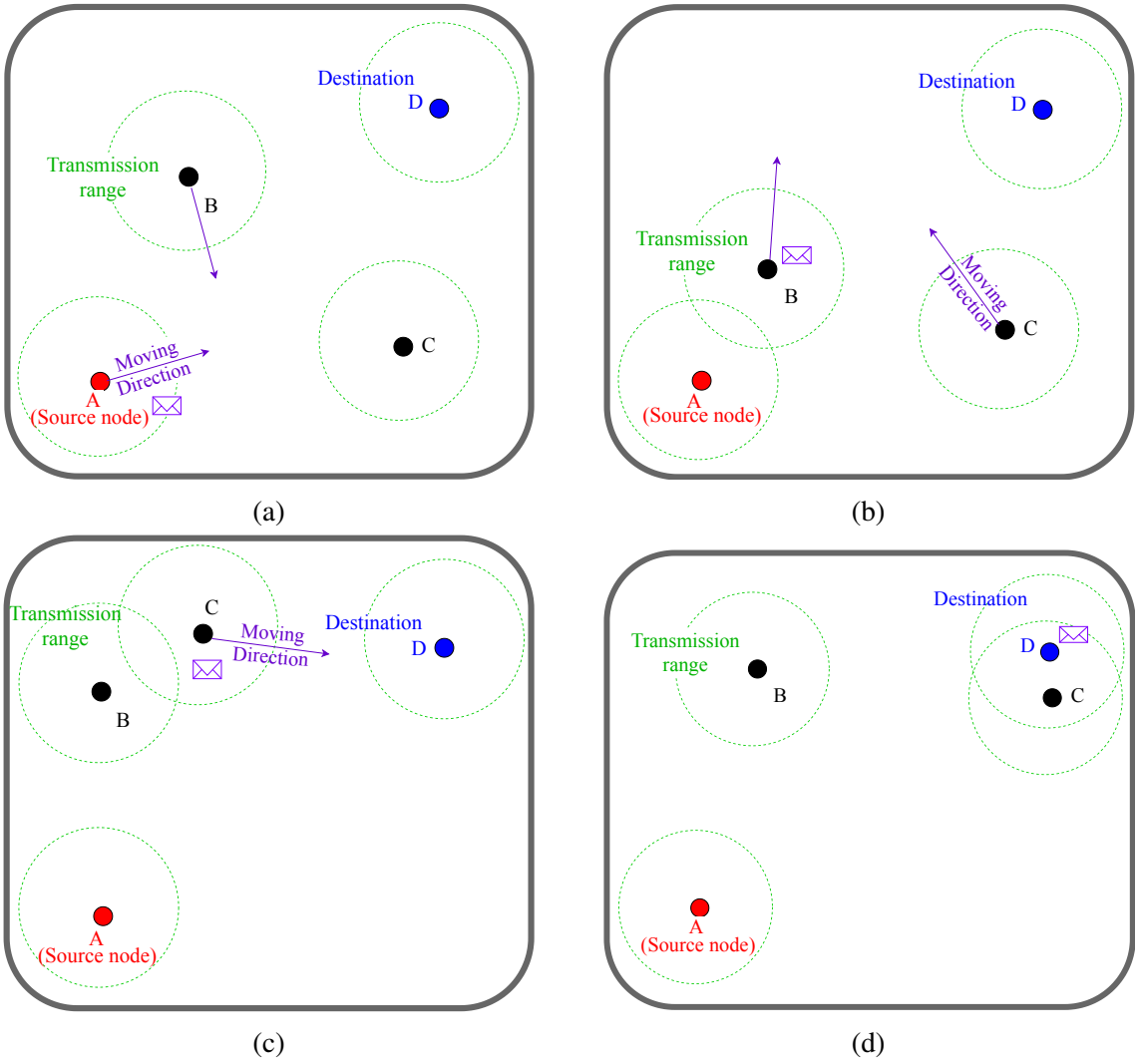


Figure 2.4 Illustration of transitive communication, where node *A* passes a message to node *B* and *C* in order to reach its destination *D*.

Figure 2.4 shows that a message is able to reach its destination exploiting mobility and the use of transitive property. In sub-figure 2.4a, node *A* tries to deliver a message to its destination *D*. However, there is no direct path between *A* and *D*. Thus,*A* uses the mobility of intermediate nodes *B* and *C* to deliver the message to *D*, as shown in sub-figures 2.4b,2.4c and 2.4d.

The same objective is adopted by other proposals, such as [22], [40], [72] and [10]. As with PRoPHET, they rely on estimating the delivery ratio to drive the best neighbouring

node. However, they differ in the mechanism used to measure this probability. Protocols under this category generally perform better than epidemic routing, with respect to memory usage, delivery time and buffer requirements. This is due to the utilisation of available information for metric measurement which leads the protocol to be selective of relays and routing paths and restrictive of unnecessary message replications. One disadvantage of this approach is that it requires more computation resources to perform the needed calculations for discovering potential relays. Another issue that needs to be considered is privacy concerns, which originate from the process of exchanging private context (e.g location or future destination) with other encounters.

Calculating probabilities for each neighbouring device requires maintaining state for all devices in the network, which is impractical for proposed application scenarios (see 1.2). In very crowded network environments, it is imperative to aggregate routing information so that locally stored network state (and the associated network overhead when disseminating state) can be controlled.

Coding-based approaches. Code based protocols [119] [19] [113] [30] [59] employ network coding to achieve transmission reliability. Supporting this feature in our environment is not practical given that it will affect the protocol's efficiency due to the computational resources (and the associated energy consumption) required to support encoding and decoding.

2.2.2 Position-based routing

Unlike topology-based routing, position-based routing requires a localisation system to efficiently forward messages to their destinations. Generally, each mobile device is required

to know its own location and have the ability learn or estimate the position of the destination [118]. Positioning services can be local, centralised or hybrid (see Section 2.3).

Vector Routing (VR) [44] does not require any knowledge of the location of the destination device and relies on restrictive/selective flooding to reach a particular destination. The amount of replication is decided based on the angle of encounter ω (utility metric), where $\omega \in [0, \pi]$. The Smaller the ω the smaller the amount of replicas and vice versa. Vector Routing places such restrictions in an attempt to prevent any form of redundant replication. Assigning a large number of replicas for an encountered node moving to a similar direction (indicated by the small encountered angle) will only result in replication redundancy, likewise large replication handed to nodes moving in different directions specifically when $\omega = \pi$ can also lead to the same outcome as the encountered node may move along the previous trajectory of the message carrier [2]. Thus, VR employs a function that takes the encountered angle ω as its input and decides on the amount of replication and the suitability of the relay/current encounter. VR may be ineffective in terms of redundant replication in certain scenarios, as discussed in [118].

Unlike VR, RoRo-LT [110] relies on long term observations to geographically forward messages within an opportunistic network. As with VR it does not require any knowledge of the location of the destination device. RoRo-LT devices keep a historical record of their own activities and locations. Devices self-observe their behaviour for long periods (often in the order of weeks) before the protocol can fully operate within a defined network environment. Once useful information is extracted, the protocol uses the gathered knowledge as a baseline when examining the current behaviour of a device. Assuming devices follow their own routines, the protocol can reliably predict a node's mobility

[2]Moving along the same previous trajectory of another node implies that the current node will likely face the same encounters leading to redundant replication.

patterns and, consequently, future encounters. Similarly, to VR, RoRo-LT tries to increase message delivery by selecting relays that will be distant or will not be encountered in the future; VR tries to spread the message by replicating to nodes in different directions. However, it has been shown that the protocol could perform poorly as the process of collecting information for a long term is in many cases unrealistic [118]. This would certainly be the case in our targeted network environment (crowded scenarios and one-off events), where devices may appear for the first time, stay for a few hours or days, and (potentially) never return.

Motion Vector (MOVE) [118] protocol assumes that the location of the destination is fixed and known, and relies on direction and distance to decide the suitability of an encountered node as its next relay. Nodes moving towards a specific destination are more likely to be selected as custodian nodes [118]. In situations where all neighbouring nodes are moving towards/away from a destination, the best node is decided according to a distance metric. MOVE is tailored for stationary destinations which implies that it will not perform well in environments with moving destinations and no global positioning system. In addition, the protocol does not consider nodes speed when selecting a candidate; faster speed contributes to a faster delivery [118]. As a result, these un-addressed issues will only introduce delay and/or decrease delivery ratio [118].

GeOpps [56] and *GeoSpray* [118] are geocasting protocols; i.e. they route towards a specific geographic region. They could be used as unicast protocols by flooding a message when this reaches the destination region. Unlike MOVE, *GeOpps* [56] considers nodes' speed in the selection process. In addition, it suggests a path that links a source node to its recipient based on a list of computations (as [56] explains in detail). Once this path is approved, the forwarding process is initiated, and the next relay is selected according to

a nearest point (NP) policy; NP or location of the next relay must be near the suggested route and close to the destination. GeOpps uses the Minimum Estimated Time of Delivery (METD) metric to verify the suitability of an NP [56]. The average moving speed is also considered in this qualifying process, facilitating a lower latency compared to MOVE [118]. *GeoSpray* [118] further extends GeOpps by introducing restricted replication and packet prioritisation; to address the limitations of network resources, messages are prioritised based on available bandwidth and buffer capacity. Unfortunately, both protocols are not particularly scalable as the process of establishing a route and the complex computations to select an NP are heavyweight [118], especially in large and dense environments with moving devices. They also do not offer any solutions to address the possibility of never encountering a relay with a lower METD value [118].

AeroRP [84] was designed for aeronautical networks (ANs) and uses the Time-To-Intersect (TTI) metric to decide on the best relay based on the node's current speed, distance and angle to its destination [84]. Compared to GeOpps [56], the routing policy of AeroRP is limited. Message custody can only be transferred to relays moving along the same direction towards the destination [118]; according to this policy, a message carrier moving away from a destination will never forward its packets to any encounter moving towards a recipient. Sharing similar specification with GeOpps protocol indicates that AeroRP will inherit the limitations mentioned above.

Delegation Geographic Routing (DGR) [13] adopts an optimization policy, which overcomes AeroRP's TTI limitation [118]. The policy requires nodes to historically record their past encounters after each successful transmission and accordingly, it forces the protocol to revise the relay qualification process. Instead of comparing the TTI metric of an encountered node with the custodians' metric, DGR compares the encountered

node's metric against the TTI of a past encounter historically recorded by the message carrier. DGR also uses a heuristic approach [118], which enable the protocol to estimate a packets' lifetime and detect the ones close to expiry. Once identified, DGR uses replication to increase delivery probability of those messages. DGR suffers from most of AeroRP disadvantages, making it incompatible with crowded network environment and non-stationery destinations. Moreover, using the recorded metric of a previously encountered node will further implicate its performance in situations where routines are unpredictable and localisation is coupled with errors.

The *Mobility Prediction Based Adaptive Data (MPAD)* [129] protocol is also tailored for sparse, mobile networks, which consist of mobile sensors and stationary sinks (destinations). Sensor nodes collaborate with each other in order to exchange, gather and deliver a bundle of information to any available sink. MPAD estimates the delivery probability for each encounter based on movement predictions, which consider the moving direction of a node; the intersection between a node's trajectory path (A) and a sink's transmission range determines the moving direction. This metric is then used to decide the next candidate to delegate the custody of the respective bundle; the higher the probability, the more suitable a relay is. In situations where this approach fails, the protocol resorts to a different metric that is calculated using the two tangents touching the transmission range of node (A) and the sink in order to obtain the communication angle. Distance is then inferred and used to measure the delivery probability. A small distance indicates a large communication angle and a high delivery probability[3]. Assuming that the destination location is known in advanced makes the protocol impractical in environments that place no restrictions on end nodes' mobility. Blind and unnecessary replication triggered by messages close

[3]The nearer a destination to a sink node, the larger this angle.

expiry is another disadvantage that will negatively affect overhead and latency in crowded networks. Furthermore, searching for a single destination within a crowded network, where the possibility of location system error is present, affirms this belief.

Distance Aware Epidemic Routing (DAER) [34] considers end nodes' mobility and assumes the existence of a positioning server that provides online localization information [118]. DAER routing relies on a distance metric, movement direction and replication to deliver messages to their destinations. Specifically, the custodian nominates encounters with a close distance to a destination (closer than itself) to be the next relay, then generates a replica and passes it on to the appointed node. After every successful transmission, the custodian inspects its direction of movement, which decides the fate of the original message and as a result limits replication. Accordingly, the original packet is discarded when the custodian moves away from its destination; replicating in the reverse direction of a destination will only induce more overhead without any significant gains. A major concern with this approach, is that the process of exchanging real time information between network nodes and a centralised location service system will considerably contribute to communication and routing delays [118], especially in crowded networks. Furthermore, DAER assumes the existence of a global server, which, in many scenarios, may not even be possible. Localisation errors on the other hand can reduce the prediction accuracy of the movement direction. DAER's replication mechanism in congested environments may end up flooding the network, which is undesirable. The protocol does not make any use of other metrics such as speed to improve its performance [118].

Packet Oriented Routing (POR) [58] further extends DAER[34] by controlling the amount of replication based on the distance metric. POR will reduce the amount of replication for large distances to destination devices. Although this feature improves

transmission reliability (considerably in a limited bandwidth network), it still cannot overcome the design limitations and disadvantages of DAER [118]; e.g. POR requires a centralised positioning server.

LAROD [53] (short for Location Aware Routing for Delay tolerant networks) was designed for sparse networks. It establishes a Location Dissemination Service (LoDiS), which enables each node to maintain a local table of node positions. The content of this database is refreshed and updated with the support of periodical broadcast messages, where the broadcast rate is higher for nearby nodes and lower for far ones[4]. The protocol finds relays based on the distance metric, as DAER [34] and POR [58] do; i.e. it selects any encounter closer to a destination than itself. It also adapts the replication mechanism so that a message is directed towards a destination whilst preventing replication redundancy. As the process of information propagation takes time, the protocol assumes that the accuracy of destination information drops for nodes further away. As a result, upon each encounter, the location of the final recipient carried by the data message is updated with fresher information whenever such information is available. LAROD's reliability heavily depends on the accuracy and validity of the recorded information about network nodes [118]. All these protocols are designed for sparse network environments and would not be practical for the targeted scenarios that involve very crowded networks.

GeoDTN [61] takes into account the social habits of users. Here, each node keeps a record of historical information in the form of a cluster (represented with a bivariate distribution and confidence), to demonstrate the age and reliability of its information. Upon encounter, nodes' locations are exchanged as well as their clusters. The received information is then merged according to the associated confidence. The protocol further

[4]The DREAM Location Service (DLS) supported by *LAROD* allows the deployment of two different broadcast rates [53].

exploits this information by computing a metric, which is useful in predicting future encounters. Once these steps are accomplished, GeoDTN decides whether this relay is qualified to bear the packets' custody. The decision is made based on the distance, rescue and scoring mode, respectively [61]. Specifically, the first mode (distance mode) sets the minimum distance as a prerequisite of custody transfer; i.e. a relay is selected if the distance metric is smaller than a predefined threshold. Once nodes satisfying this condition cease to exist, the protocol transitions into the rescue mode, which tries to overcome the local maximum problem. As a consequence, the relay qualification mechanism is adapted to randomly select any encounter that satisfies a predefined probability; i.e. packets are randomly forwarded based on a predefined probability. In situations where the distance metrics of pairwise nodes are above/higher than the distance mode specified threshold, the scoring mode becomes the protocols' primary mode, in which it assigns the packets custody to the one with the highest score. GeoDTN inherits LAROD's disadvantages; it does not scale with the size of the network nor it performs well in crowded environments. Dealing with the price of growing/maintaining a large table of information (clusters) and routing to absolute locations still remains an issue in GeoDTN. Moreover, social habits may not be known or relevant, especially for one-off network deployments (e.g. music festivals). Thus, assuming the ability of extracting mobility patterns and predicting future encounters from movement routines cannot be realised at all times.

Similarly the *Approach and Roam (AaR)* [12] protocol, which is an extension of Location Aided Routing (LAR), relies on exchanging historical records to successfully route packets to their destinations. AaR routes messages in two phases: Approach and Roam. In the approach phase, the protocol estimates the movement range of the destination using its last known speed and position. The packet is then replicated towards the identified

range. Once reached, the protocol switches to the roaming phase, which adapts the conditions of the approach phase and restricts replication within the specified area. Unlike the previous protocols, AaR represents the final destination position as an area instead of an absolute location, which scales down the information mismatch issue in sparse networks. However, this is not the case in congested, slow paced environments, where the size of the estimated range is very small and/or comparable to absolute locations. Thus, a mismatch in information in such circumstances will most likely lead to low performance. Furthermore, AaR does not provide any mechanism to guard against localisation errors. In addition, the protocol is incapable of efficiently scaling with the network; the process of recording and maintain information of individual encounters in a congested setting will negatively impact the network overhead and memory usage. Another similar work to AaR is *Converge-And-Diverge (CaD)* [118], which adopts an optimisation policy aiming to reduce overhead. Despite the suggested enhancements, AaR and CAD still remain impractical for congested networks, as discussed above.

The Greedy Perimeter Stateless Routing (GPSR) protocol[47] is the underlying forwarding mechanism used by both WSR[2] and GeoHawk. GPSR supports two operation modes: *greedy* and *perimeter*. By default, the protocol operates in greedy mode, which selectively forwards messages to nodes closest to a given destination. Greedy forwarding continues until the message is delivered or a suitable relay is unavailable. In the latter case, a recovery mechanism referred to as perimeter mode is triggered, enabling GPSR to extract routing information from a local planar graph and resume forwarding based on the right-hand rule; planar graphs maintain information on neighbours in range. The right-hand rule increases the opportunity of destination encounter, as it allows messages

to traverse the network avoiding further overhead caused by path loops. The protocol switches to its default greedy mode whenever a suitable candidate is encountered.

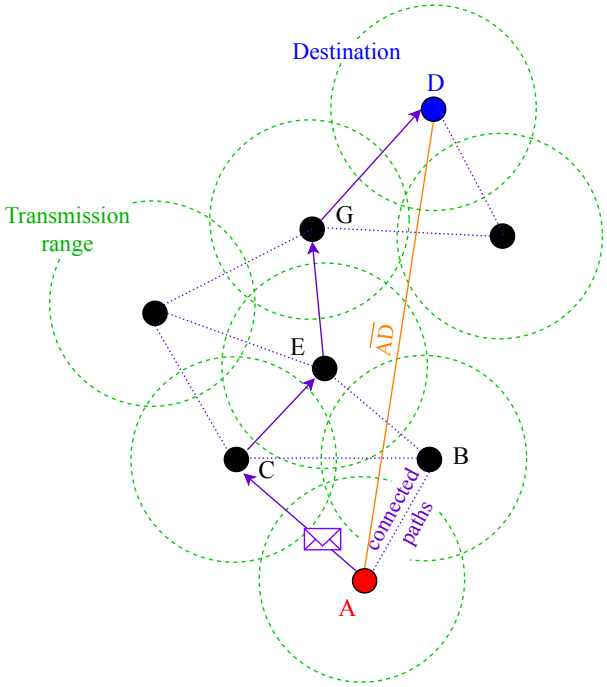


Figure 2.5 Illustration of perimeter mode.

An illustration of GPSR's perimeter mode is depicted in Figure 2.5. In this figure, node *A* attempts to deliver a message to destination *D*, however neighbouring nodes *C* and *B* have equal distance to node *D*. Thus, GPSR tries forwarding a message along the interior *face* of the planar graph using the right-hand rule, which selects relays with the smallest angle in the counter-clockwise direction. As a result, node *A* forwards the message to *C*. Accordingly, *C* forwards the message to *E*. *E* selects *G* and *G* delivers the message to *D*. However, messages may traverse a number of inner/outer faces before they even get delivered to their destination. This is because GPSR tries to eliminate routing loops by always checking for an intersection between the forwarding edge and the one that connects a destination to the node that triggered the perimeter mode/face change (which is $\overline{AD}$

marked in orange)[5]. However, GPSR is unsuitable for crowded networks as a stand-alone protocol, especially when the location of the destination is not known and fixed ; i.e. the destinations' position has to be known by GPSR prior to routing otherwise it cannot function. Furthermore, GPSR suffers from information mismatch, when localisation errors are present. GeoHawk proposes a novel synthesis were we incorporate GPSR with other mechanisms/techniques in order to provide efficient and effective routing even in very crowded areas.

Hybrid protocols that can operate with and without knowledge about the location of the destination device have been proposed. The *History Based Vector Routing (HVR)* [45] protocol relies on network nodes recording the information of historical encounters (e.g. speed, movement direction and transmission range), which is then used to guide the selection of candidates for delegating custody of messages. The collected information enables HVR to predict the destinations' movement range as well as estimating the probability of encountering a destination via an identified neighbour node.

Candidate's exchange their current information (e.g. location, speed, moving direction and transmission range) as well as their historical vector (recorded information of previous encounters) with the custodian node (i.e. message carrier). Upon receiving this, the custodian computes the rendezvous probability (as described in [45]) for each encounter to decide on the best next relay. The calculation is based on the current information of an encounter with the one historically recorded by the custodian to envision the size of the overlapping area between the candidate and destination regions. The larger this area (aka. the higher the probability), the more preferable a candidate is, and as a result the packet is replicated to the selected neighbour (i.e. to the node with the highest probability).

[5]Packets that get stuck in a specific route (e.g. looping indefinitely) for long periods end up wasting network resources, affecting delivery rates. Loop detection is therefore a necessity.

When historical information of an encounter is not available, their assumed rendezvous probability is set to zero. The protocol also has the capability of switching to a VR [44] mode when it cannot predict the destination movement range. The built in mechanism of HVR also supports conditional replication as a way to safeguard against the local maximum problem.One of the prime concerns with HVR is its inability to deal with situations where the transmission range of neighbour nodes never overlaps the destinations' predicted area in view of the long distance that sets them apart [118]; i.e. never encounter each other. This negatively affects the protocol's routing decision and its performance [118]; i.e. the protocol cannot switch to VR mode nor can it transfer the packet custody to any of its neighbours when destination information is available yet overlapping areas do not exist. In addition, the scalability of HVR is questionable in congested networks [118]. The complexity of computation and the growth of historical vectors in crowded environments will only deplete network resources and introduce more overhead. Moreover, slow mobility events emphasise on the mismatched absolute location issue mentioned in a previous section; with slow movement, the predicted destination area will likely be insignificant, thus, a mismatch between historical information and the current destination position might lead to inaccurate routing decisions especially when localisation errors exist. The absence of destination information results in operating in VR mode, which can lead to the increase of network overhead in overpopulated scenarios.

Similarly, *Geographic Based Spray and Relay (GSaR)* [14] relies on historical information (node speed, movement direction and recorded time of encounters) and replication to relay packets to their final recipient. However, unlike HVR [45], GSaR predefines the permissible quantity for replication (L) based on the specific network deployment; a network with sufficient mobility requires a small L value (a small number of replicas)

and vice versa. As soon as an encounter takes place, historical information is exchanged, updated and the destination movement area is estimated. The protocol then decides on the creditability of a candidate via a qualification process that exploits a set of conditions alongside collected information. The information consists of the encounter node's location, speed, direction, the deduced time (i.e. the total time required to reach the predicted area) and distance (i.e. the measured distance between the node and its destination). GSaR selects relays that can either contribute to the reduction of delivery delay, routing overhead or to the improvement of delivery probability [118]. The protocol aims to achieve these objectives by transferring the packets' custody to relays that are rapidly moving towards the predicted destination area, while rejecting the ones moving away from it [118]. The moment the packet reaches the identified area, GSaR then adjust its routing approach in a way that maintains replication within this range. In cases where the destination information is inaccessible, replication is used to improve the chances of delivery; replication in this phase operates in a similar fashion to VR, where the speed and direction (encounter angle) of encountered nodes are considered in the relay selection procedure. Although GSaR is more reliable than HVR [45], it still inherits most of HVR limitations; in spite of HVR limitations, the only excluded disadvantage is the one that is related to the calculation of the overlapping area. GSaR suffers from scalability and absolute location mismatch issues within crowded networks.

The *Weak State Router (WSR)* [2] was designed for intermittent and sparse environments, and forwards packets based on probabilistic hints inferred from historically recorded information. Here, each node maintains a view of the network in the form of regions (i.e. a region consists of a Bloom filter that maps node identifiers to an area), promoting WSR efficiency and scalability. The foundation of any region is the location information

received from beacons (i.e. control message that carry information about encountered nodes: speed and location) or announcements (small packets that contain node information and are routed towards randomly generated directions). In particular, nodes try to integrate the acquired information with geographically related regions, which expands the region radius and knowledge about containing devices. Contrary, the remoteness of obtained geographical information from a region prevents their integration. As a result, the recipient creates a region that represents the new set of information. Taking into consideration that information ages over time, the protocol supports a spatial and temporal decay mechanism to demonstrate the region's age and confidence. Once the strength is below a predefined threshold, the region is rendered useless and discarded. Routing in WSR is based on the assumption that accuracy and availability of information increases as packets get closer to their destination. Thus, the protocol always strives to route packets towards the strongest region available (see Figure 2.6), which is determined by its spatial and sentiment (i.e temporal) strength. When the source node has no information on a destination, WSR biases the packet towards a randomly selected direction, in an attempt to increase the chances of encountering an intermediate node with better information or even meet the destination.

GeoHawk is inspired from WSR as it provides a key solution to network scalability and overcomes limitations of protocols mentioned above. In contrast to GeoHawk, WSR cannot operate efficiently in dense opportunistic environments nor can it handle the associated consequences of localisation errors. WSR is tailored for sparse networks with high mobility and support for large transmission ranges, which renders it inapplicable to our targeted network environments. WSR's routing is based on unicast forwarding, which severely decreases the delivery chances of messages but improves battery life. The setback of probabilistic routing is that there is a possibility in which packets end up reaching a

false positive region and remaining within its defined boundary until they expire. The mismatch issue is even magnified with localisation errors. Furthermore, without any form of replication, delivery probability is likely to drop while delivery latency is expected to increase. This is attributed to the fact that searching for a specific recipient within a very crowded environment is a very challenging problem which, we believe, can only be solved by flooding (see Chapter 4). On top of this, periodically announcing single nodal information will affect the speed of state convergence, leaving nodes with no or little information on numerous parts of the network. As a result, node discovery is severely hindered limiting the effectiveness of the protocol in crowded network environments.

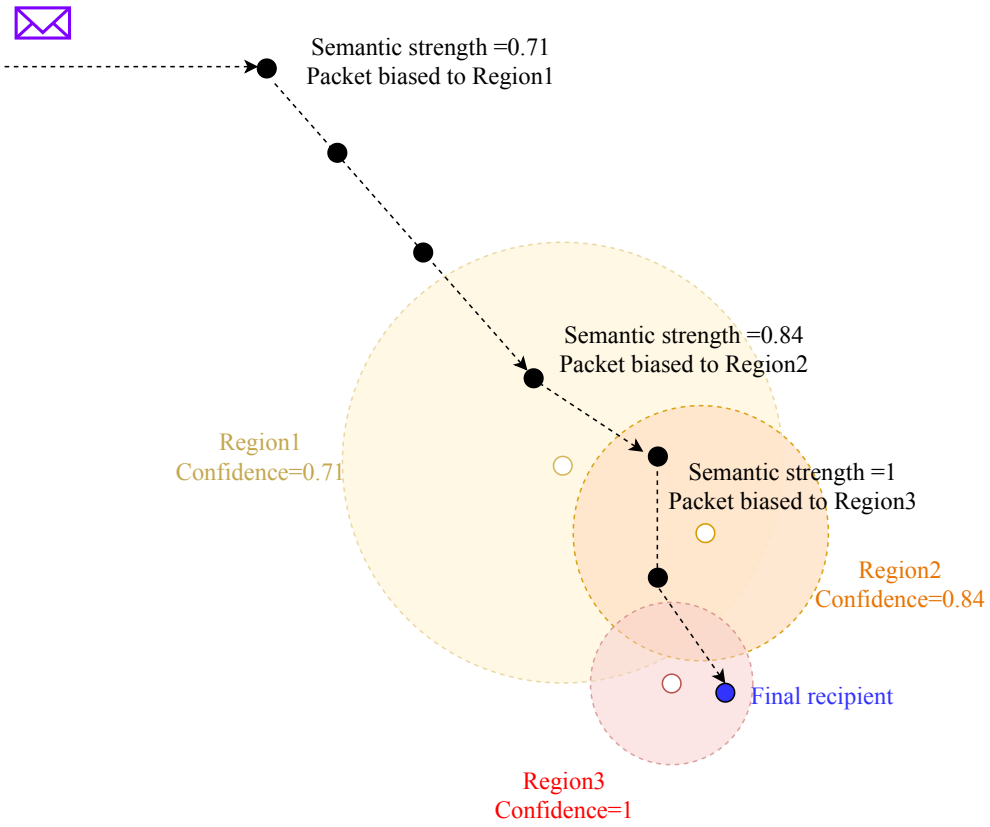


Figure 2.6 WSR protocol.

In Table 2.1,2.2 and 2.3, we summarise key features and limitations of protocols discussed above. It becomes clear that these protocols are not designed for very dense

opportunistic networks, therefore in in the following sections we only focus on the relevant, state of the art protocols.

Table 2.1 Position-based protocols summery (part1) [118].

Protocol	Application scenario	Dest. state	Knowledge of Dest. location prior routing	Localisation	Relay policy	Knowledge for routing decision	Limitations
VR [44]	VANETs	Move	Unessential	GPS	Replication	Encounter angle ω and speed	Results in redundant replication
RoRo-LT [110]	PSNs	Move	Unessential	GPS	Replication	Distance	Predictions from long term observation is an unrealistic approach for one-off events
MOVE [118]	VANETs	Static	Essential	GPS	Forwarding	Direction, distance	Suitable for environments with static destinations and an accessible global positioning system
GeOpps [56]	VANETs	Static	Essential	GPS	Forwarding	Selection of the NP, distance, speed	1-Protocol cannot scale 2-Requires heavyweight and complex computations
AeroRP [84]	ANs	Static	Essential	Proximity-based location	Forwarding	Direction, distance, speed	1-Limitation in routing technique 2-Inherits GeOpps limitations
DGR [13]	VANETs	Static	Essential	GPS	Replication	Direction, distance, speed	1-The accuracy of the recorded metrics can affect the protocol's performance when routines are unpredictable and localisation is coupled with errors 2- Inherits AeroRP limitations
MPAD [129]	UWSNs	Static	Essential	GPS	Replication	Communication angle	The Blind and unnecessary replication of nearly expired packets and the search of a single destination are not suitable for crowded networks with unreliable location information

Table 2.2 Position-based protocols summery (part2) [118].

Protocol	Application scenario	Dest. state	Knowledge of Dest. location prior routing	Localisation	Relay policy	Knowledge for routing decision	Limitations
DAER [34]	VANETs	Move	Essential	GPS	Replication	Distance	1-Exchanging real time information contributes to communication and routing delays 2- The replication mechanism may end up flooding the network in a congested setting 3-Many scenarios cannot support access to a server
POR [58]	VANETs	Move	Essential	GPS	Replication	Distance, message size	Inherits DAERs' limitations
LAROD [53]	ANs	Move	Essential	LoDiS system	Replication	Distance, replication area	Its reliability heavily depends on the accuracy and validity of the recorded information
GeoDTN [61]	PSNs	Move	Essential	GPS	Replication	Distance, movement areas	1-Limitation in routing technique 2-Faces the issue of dealing/maintaining large tables 2-Inherits LARODs' limitation
AaR [12]	VANETs	Move	Essential	GPS	Replication	speed, direction, speed, movement area	1-Faces scalability and location mismatch issues 2-Does not provide any mechanism to guard against localisation errors

Table 2.3 Position-based protocols summery (part3) [118].

Protocol	Application scenario	Dest. state	Knowledge of Dest. location prior routing	Localisation	Relay policy	Knowledge for routing decision	Limitations
GPSR [47]	VANETs	Move	Essential	GPS	Forwarding	Distance, encounter angle	1-Performs poorly in the absence of destination information and movement 2-Suffers from location mismatch issues when localisation errors are present
HVR [45]	VANETs	Move	Preferable	Unspecified	Replication	Speed, direction, movement area, encounter angle	1-Limitation in routing technique 2-Faces scalability and location mismatch issues 3-Requires heavyweight and complex computations
GSaR [14]	VANETs	Move	Preferable	GPS	Replication	Speed, movement direction and recorded time of encounters	1-Suffers from scalability and absolute location mismatch issues within crowded networks 2-Inherits most of HVR limitations
WSR [2]	VANETs	Move	Preferable	GPS, omnidirectional antennas	Forwarding	Distance, encounter angle, moving area	1-Routing is based on unicast forwarding which affects the delivery opportunities and latency of a packet 2-Tailored for sparse high mobile networks with large transmission ranges 3-The mismatch issue is magnified within dense settings and/or localisation errors

2.3 Localisation

Location-aware systems and geographic network protocols [11] rely on estimating the location of the device they operate on. There are different ways to express a location. An *absolute* location is specified using an absolute value (e.g. geographical coordinates) [52][46]. A *relative* location is specified in relation to the position of a known anchor; e.g. an access point (AP) [52][46]. A *symbolic* location is described through an abstract name; e.g. living room, home, work. [52][46]. Absolute locations are commonly used internally in systems and routing protocols, while relative and symbolic locations are used by applications that require human reasoning [52]. Absolute values can be mapped to relative or symbolic ones to satisfy application-specific requirements.

In this thesis we focus on absolute location and its estimation in outdoors and indoors environments. There exist different ways of estimating the location of a device, which can be classified as follows: (1) in *local systems*, devices calculate their location locally, e.g. using GPS [52][46]; (2) in *centralised systems*, devices offload location estimation to a centralised system, which increases network overhead and decreases CPU overhead and relevant energy consumption (e.g. active badge systems [52][46]); (3) in *distributed/collaborative systems*, devices and/or network infrastructure collaboratively calculate devices' locations by exchanging messages [52][23]. Hybrids of the above approaches also exist.

2.3.1 Dead Reckoning

Dead Reckoning (DR) uses inertial sensors (e.g. accelerometer, odometer and gyroscope) to calculate a device's speed and direction, and extrapolate its current position, given

its previous known position [52]. Dead Reckoning can provide a rough estimate when no alternative is available. Usually this approach is combined with other techniques to refine the estimation, as errors accumulate in time [52]. Pedestrian Dead Reckoning (PDR) inherits DRs' characteristics and is tailored for pedestrians. PDR is based on readings from an accelerometer to estimate step count and orientation (moving direction) of the device. The position is then extrapolated from the previous one [105]. PDR has been widely adopted in the context of mobile device localisation [91][38][89][102][105].

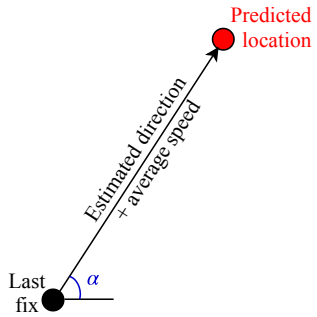


Figure 2.7 Dead Reckoning example: the last fix (black circle), speed and direction are used to extrapolate the new position (red circle)

2.3.2 Proximity-based Localisation

In proximity-based localisation, a device identifies its own location as the one of well-known anchors (e.g. access points) when they come into their range. The accuracy of this estimation depends on the range of the underlying wireless technology (e.g. Bluetooth, WiFi or LTE) that is used to identify the anchors [95][52]. ARTOS [68] and Smart Floor [3] deploy RFID tags and sensors respectively to physically locate a device, while approaches like [8][33], utilise in range anchors to approximate location.

2.3.3 Lateration

With lateration, a device's location is estimated using the (known) position of a number of reference points and the device's distance from these points. The required amount of anchors necessary for lateration depends on the dimensionality of the space (as shown in Figure 2.8); e.g. a 2D plane requires at least three anchors, while a 3D plane requires a minimum of four [95][52][46]. The distance to an anchor is calculated using one of the following methods.

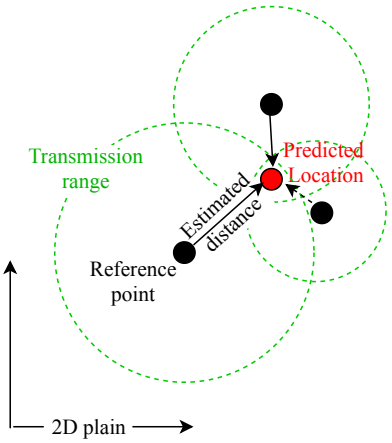


Figure 2.8 Illustration of lateration in a 2D plane. Three anchors (coloured in black) are used to identify the location of a mobile device (coloured in red) through the estimated distance.

Time Of Flight/Arrival (TOA). The distance between a device and an anchor is inferred based on knowledge about the characteristics of the physical medium and the time it takes to transmit a specific amount of data [52]. TOA requires clock synchronisation on both sides (which can be difficult to achieve and expensive in terms of energy consumption), as the accumulation of small time skews leads to high errors in the distance estimation [52]. Other approaches measure the distance using the round trip time [52].

Time Difference Of Arrival (TDOA). Unlike TOA, the time difference of arrival used within the lateration technique overcomes the synchronisation problem [46]. With TDOA

a sender emits two types of signals with different speeds at the same time (ultrasonic and radio waves), once the receiver receives the first signal (radio wave), it uses this to measure the time duration of the second signal arrival, in order to provide the system with a better precision[52][46]. The drawback of this approach is the need of two signalling technologies, which makes it expensive to deploy [46].

Signal Strength Attenuation. This approach is based on estimating the distance by measuring the reduction of the received signal strength [52]. As a signal propagates in space, its strength decreases [52][46]. There are other attenuation factors such as land topography, obstacles, equipment-related (antenna position, height and strength) and the characteristics of the transmission medium that contribute to the weakening of a signal [52][46]; e.g. the attenuation factor of a Radio Frequency (RF) in a free space, where the distance between a device and an anchor (r) is large, is $1/r^2$, a larger attenuation factor in a near-field $1/r^6$ would indicate the presence of obstacles [52]. Signals within indoor or complex environments are most likely to suffer from reflection, refraction and multi-path packet arrival [52][46], rendering this approach problematic for an opportunistic networking environment.

The Active Bat system[121] uses lateration and TDOA to identify users' positions. It does so by tracking the location of an end-user using ultrasonic and infra-red signals; the sensors and a centralised server emit RF signals periodically [121]. Similarly, to the Active Bat system, Cricket [85] processes RF signals but in a decentralised manner. AHP [77] operates in the absence of anchors. It adapts its functionality from the Distance Vector protocol, which uses intermediate nodes to measure the device distance to an anchor [77].

Requirements such as the presence of extensive network infrastructure, advanced sensing equipment and clock synchronisation, render lateration impractical for opportunistic

network environments. A self-localising algorithm designed for ad-hoc networks is described in [15]. The authors propose the use of TOA along with omnidirectional antenna to detect and estimate the distance to neighbouring nodes. Location information is exchanged among neighbours allowing a node to build a local coordinate system and predict its position [15]. This approach relies on network infrastructure and requires an omnidirectional antenna to perform the necessary computations, which makes it impractical for the targeted network environments. Moreover, signal strength would be affected by the network density, which would, in turn, affect the outcome of the localisation.

2.3.4 Triangulation

The concept of Angle of Arrival (AOA) is the foundation of triangulation [95][52][46]. In a 2D plane, AOA requires at least two reference points equipped with either directional or array antenna [95][52][46]. Once the reference points measure the angles of the received signals, the generated lines' intersection yields the position of the device [95][52][46]. UbiSense is a centralised system equipped with an array of antenna that emit ultra-wideband (UWB) and RF signals and uses AOA along with TDOA to discover the direction and distance to a sensor [52]. The directionality-based location scheme in [74] also relies on triangulation for location identification. Triangulation is not applicable to our network environment, which is very dense, opportunistic and mobile.

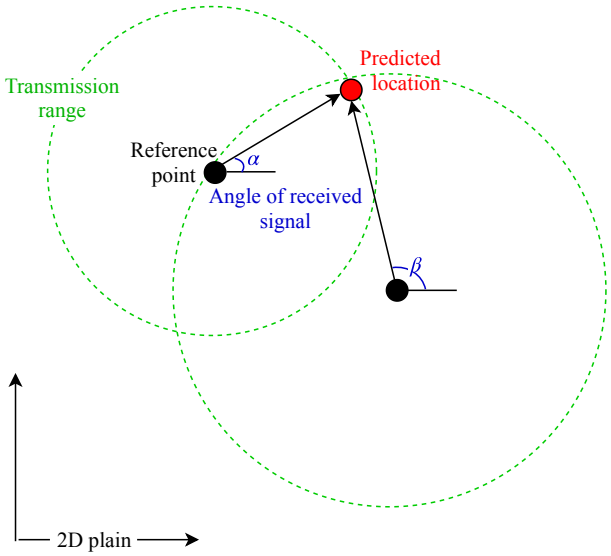


Figure 2.9 Example of triangulation in a 2D plane.Two anchors (coloured in black) are used to identify the location of a mobile device (coloured in red) by measuring the AOA of their signals.

2.3.5 Fingerprinting

Fingerprinting is based on the concept of pattern recognition (pattern-matching process) [52][46]. Initially, the system undergoes a training phase, the outcome of which is a mapping of locations to specific signal strength patterns [52][46]. The distance granularity between locations should be planned in order to attain satisfactory accuracy [52][6]. The pattern-matching process can be processed either locally on the client side or at a centralised system [52][46]. To resolve the location of a specific device during the operating phase, the received signal strengths are compared against the stored information, and the closest match (location) is then returned back to the device (either locally or centrally by a server). One of the earliest fingerprinting systems is the active badge system [121], in which a server maps sensors' positions to corresponding signal strengths. A badge carried by

an end user emits infra-red signals and the sensor receiving the signal communicates this information back to the central server in order to identify the badge position [121]. Fingerprinting is also at the heart of systems proposed in [81][6][55][64][7][41][128][70]. HVAC is another form of fingerprinting, which measures the change in air pressure instead of signal strengths and processes this information to classify the devices' position and motion [80]. Fingerprinting is problematic in environments where network infrastructure does not exist or is partially deployed. The existence of obstacles or high user density also severely limits the applicability of fingerprinting as these factors affect the signal strength [77].

2.3.6 Scene analysis

Scene analysis relies on image recognition. A device captures an image of the surrounding environment and identifies its location using blob detection [52] or simply through landmark observation [46]. Image processing is widely used for localisation in the field of robotics [103][9]. The drawback of this approach is it is computationally expensive (and therefore energy hungry). There also obvious privacy implications. It would also be impractical to apply scene analysis in our network environment given the density of users and devices (i.e. larger objects can obstruct the view) and [46].

2.3.7 Collaborative Localisation

A number of approaches that provide location services in a collaborative fashion have been proposed in the literature. The systems described in [54] [78] combine fingerprinting with lateration and particle filtering, respectively. Lateration and proximity-based localisation

using neighbours' positions are integrated in the system described in [94]. Said approaches suffer from limitations related to lateration and proximity-based localisation, as described in the previous sub-sections.

With respect to indoors localisation, the system described in [92] employs a sequential Monte Carlo filtering approach for localisation [92], in an attempt to reduce the noise from RSSI fluctuation, by combining the received signal strengths, using a particle filter. The system would not be practical in our network set-up given that it is suitable for deployments in small areas without obstacles and with low user density. The approach described in [18] is similar to [92] except that it considers the presence of density and group clusters. Moreover, the paper has reported that signal strengths are affected by the density of users in the targeted area; the denser the area, the more fluctuating a signal strength is. As a result, they combined a fingerprinting system along with a filtering technique, while correcting the prediction using proximity of neighbouring nodes. Another indoors localisation system is described in [116]. This system improves the outcome of the fingerprinting technique using particle filtering. Nevertheless, these systems require fixed infrastructure and would not be applicable in crowded network environments without any network infrastructure, where communication delays can be high, and devices are equipped with regular sensors. Training the fingerprinting subsystem in such an environment would be impossible.

Collaborative localisation has been a prominent research area within the broader robotics research field, as robots require location information to navigate in known and unknown spaces. The localisation system presented in [90] relies on proximity to compensate for the prediction errors of the particle filter-based observation phase. Robots use various sensors to identify other team members and to estimate their relative distance and orientation.

The paper reports robustness towards 10% of odometry skews [6]. Ignoring the limited capabilities of a mobile device, localisation with proximity and filtering techniques is a good solution to environments that lack a network infrastructure. The authors in [86] assume no prior knowledge of a robots' position and rely only on self-localisation sensors (observation phase). The system processes images (obtained from a mounted camera), detects the range and bearing to other robots and uses this information to correct the robot's location estimation (measurement model) [86]. Beliefs are then exchanged to enable the system to draw samples from both beliefs [86]. Similarly, the paper in [28] makes a further assumption in which Dead Reckoning and environment measurements are used in its observation phase. The systems' beliefs are corrected in the same manner (using image processing) and re-sampled normally. These systems are closely related to the method we propose; we exclude image processing which is impractical in our problem space and adapt various other components to our large-scale, high-density opportunistic network environment and in the context of opportunistic network routing. Both systems derive confidence about the estimated location from the spread of the particles [7], while we treat the confidence as a separate metric; we re-weight particles against the collaboratively estimated mean and not directly with a neighbours' reading. Investigating a sparse DTN environment, the paper [92] attempts to localise stationary sensors using robots' mobility, signal strengths and extended Kalman filters, however the set-up is very different to ours, hence this approach would not applicable to our network environment.

Integrating collaborative localisation to pedestrian dead reckoning has also been researched. In [57] the authors describe how step counts can be deduced from accelerometers

[6]The paper in [90] deploys and odometer for each robot to enable them to detect and estimate their location. However, like DR 2.3.1, using odometers for long periods results in accumulated error, yet the proposed approach enabled the robots to withstand approximately 10% of this error.

[7]Each location estimation (hypothesis) associated with a confidence (weight) is referred to as a particle (See section 3.4.3 for more detail).

and compass readings and how landmarks and GPS readings help in smoothing PDR errors. The system relies on a Markovian model, which is trained to predict a device's position from its historically observed movement and activities [57]. Such a model is not practical for our problem set-up since the existence of historical movement data for a specific user cannot be assumed; indeed, in many of the use cases we are interested in, users may visit specific large and crowded areas or events only once. Proximity is also used by [43] to help in putting a bound in the respective PDR' error. Once a node encounters its neighbour, they exchange their location information; this information is then merged to calculate the estimated new location by simply averaging the two estimations. This system is extended in [42], by allocating a confidence parameter to the current location estimate. The confidence reflects the uncertainty of the step size and inaccuracies of PDR. Here, averaging is not only performed on the exchanged location but also on the obtained confidence [42]. These approaches do not distinguish between new and repeated encounter types, always attempting to integrate received information leading to a trade-off between meeting frequency and estimation precision. The collaborative PDR discussed in [50] uses a bivariate normal distribution to estimate nodes' location and confidence (in a 2-dimensional space); i.e. the distribution mean represents the estimated location and its variance represents the node's confidence about the estimation. Proximity is also used to bound PDR's error but contrary to the averaging formula, this system puts more weight on the information with higher confidence [50]. In an attempt to maintain precision, the system places an integration restriction on information obtained from previously encountered nodes [50]. Our approach is inspired by this system, however we integrate particle filtering and accurate location readings (e.g. through GPS and indoors beacons) so that we can maintain a pre-defined estimation error bound. LOCALE [125] is another

approach that uses a distribution to model the location (and confidence about it) of a device. However, LOCALE introduces an additional step (transformation step) before linearly combining local and received information into a single representation (proximity)[125]. The transformation step tries to bridge the gap between the local and neighbours' views through a series of sophisticated computations[125]. LOCALE is designed for sparse DTN networks therefore it would not be applicable to our problem space.

Table 2.4 summarises the discussed localisation techniques. From this we can deduce that the collaborative localisation is the best suitable approach for our environment provided that a good combination is selected (see section 3.4 for more details).

Table 2.4 Localisation systems summery. [118].

Location system	Equipment	Knowledge for localisation	Limitations
DR	Inertial sensors (e.g. accelerometer, odometer and gyroscope)	Speed, direction	Errors accumulate in time
Proximity-based Localisation	Known anchors (e.g. access points)	Anchors' positions	1-Requires infrastructure 2-The accuracy of the resulted estimation depends on the underlying wireless technology used to identify the anchors
Lateration	Reference points, and/or (synchronized clocks, or two types of signals)	Location of reference points, distance	1- Clock synchronization can lead to error accumulation 2-It is expensive to deploy two singling technologies 3-complex environments may result in attenuation of signals strength
AOA	Reference points equipped with either directional or array antenna	The angles of received signal strength	1-Unsupported capability by mobile devices 2-The location of reference points may not be accurate or accessible in dense, opportunistic and mobile scenarios
Fingerprinting	A local/globe system that can perform the pattern-matching process	Measurable and distinctive factors (e.g. signal strength and air pressure)	1-Requires infrastructure 2-The presence of obstacles will lead to signal attenuation
Scene analysis	Camera	Image	1-Computationally expensive and energy hungry 2-Privacy implications 3-Large objects can obstruct the view
Collaborative localisation	Any combination of the above techniques	Inherits the same combination of inputs	Inherits the limitation of the adopted approaches. However, the combination can mitigate limitations or enhance the expected outcome

Chapter 3

Routing in Dense Opportunistic Networks

In this chapter we present the design of GeoHawk, a unicast routing protocol that efficiently routes information to mobile devices which are part of a very dense network. Devices connect to and exchange data among each other opportunistically, whenever they come into each other's communication range. Each message in the network is destined to a specific device whose identifier is carried in its header. Apart from the destination identifier, each message also carries information about the 'destination' region, a circular area into which the destination device is believed to reside. Although the destination identifier never changes, a 'destination region' does change when intermediate devices have 'better' information about where the destination device is. 'Better' here refers to more accurate and/or recent information with respect to the location of the device.

The proposed protocol consists of two major components that deal with (1) how routing information is communicated and maintained by each mobile device and (2) how messages are forwarded in the network when devices connect to each other opportunistically. Our

protocol builds on the work proposed in [2] and incorporates the notion of Geo-Regions and Weak Bloom Filters (WBFs) (see Section 3.1 for a brief description of these concepts). GeoHawk differs significantly from the protocol presented in [2], as follows:

- Routing information is disseminated globally across the whole network (see Section 3.2) in announcements that are forwarded to randomly selected directions [1]. Upon its creation, an announcement contains aggregated routing information relevant to its source (a region along with node membership), which is further aggregated with regions stored in nodes that process the announcement as it is being propagated in the network (see Section 3.2.3). This way, devices learn how to start forwarding data to remote parts of the network. This is in contrast to [2] where nodes announce only their own location and no aggregation takes place. Globally disseminating announcements is crucial in a very densely crowded network where users move about slowly so that devices can get routing information about remote nodes in a timely fashion.

- In our work, mobile devices may not have accurate information about their own location. For example, location may be calculated in a collaborative fashion, as described in Section 3.4. GeoHawk deals with this uncertainty by employing varying region radii that reflect the confidence nodes have about their own location estimation. Region aggregation (see Section 3.2) also takes into account the uncertainty of the location of all nodes that are being aggregated.

- The temporal decay in our protocol is triggered through a timer and a maximum lifetime associated with all regions, which is in contrast to WSR [2] where regions

[1] Understanding the inevitable trade-off mentioned in Section 4.4 makes it possible to select a favourable frequency rate at which these announcements are generated.

are decayed either when the processing node resides within the region or when the mapped Bloom filter reaches its cardinality limit. Using the users' position to trigger temporal decay in a dense movement environment can lead regions to enter this mode at an early stage, leading to the loss of information.

- GeoHawk assesses how likely it is for the destination device to reside within the destination region using pre-specified threshold values on specific metrics (see Section 3.3.2). This contributes in reducing network overhead due to unnecessary flooding (see point below).

- When a message makes it to its destination area, it gets flooded within it, if the probability that the destination device actually resides in that area is high. This is decided based on how close the node that biased the message towards the destination region was (see Section 3.3.2). A Time-To-Live field is used to prevent messages from being routed in the network indefinitely.

- GeoHawk supports a packet redirection mechanism, which enables sending a message towards a different region once reaching the destination region and the protocol assesses that the destination device is unlikely to be in the current region (thus preventing unnecessary flooding).

- GeoHawk supports a ticketing mechanism as a means to increase delivery probability when credible information about the destination device points to different directions. This significantly increases delivery ratio and decreases latency without significantly affecting the induced network overhead.

3.1 Geographical Regions and Weak Bloom Filters

In our work we incorporate the notion of weak state, as defined in [2]. A piece of routing state consists of a circular region in the network (defined as a point and radius) and a weak Bloom filter [2]. The Bloom filter is used to test membership of a node identifier in the set of nodes that are believed to reside within the respective region. Both parts of this state are weak and decayed to a point where the state is discarded. Regions are spatially decayed by expanding their radius according to a pre-defined rate so that they can be valid for longer given that nodes move. Spatial decaying is also triggered when a new routing state is received and aggregated with locally stored one and is done by increasing the radius of a region, when it is merged with other regions (see Section 3.2.1). Temporal decaying is triggered through a timer and a maximum lifetime associated with all regions and is done by flipping 1s to 0s in the Bloom filter, according to a pre-specified weakening interval; i.e. at each decaying instant, a predetermined probability (refereed to as decay probability) is used to reset bits within a bit set. Because of that, membership testing may not always yield a true (which in any case may be a false positive) or false value; instead, membership testing yields a probability that a specific node identifier has been inserted in the Bloom filter. For example, if k hash functions are used for constructing Bloom filters and, at a specific point, n of the k bits (where $n \leq k$) for a specific device identifier have a value of 1, then said probability is $\frac{n}{k}$. Associated with the weakening of the Bloom filter is the temporal strength of the routing state for a destination node, the number of 1s as a result, of testing membership of that node in the Bloom filter[2].

[2]Note that the spatial strength refers to the routing state as a whole, whereas the temporal one is for a specific destination device.

When the decaying process is triggered, the routing state cannot be aggregated with any other routing state. More specifically, the respective region cannot be merged with other regions nor new node identifiers can be added to it. Decaying cannot be reversed, and the routing state will be discarded when a low threshold on the Bloom filter cardinality is reached. When the temporal decay process results in a number of bits (i.e number of 1') below the cardinality threshold, stale and useless routing information is discarded.

3.2 Building Routing Tables

Mobile devices in GeoHawk build and update their own routing tables by exchanging information with other encountered devices in an opportunistic and completely distributed fashion. Routing information is exchanged in the form of announcements that traverse the network and direct messages that are exchanged when devices encounter each other. GeoHawk does not assume any prior knowledge of the location of mobile devices in the network (i.e. a GeoHawk node will learn about other nodes upon encountering other nodes or receiving aggregated routing information) nor any centralised 'oracle' that communicates the location of mobile nodes upon request. Moreover, we assume that nodes know their own location but this information may not be accurate and, in addition, degrade over time. More specifically, GeoHawk is designed to be compatible with the collaborative localisation approach described in Section 3.4 and the uncertainty of the location estimation in the notion of geo-regions; i.e. larger radii denote less certainty about both the location of the node in its centre and the location of other nodes that reside within the same region, as these are specified in the Bloom filter that is paired with the region.

Given that our target environment is a very dense one in terms of mobile devices (and users), maintaining routing information about each device separately would be completely unrealistic, in terms of required network bandwidth and memory to communicate and store mappings of nodes to locations, respectively. Instead, we aggregate routing information by associating group of devices (represented as Weak Bloom Filters [2]) to specific geographical regions (Section 3.4.3 explains Weak Bloom filters in detail). Location information is exchanged, and new regions are created independently by mobile devices either through direct information exchange between neighbours or by disseminating location announcements in the network.

3.2.1 Aggregating Routing State

Mobile devices aggregate routing state that they locally store with state disseminated in the network. State is exchanged directly through control messages between two neighbouring devices upon encounter (see Section 3.2.2), and through advertisements that are initiated by a single source node and forwarded towards a fixed and randomly selected direction (see Section 3.2.3). Note that the content of advertisements is also aggregated with locally stored state, but this is described in Section 3.2.3.

Region aggregation is shown in Algorithm 1. Upon receiving new routing state, a mobile device attempts to aggregate it with an eligible entry stored in its routing table (*Lines* 1 *to* 11 of *Algorithm* 1). Routing entries that are currently being decayed cannot be aggregated any further and will be discarded soon (see *Line* 2 of *Algorithm* 1). For each non-decaying state, the mobile device then calculates the smallest enclosing circle for the regions defined in the locally stored state and the incoming one (*Line* 3 of *Algorithm* 1). If the device itself resides within this circle, aggregation is not allowed (*Line* 4 of *Algorithm*

1). This puts a limit on the spatial weakening that aggregation results in, ensuring that aggregated state will be useful for forwarding messages. Finally, the device calculates the number of 1s in the resulting Bloom filter, assuming that aggregation is done (by ORing the Bloom filters in the stored and incoming state) (*Line* 5 of *Algorithm* 1). If the cardinality of the (provisionally) aggregated Bloom filter is larger than a maximum threshold, then aggregation is aborted (*Lines* 6 *to* 9 of *Algorithm* 1).

If none of the locally stored entries can be aggregated with the incoming state (due to spatial or temporal constraint violations), the device creates a new routing entry, copying the region specification and Bloom filter from the incoming state (*Lines* 10 *to* 11 of *Algorithm* 1).

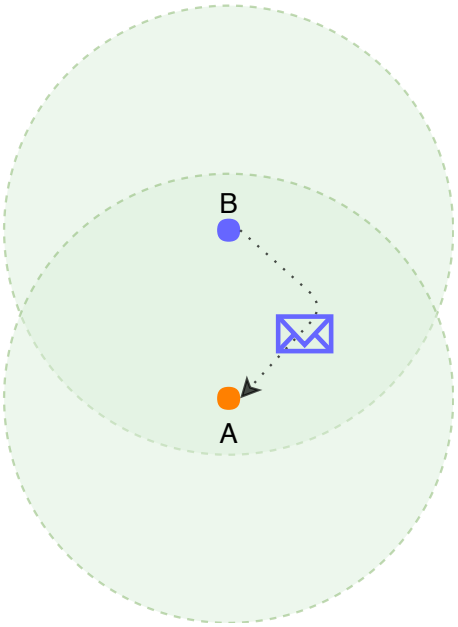
Algorithm 1: Region aggregation

Input: Region inR, Node curN; int counter;

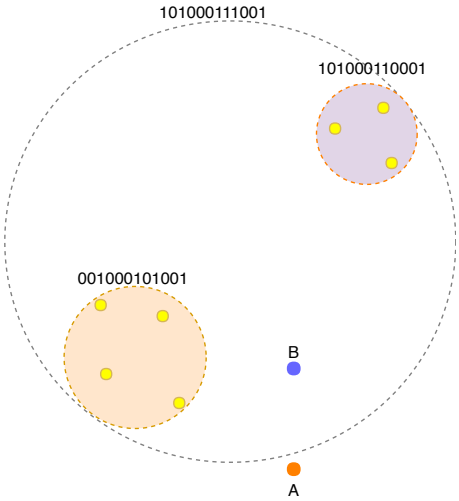
```
1  foreach Region E in curN.routingTable do
2      if E.isInDecayMode == false then
3          Circle C = computeSmallestEnclosingCircle(inR, E);
4          if curN.isInCircle(C) == false then
5              BloomFilter provisionalBF = inR.BF | E.BF;
6              if provisionalBF.cardinality < maxCardinality then
7                  E.BF = provisionalBF;
8                  E.circle = C; counter++;
9                  break;
10 if counter == 0 then
11     curN.routingTable.add(inR);
```

Figure 3.1 illustrates an example of aggregating routing state. Node *A* (shown in orange) receives a message as it encounters Node *B* (shown in purple), which can be either a control message that communicates *B*'s location (and confidence about it) or an advertisement that carries information about an aggregated region 3.1a). Figure 3.1b shows how aggregation is done at node *A*. Specifically, *A* integrates the newly received region (from an advertisement in this case) with an existing one (shown as the orange circle). The yellow dots in the region denote existing nodes whose identifiers have been previously inserted to the Bloom filter that is stored along with the information about its mapped region. Note that this is for illustration purposes only; the only membership information stored along with a specific region is the Bloom filter, therefore the exact nodes that reside in a region are not known. The Bloom filters of the two regions are ORed to form the Bloom filter of the aggregated region, as shown in Figure 3.1b.

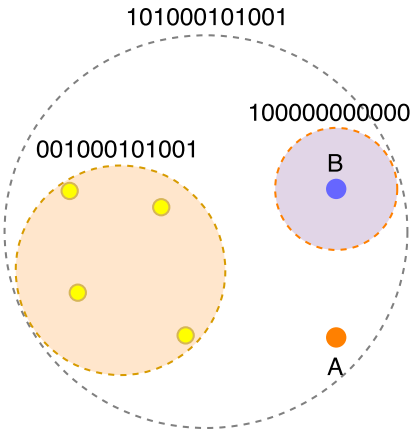
Figure 3.1c illustrates a scenario where *A* tries to integrate the newly created region from a control message. The aggregation is blocked due to violation of the spatial constraint; i.e. if aggregation went through, node *A* would end up residing in the aggregated region. The spatial constraint places a limit on the size of stored regions, the absence of which would result in very large regions that inevitably contain a very large number of mobile devices; apart from violating the Bloom filter cardinality constraint, very large regions would add little knowledge about the location of specific devices.



(a) **Node *A* receives a control message or announcement from node B.**



(b) **Node *A* successfully integrates the advertised region with a locally stored one.**



(c) **Aggregation of a control message with an existing region is aborted due to violating the spatial constraint.**

Figure 3.1 Aggregating incoming state with existing regions.

3.2.2 Direct Location Information Exchange

Mobile devices exchange information about their location when they encounter each other and build new regions in their local routing tables, in a way that is reminiscent to the creation of geo-regions in [2]. In contrast to [2], the initially created region is not just a

single point in the map; instead, we incorporate the location uncertainty, which is prevalent in the studied communication environment, by assigning a radius in the region. The radius reflects the confidence of the location estimation (see Section 3.4); the confidence value is carried along with the location information. [3]After creating a new region, each mobile device independently aggregates it with existing nearby regions that are stored locally, as discussed in Section 3.2.1.

3.2.3 Routing and Processing Announcements

Fast propagation of routing information is crucial for the targeted communication environment where the density is very high and mobility is relatively slow. Announcing aggregated information (as pairs of regions and Bloom filters representing node membership in these regions) is the only way for nodes to be able to efficiently send messages to devices that are far away in the network.

Each mobile device in the network periodically sends announcements that contain (potentially aggregated) routing information relevant to its own location; each message is forwarded towards a specific direction in the network which is randomly selected by the source node. This is done by selecting a specific point in the two-dimensional space so that the line between the mobile device and that point is on the randomly selected direction[4]. Messages that carry such announcements are destined to a specific location in the map and not to a specific mobile device. We employ GPSR [47] to efficiently route an announcement to its destination.

[3]When combined with the collaborative localisation approach described in Section 3.4, we set the radius based on the covariance matrix that defines the uncertainty of the location (and conversely the confidence for this information) so that the probability that the mobile device resides within the defined region is has a specific desired value (e.g. 0.9 in our experimental evaluation).

[4]Note that knowledge of the map is not required, therefore one could just select a point that is "out" of the map. We only care about the direction that the message will be propagated to.

We control the lifetime of messages through a hop-based Time-To-Live field that is carried in the announcement and decreased at each hop a message takes [88]; an announcement is discarded either when the TTL field drops to zero [88] or when the underlying GPSR protocol detects a loop when operating in perimeter mode [47].

As the announcement is propagated into the network, it gets aggregated with routing information that intermediate nodes store locally. The objective here is to provide mobile devices that are far from the source of the announcement with aggregated information about the location of nodes closer to the source device. Announcement propagation is done in a way that devices further in an announcement's path receive heavily aggregated information so that they can start the process of forwarding messages to specific destinations, as described in Section 3.3. Nodes closer to the source of an announcement have more accurate information about the area where the source node resides; they store smaller regions with more devoid Bloom filters. Consequently, messages are forwarded towards devices that hold increasingly more accurate information about the destination of messages. This is reminiscent to IP routing where in most cases the source node just knows of a default gateway to the router it is attached to. IP routers on the path to the destination hold more specific information about the destination, a property that is enabled by IP address aggregation and variable length masks.

Announcements are created at their source node where aggregation with locally stored routing state occurs. This is described in Algorithm 2.

Algorithm 2: Aggregation at the source node.

```
Input: Node srcNode;
1  Region r = new Region(srcNode.location,
   max(srcNode.confidence,transmissionRange));
2  r.BF = srcNode.ID;
3  foreach Neighbour i in transmissionRange do
4      BloomFilter provisionalBF = r.BF | i.ID;
5      if provisionalBF.cardinality <= maxCardinality then
6          r.BF = provisionalBF;
7      else
8          break;
9  if r.BF.cardinality <= maxCardinality then
10     foreach Region E in srcNode.routingTable do
11         if E.circle.isWithin(r.circle) then
12             BloomFilter provisionalBF = r.BF | e.BF;
13             if provisionalBF.cardinality <= maxCardinality then
14                 r.BF = provisionalBF;
15 Announcement a = new Announcement(r, TTL, direction);
```

The source node first creates a region containing only itself (*Line* 1 *to* 2 of *Algorithm* 2); the centre of the region is its estimated location (denoted to as srcNode.location). The radius of this region is set to be the maximum between the confidence (srcNode.confidence)

about the location (zero if GPS is used) and the associated transmission range. The source node then aggregates, one by one, all its neighbours (*Lines* 3 *to* 8 of *Algorithm* 2); i.e. the nodes that are within its transmission range. Aggregation stops if the provisional cardinality of the resulting Bloom filter (through ORing its current value with the identifier of the node that is being aggregated) exceeds the maximum allowed Bloom filter cardinality (*Lines* 5 *and* 8 of *Algorithm* 2). Provided that the maximum cardinality has not been reached, the node further attempts to aggregate regions that can fit within the currently defined one (*Lines* 9 *to* 14 of *Algorithm* 2).

If the Bloom filter cardinality is still smaller than the maximum allowed one after all neighbours are aggregated (*Line* 9 of *Algorithm* 2), the source node will attempt to aggregate the advertised region with regions that it locally stores (*Lines* 10 *to* 14 of *Algorithm* 2). Only regions that can fit within the currently defined region can be aggregated (*Line* 11 of *Algorithm* 2). This is to accommodate scenarios where the confidence about the node's location is low, and, therefore, the radius of the advertised location is large. In this case more information will be potentially aggregated. When GPS is used, this radius equals the transmission range, which means that no further aggregation can happen after aggregating the neighbouring nodes.

The advertisement is then forwarded to the next hop using GPSR (*Line* 15 of *Algorithm* 2). All nodes along the path of the advertisement, update their local state using the routing state carried in the advertisement, as described in Section 3.2.1. Subsequently, each of

these nodes will attempt to update the content of the advertisement through aggregation with their local state, as described in Algorithm 3.

Algorithm 3: Aggregation at intermediate nodes.

Input: Node n; Announcement a;

```
1  if a.region.BF.cardinality <= maxCardinality then
2      Region r = new Region(n.location, max(n.confidence,transmissionRange));
3      r.BF = n.ID;
4      Circle c = computeSmallestEnclosingCircle(a.region, r);
5      a.region.circle = c
6      a.region.BF = a.region.BF | n.ID;
7  if a.region.BF.cardinality <= maxCardinality then
8      foreach Neighbour i in transmissionRange do
9          BloomFilter provisionalBF = r.BF | i.ID;
10         if provisionalBF.cardinality <= maxCardinality then
11             r.BF = provisionalBF;
12         else
13             break;
14 if a.region.BF.cardinality <= maxCardinality then
15     foreach Region E in n.routingTable do
16         if E.circle.isWithin(a.region.circle) then
17             BloomFilter provisionalBF = a.region.BF | e.BF;
18             if provisionalBF.cardinality <= maxCardinality then
19                 a.region.BF = provisionalBF;
```

Intermediate nodes first check whether it is possible to further aggregate routing information in the current advertised region by looking at the cardinality constraint (*Line* 1 of *Algorithm* 3). If further aggregation is possible, the node creates a region containing itself (*Lines* 2 *and* 3 of *Algorithm* 3) with a radius that is equal to the maximum of the confidence and its transmission range (as discussed above). It then calculates the smallest enclosing circle (*Line* 4 of *Algorithm* 3) between the newly created region (r) and the one carried in the advertisement (a.region). This is now the new region, spatially-wise (*Line* 5 of *Algorithm* 3). The node then attempts to aggregate its neighbours into the spatially aggregated regions (*Lines* 7 *to* 13 of *Algorithm* 3). If the Bloom filter cardinality is still smaller than the maximum allowed one after all neighbours are aggregated (*Line* 14 of *Algorithm* 3), the source node will attempt to further aggregate regions that it locally stores (*Line* 15 *to* 19 of *Algorithm* 3). Only regions that can fit within the currently defined region can be aggregated, also taking into account the cardinality constraint (*Lines* 16 *and* 19 of *Algorithm* 3).

In the example shown in Figure 3.2, node *A* randomly selects a direction to forward an announcement using GPSR (Figure 3.2a). Initially, the announcement contains *A*'s location along with its estimation confidence represented as the radius of the region, as shown in Figure 3.2a. Source node *A* then attempts to perform the first form of aggregation, by integrating its own position with the positions of its neighbours (shown in yellow) and/or existing regions that can fit within this region (see Figure 3.2b). It replaces the announcement's initial content with the outcome of this integration. Assuming that the source node is confident about its own location, the maximum transmission range defines the region's radius; otherwise the maximum radius reflects the (low) confidence about its location. Integration is restricted by the maximum cardinality defined for the

Bloom filter and the defined region radius. Once integration is successful and available capacity (in terms of the maximum allowed cardinality) exists, *A* further tries to aggregate existing information (neighbours/ regions) that can be fitted within the defined area. When integration of potentially multiple regions is completed, *A* proceeds with forwarding the announcement to its next hop. The announcement is then received and processed by Node *B*. *B* aggregates its local neighbours, defines the maximum area (*B*s' transmission range or location confidence) and computes the new radius. The updated region is the smallest circle that contains both regions (see Figure 3.2c). *B* then looks up its routing table for regions (shown in green in Figure 3.2c) along with the respective Bloom filters that could potentially be aggregated to the advertised region, and attempts aggregation if none of the constraints are violated. In this example, *B* aggregates the initially advertised region with one of its locally stored regions that can be fitted with the smallest defined circle (as shown in Figure 3.2c). The updated Bloom filter is the result of ORing the respective Bloom filters, while the regions' radius remains unaffected. The announcement is then subsequently propagated to node *C* (shown in orange in Figure 3.2d), which performs the same steps.

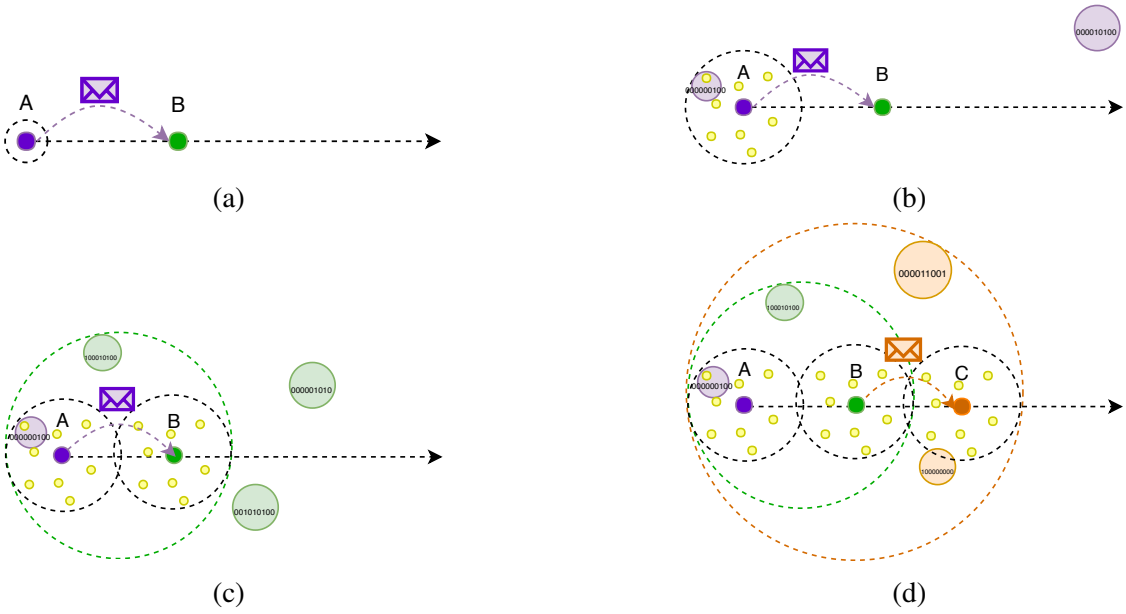


Figure 3.2 Announcing and aggregating regions - constraints are satisfied.

Figure 3.3 depicts a scenario where the Bloom filter constraint is violated. In the first step, Node *A* aggregates the region defined in the message with its own location and a locally stored region (shown in purple in Figure 3.3a), then relays the updated announcement to node *B*. *B* follows the same procedure and aggregates its local neighbours, then computes the smallest enclosing circle to define the new boundary of the aggregated regions. Once this phase is done, *B* searches it local table and tries to aggregate any region (shown in green in Figure 3.3b) within the newly defined area as long as the conditions are not violated. The updated announcement is forwarded to the next neighbouring node (*C*). After completing the first two blocks in Algorithm 3, *C* cannot aggregate the region defined in the announcement with the one stored locally at *C* (shown in orange and marked with an X), as this would violate the maximum cardinality constraint, as illustrated in Figure 3.3c. As a result, the advertised region in the message only integrates *C*s' neighbours and the announcement is further forwarded to node D, as depicted in Figure 3.3d. *D* can go on with region aggregations following the same procedure.

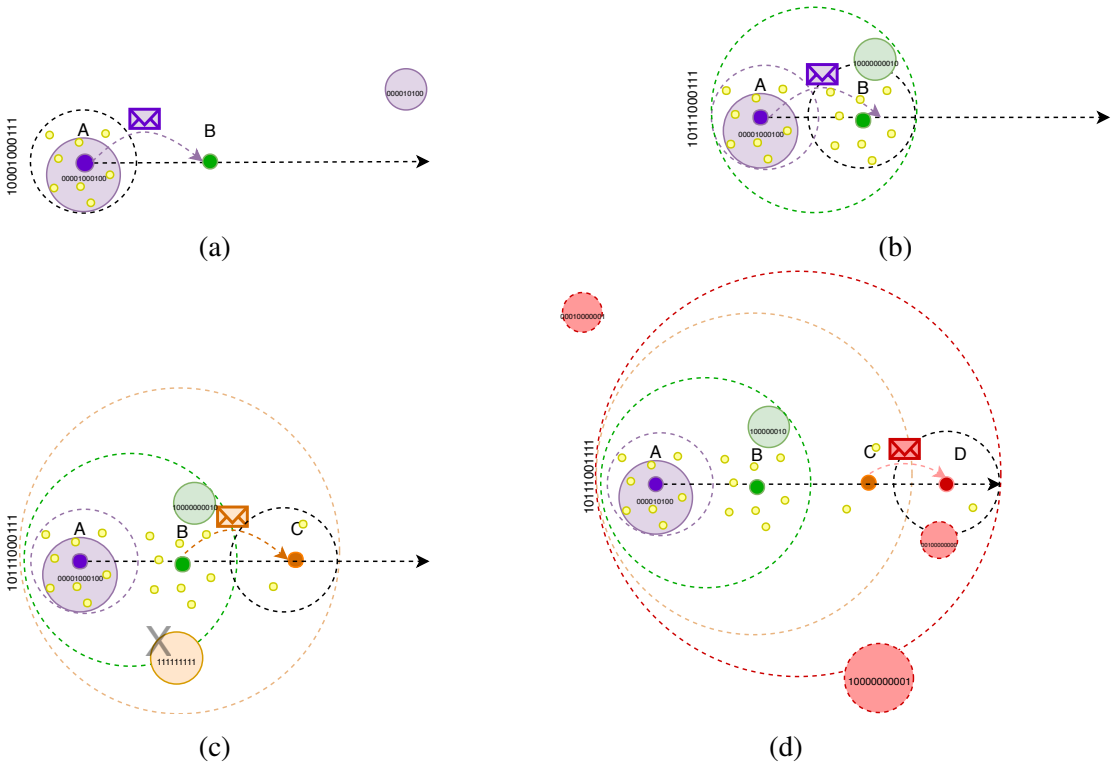


Figure 3.3 Announcing and aggregating regions - per-hop radius increment constraint violation.

3.3 Routing Messages

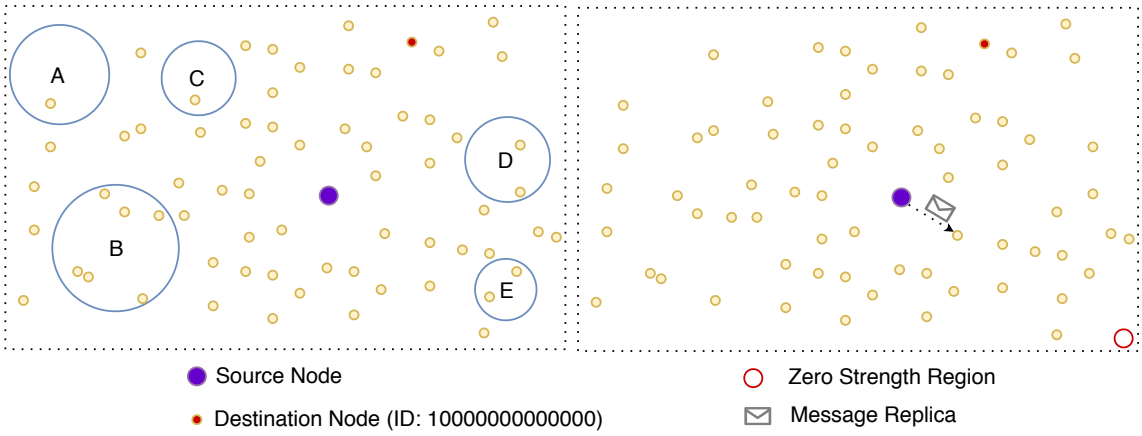
Message forwarding consists of two phases. In the first phase, a message is forwarded to one or more "destination" regions; i.e. regions where the destination device is believed to be residing. In the second phase the message is propagated within the destination region(s) until it reaches its destination (a specific mobile device whose identifier is included in the message).

Messages carry two counters; a *Hop* and a *Bias Freshness (BF)* counter. The Hop counter is increased at each node that processes a message. The BF counter denotes the number of hops since the last time the message path was altered by an intermediate node and is increased at each node that processes a message. Whenever the destination region

of a message is altered, as described below, the BF counter is reset back to zero. Both counters are used in the second phase of routing (see Section 3.3.2), after a message has reached its destination region. Each message also carries a *Time-To-Live* field that is used to eliminate the otherwise rare loops [2].

3.3.1 Routing a Message to its Destination Region

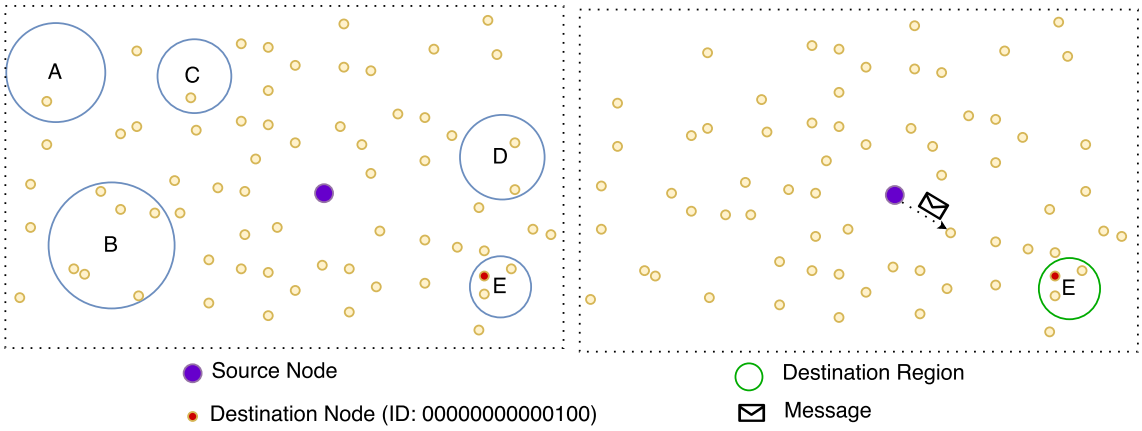
For a newly created message to be sent to a specific destination node, the source node iterates through its routing table and, for each entry, it tests whether the identifier of the destination node is a member of the associated Bloom filter; i.e. it checks whether the destination node is believed to be residing in one or more regions known to the source node. If no such region exists, the source node randomly selects a *zero strength* region in the network. *Zero strength* regions are not stored in the routing table and represent points in the map with zero radius along with all-zero Bloom filters (hence the zero spatial and temporal strength, respectively). The aim here is to increase coverage and consequently the probability that the message makes it to its destination, when the location of the destination node is unknown. This is illustrated in Figure 3.4, where the destination node identifier (10000000000000) tests negatively against all routing entries in the source node's routing entries. As a result, the source node forwards the message towards the shown (randomly selected) *zero strength* region. The assumption here is that nodes along the path to a *zero strength* region will store routing state about the destination node and, therefore will redirect the message towards the region where the destination node actually resides.



Region	Bloom Filter
A	00010101000010
B	00110000000001
C	01000100000000
D	00001000100001
E	00000000001100

Figure 3.4 Routing to *zero strength* regions.

If one or more entries match the destination identifier, the node will select the strongest entry (temporally and spatially) to route the message to. More specifically, it will select the entry whose associated Bloom filter yields a higher probability that the destination node resides in the respective region. If the matched entries are equally temporally strong, the one with the smaller radius (i.e. the spatially stronger) will be selected. Figure 3.5 illustrates a scenario where the destination identifier (00000000000100) tests positively only against the Bloom filter for region *E*. Consequently, the message is forwarded towards this region.



Region	Bloom Filter
A	00010101000010
B	00110000000001
C	01000100000000
D	00001000100001
E	00000000001100

Figure 3.5 Routing to the temporally stronger region.

Messages carry the temporal strength of the associated routing decision; the spatial strength is also included given that messages carry their destination region (centre and radius) along with the identifier of the actual destination node.

As with routing announcements, GeoHawk employs GPSR [47] to forward messages to a destination region. GPSR routes messages to the centre of the destination region carried in a message. At each subsequent hop, GeoHawk may bias the destination of the message by changing the destination region of received messages. To do so, it compares the strength of the routing state stored at the mobile node against the strength of the information that last biased the destination of the message (or the one set at the source node, as described above). This is done as described in [2] where priority is given to the temporal strength. Ties are broken by looking at the spatial strengths (i.e. the radius of the destination region).

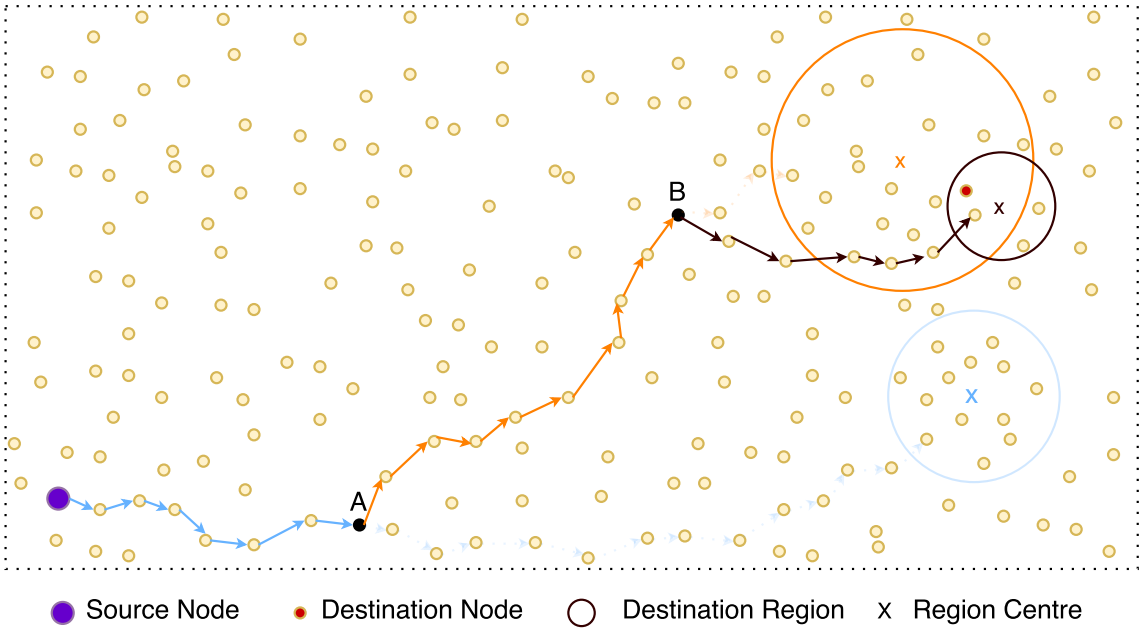


Figure 3.6 Biasing the route of a message.

In Figure 3.6 the message is initially routed towards the destination region shown in blue. The destination node does not actually reside in this region; e.g. because the node has moved and the routing state at the source node is outdated. Note that the message is actually routed to the centre of the blue region (shown with an X) using GPSR. The path that the message takes is illustrated with the arrows of the same colour as the one of its destination region.[5] Node *A* stores routing information that is temporally stronger compared to the one that initially defined the destination of the message. As a result, of that, the message is rerouted towards the orange region (i.e. to its centre), into which the destination node resides. Finally, at node *B* the message is rerouted to the black region which is smaller than the orange one and contains fewer devices. As we explain in Section 3.3.2 this results in less network overhead and energy consumption compared to delivering the message within the orange region. Note that during this phase, it is very unlikely that the message would make it to the destination node or the centre of the region. Given the

[5]The faded arrows represent the path the message would have taken if it was not rerouted by intermediate nodes.

high user density in the network, the expected number of users in destination regions (even in small ones) is high. GeoHawk's second phase ensures that the message will make it to its destination node. This mode of delivery is based on flooding the message within the destination region and is activated when the message reaches its destination region, as described in Section 3.3.2.

3.3.2 Routing a Message within the Destination Region

During the first phase of GeoHawk's routing process, a message is forwarded towards the centre of its destination region (defined as a pair of (centre, radius)) using GPSR. The destination region may change (biased) as the message traverses mobile devices that potentially have better information about the destination device, as explained in the previous section. When a message reaches its destination region, the second phase of GeoHawk's routing process is activated. The objective of this phase is to forward the message to the destination node quickly and with the minimum possible network overhead (and consequently energy consumption). Note that the location of the destination node within the region is unknown, therefore we cannot use a protocol like GPSR to route the message to it. To be more precise, there can be no hard guarantees about the very existence of the node within its destination region [6], although GeoHawk's first phase strives to minimise such a routing failure. In order to assess whether a message's destination node resides within the destination region defined in the message, we calculate the ratio r_m of the number of hops before the message was last biased over the total number of hops that the message took during the first phase; i.e. $r_m = \frac{\#Hops - \#BF}{\#Hops}$. This fraction is 0 if the message was never biased and 1 when it was last biased by the node currently processing

[6] Both aggregation and Bloom filters' false positives contribute to this factor.

it. A (configurable) threshold is used to differentiate between two distinct cases. A second filtering mechanism is used to prevent spurious flooding of messages in regions. More specifically, GeoHawk allows flooding only when the membership test of the destination node yields a probability higher than a threshold; i.e. only when the temporal strength of the message in the destination region is acceptably high. If both of these conditions hold, GeoHawk will flood the message. In the opposite case, GeoHawk starts over as described in 3.3.1.

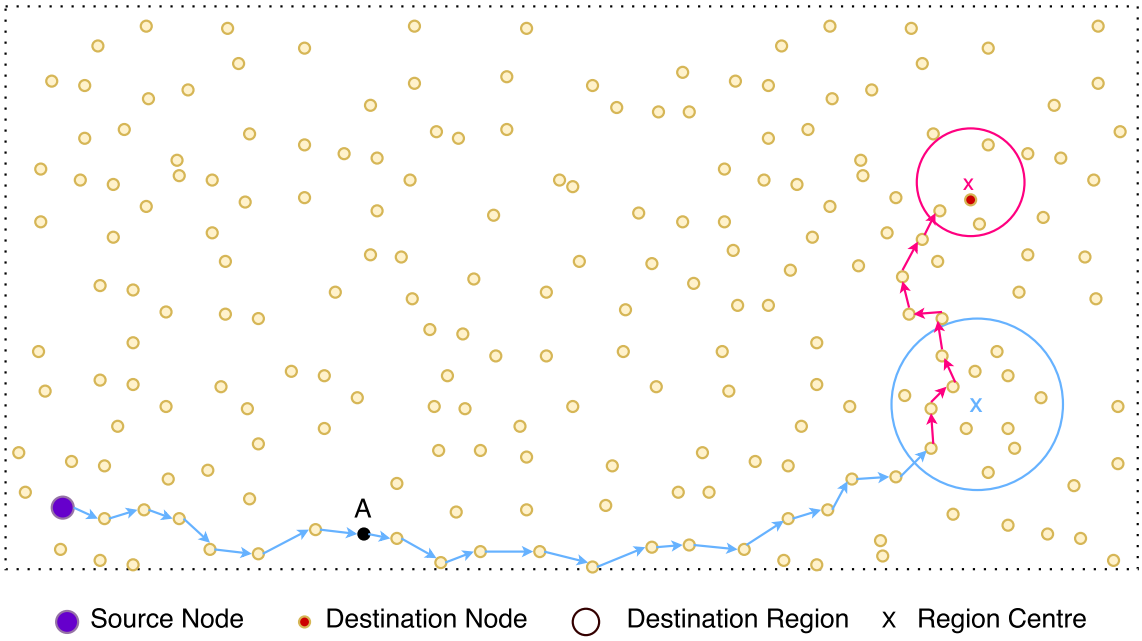


Figure 3.7 Redirecting a message to a different region when r_m is less than the bias threshold.

The latter case is illustrated in Figure 3.7. More specifically, in this scenario node *A* does not have any (strong) information about where the destination node is (nor any other node in the path to the blue region). Consequently, the message's route is never biased and the value of r_m is 0. This is a strong indication for the first node in the message's destination region that receives the message, that the destination node is not in the specified region; flooding the message in this case would be a waste of network resources and would result in unnecessary energy consumption. Instead, the node resets the routing process (to

its first phase) while retaining the last set bias. Resetting the bias value will likely cause the message to linger/be confined within a specific area until it expires. This is because neighbouring nodes will likely have similar information, which can restrict messages from exploring other parts of the network and finding the correct region. In this case the node possesses better information (red region) and, as a result, redirects the message towards its centre (using GPSR). Similarly, GeoHawk would also redirect the message if the temporal strength of the region for the specific destination node was not above the pre-specified threshold.

3.3.3 Ticketing and Replication

GeoHawk employs controlled replication of messages through a simple but effective ticketing mechanism similarly to the one described in [88]. The key objective here is to maximise geographical coverage (and increase message delivery ratio) when conflicting (but similarly strong) information exists at intermediate mobile devices. To avoid inducing excessive network overhead in the network, each message is initially assigned with a pre-specified set of tickets. A message can be replicated only if there are still available tickets. Replicating a message costs a single ticket and replicated messages carry the same amount of tickets the original message carried after the number was reduced by one, i.e. they can also be replicated themselves which enables better exploration across the whole network area. Intermediate mobile devices will replicate a message if they store stronger state which points to a different direction in the network. Direction granularity in GeoHawk is coarse and mapped to the four quadrants of the Cartesian coordinate system. The original message is then routed towards its original path and can be biased towards a different direction only when no more tickets are available.

An example is shown in Figure 3.8. The message is initially assigned with 2 tickets. Node *B* stores stronger information but because there are available tickets, the message is not biased. Instead it is replicated towards the region shown in green. Both messages now carry 1 ticket. Node *F* further replicates the message for the same reason towards the region shown in orange. At that point both messages have 0 tickets and cannot be replicated any further. if any stronger information is encountered ahead, the messages will be biased (but not replicated). The same process is followed at node *C* that replicates the message towards the region shown in red. The result of ticketing and replication is that messages may explore a large portion of a network when stronger information can be found as approaching the destination region(s). In the best case scenario the message will only be flooded in the red region, if the bias freshness and temporal strength thresholds are not met in the blue, green and orange regions.

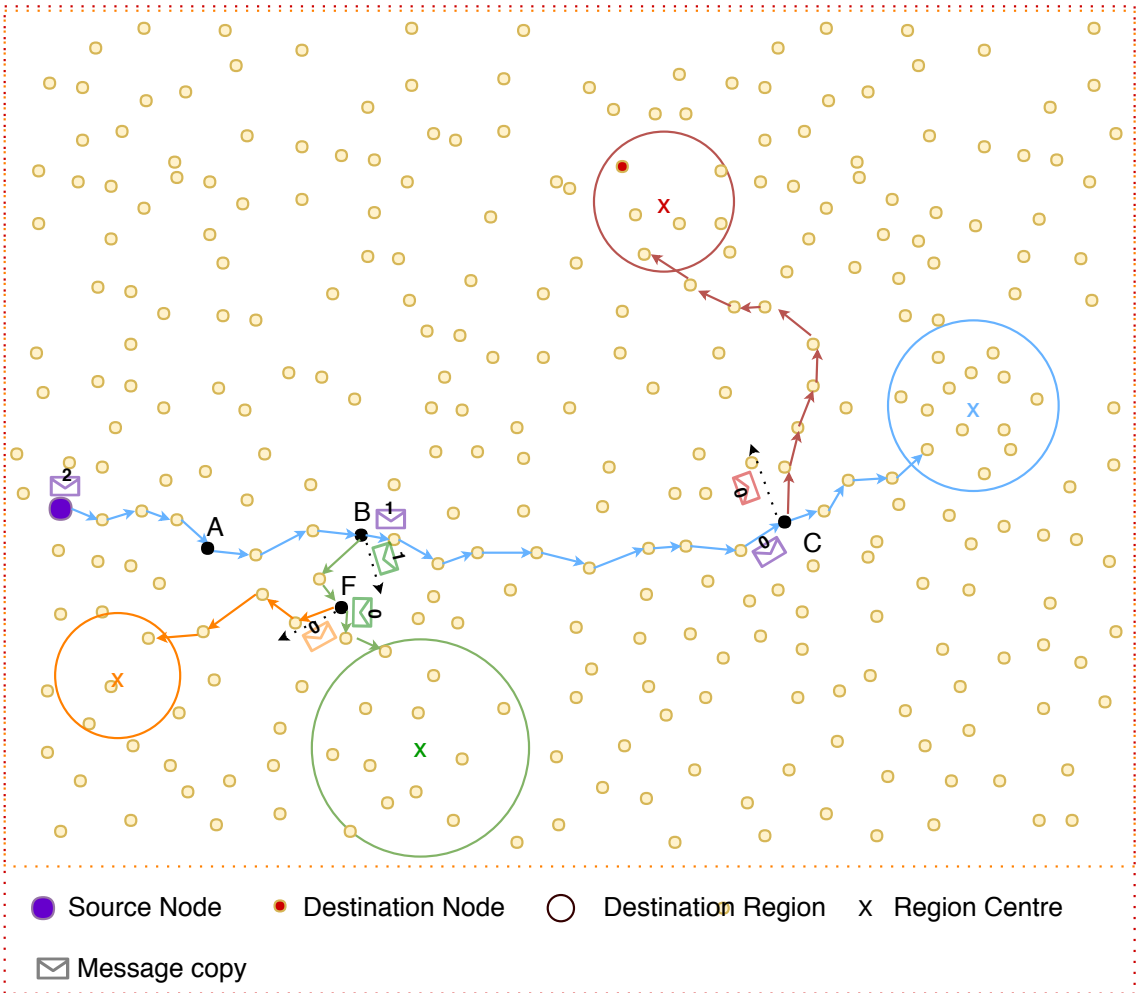


Figure 3.8 GeoHawk replication.

GeoHawk takes a measure to prevent devices from unnecessarily flooding a region even if the destination node is believed to be residing there. More specifically, a mobile device will not flood the same message (including copies of this message) more than once [7]. This is shown in Figure 3.9. Node *B* replicates the message, initially towards the red region. The replicated message is subsequently biased towards the dashed red region. The original message ends up in the dashed blue region which overlaps with the one illustrated in dashed red. Nodes that reside in both these regions will only flood the message once.

[7]In order to eliminate redundancy, each node maintains a set of message IDs that have been received through flooded for a certain period of time. Thus, nodes are able to distinguish redundant messages (i.e. it can aid nodes in deciding whether to accept an incoming message or not).

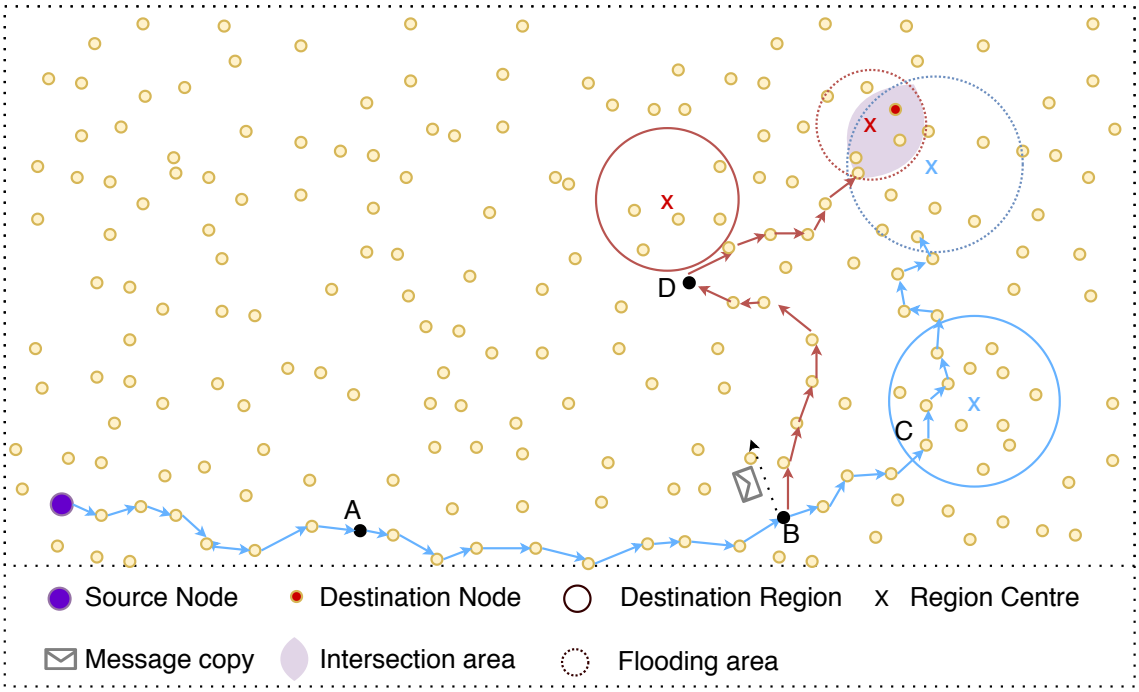


Figure 3.9 GeoHawk replication - preventing duplicate flooding.

3.4 Collaborative Localisation in Dense Opportunistic Networks

In this section we present the design of the proposed localisation approach, which estimates users' position in the absence of GPS or any other accurate positioning system. The proposed approach could operate as a standalone location service; here it is presented as a component of GeoHawk which is designed to be able to operate even when accurate location information is not (always) available.

As discussed in Chapter 2, most geographical routing protocols rely on accurate location information to route messages to their recipients effectively and efficiently. This, in turn, requires significant energy consumption (e.g. to have GPS on), advanced equipment or a pre-deployed network infrastructure [62][117][111][52][96]. This may not always be

possible, especially in dense, opportunistic networks that we study in the context of this thesis. Considering all these, we propose integrating *Pedestrian Dead Reckoning (PDR)* with a *collaborative* approach [51] to estimate the position of a device in the absence of GPS or an accurate positioning system. We employ *Particle Filtering (PF)* to smooth the accumulated error, until a new accurate location reading is possible; i.e. we assume that from time to time mobile devices are able to accurately locate themselves (e.g. by using GPS infrequently, or when they can actually get a GPS signal, or by using pre-deployed location beacons).

3.4.1 Pedestrian Dead Reckoning (PDR)

PDR is a special case of Dead Reckoning, and is used to estimate a device's position by extrapolating it from the previous along with information obtained from inertial sensors (accelerometers and/or gyroscopes), while taking into account the movement complexity of the carrier and the placement of the mobile device [106]. PDR initially provides accurate location estimates but as time goes by (and the distance from the first measurement increases), its accumulated error becomes large and the estimates unusable. In our system, we use PDR [106] as the basis for location estimation and employ collaborative localisation and particle filtering to smooth the accumulated error, which is eventually corrected when an accurate reading is made possible.

3.4.2 Proximity and Collaboration

In order to bound the (accumulated) error induced by PDR, we employ collaborative localisation [51]. In our approach, when a device encounters another device, it is assumed

that they both reside in the same location therefore the objective is to aggregate (potentially) inaccurate location information to produce a better estimate. The transmission range of a specific wireless communication technology determines the accuracy of our localisation approach; the larger the transmission range is, the less accurate an estimation will be, given that the key assumption is that devices are co-located upon encounter. The best-case achieved accuracy is therefore relatively high, given the small ranges supported by common wireless technologies in mobile devices and the fact that we target crowded events, a fact that limits the reachability of devices.

Aggregating location information is based on the approach proposed in [51]. Under the assumption that the location of a device is represented as a joint bivariate normal distribution, with the mean of the distribution (μ) being the location estimate, and the variance (σ^2) being the confidence for the estimate, the updated location estimation for both devices that are encountering each other is as follows.

$$\mu = \frac{\mu_1 {\sigma^2}_2 + \mu_2 {\sigma^2}_1}{{\sigma^2}_1 + {\sigma^2}_2} \tag{3.1}$$

$$\sigma^2 = \frac{{\sigma^2}_1 {\sigma^2}_2}{{\sigma^2}_1 + {\sigma^2}_2} \tag{3.2}$$

where μ refers to the newly computed estimation, and σ^2 denotes the respective confidence of the estimate. Correspondingly μ_1 and ${\sigma^2}_1$ represent the location and confidence of the current device, while μ_2 and ${\sigma^2}_2$ represent the location and confidence of the encountered node's estimated measurement. The main idea is that the newly calculated estimation is closer to the more confident initial estimate. Note that with the given formulas, multiple

frequent encounters between the same two devices, may (significantly) increase [8]the variance of the location estimation[9]; i.e. decrease the confidence about it. To avoid diluting the location estimation upon such multiple encounters, we place a constraint in terms of travelled distance before allowing to subsequently integrate information with a previously encountered device. This is in-line with the studied communication environment; network is dense and mobility is low; mobile devices may be in the proximity of each other for a long time (e.g. when attending a gig in a music festival). If they move away from each other then it is expected that they will roam the area before ever encountering each other again (hence the imposed condition on travelled distance)[10]. Contrary to the approach followed in [51], we use particle filtering to keep track of the location of a device; the average of the bi-variate Gaussian distribution is calculated as the weighted average of the location of all particles. We describe how particle filtering is integrated with the collaborative localisation approach below.

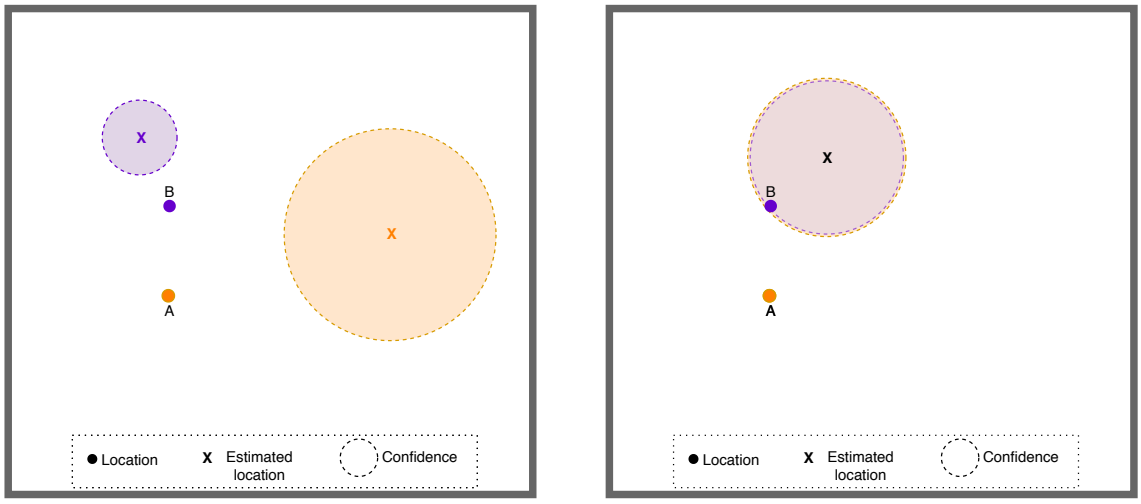
Examples of collaborative localisation are shown in Figures 3.10 and 3.11. In Figure 3.10 node *A* is not confident about its location hence the large radius in the orange circle, which denotes the lack of confidence. On the other hand, node *B* is more confident. The resulting location and confidence are always the same for both devices. As shown in 3.10b the orange radius is now smaller as a result of merging a high-confidence estimate. The purple circle is not larger compared to the pre-merging one. Figure 3.11 illustrates a scenario where both nodes have the same confidence about their location. As a result,

[8]Depending on its original value, multiple encounters can artificially inflate or deflate the value.

[9]Combining the location information of two nodes again and again (e.g. because they are in the vicinity of each other for a long time) could produce false estimation as it can result in a considerably incorrect variance, which eventually affects the estimated positions of other nodes [51]. .

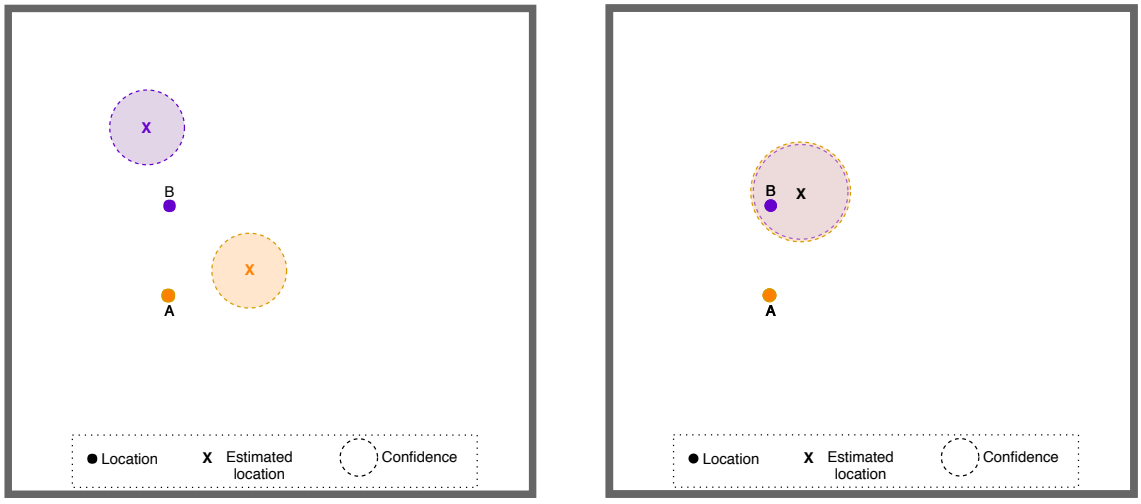
[10]The nature of mobility in application scenarios (namely, very crowded, geographically confined areas, low user mobility and intermittent access to the Internet and location services) confirms this premise.

(shown in Figure 3.11b) the calculated radii are the same, albeit larger than the original nodes, due to confidence degradation due to location integration.



(a) Before merging estimations. (b) After merging estimations.

Figure 3.10 Collaborative localisation when devices have different confidences about their location.



(a) Before merging estimations. (b) After merging estimations.

Figure 3.11 Collaborative localisation when devices have similar confidence about their location.

3.4.3 Particle Filtering

PDR and collaborative localisation are prone to a range of errors, such as intermittent connectivity, signal fluctuation, sensing device failure, sensor sensitivity [52][17], and

in particular PDR noise (error accumulation), which can be challenging to account for even with prediction and detection [51]. Thus, mechanisms such as filtering techniques are needed to eliminate, reduce or smooth noise in order to ensure an acceptable level of accuracy without affecting the obtained measurement excessively [52][17].

A suitable filtering technique that fits our problem space is Particle Filtering (PF). PF uses a set of particles which represent the posterior distribution of some stochastic process given noisy and/or partial observations (the location of a device in the presence of mobility and sensor noise in our case). PF supports both linear and non-linear systems that suffer from high uncertainty [52][17][29]. In PF, the state of the system (i.e. the location of a device in the two-dimensional space) is estimated through a set of generated particles. The life-cycle of the particles undergoes two phases; the observation phase and movement phase. The objective of the observation phase is to predict the future position of the particles according to a known movement model. At the end of the observation phase particles are moved towards the estimated position. During the movement phase particles are re-weighted based on an observation from one or more sensors [29]. Particles are then re-sampled, by selecting the ones with the highest weights and discarding the rest (either after each iteration or when they reach a certain threshold) [52][17][29]. In our network environment there is no pre-specified or known movement model. During the observation phase particles are positioned based on PDR measurements [52][46]. Collaborative localisation is consequently used to correct particles' position in the absence of accurate localisation systems.

Initialisation

Particles are initialised under the assumption that the initial position of a device can be known (e.g. through GPS). Particles are sampled from a bi-variate normal distribution with the actual location as a mean; each sampled particle is basically a location hypotheses paired with a weight to show its likelihood. The covariance matrix represents the uncertainty of the initial position estimation (or conversely the confidence of the mobile device about its location). In principle, the initial position could be unknown. In that case particles are generated uniformly at the network area. We then assign the same weight to all particles, indicating that each particle is likely to be the node's actual position. The weights represent the likelihood that each hypothesis is true [52][17][29].

Figure 3.12 illustrates the initialisation phase in our localisation system; particles (shown as yellow dots) represent a set of hypothesis (i.e. location estimations), which are normally distributed around the initially known position and generated either at the start or after every re-sample process (see Section 3.4.3. As a result they are all very close to the actual location. If the initial location was provided with less accuracy, particles would have been distributed across a much larger circular area denoting the (lack of) confidence about the device's location. Note that in this example we only use four particles. This is for demonstration purposes as in an actual deployment the number would be higher. There is a well-known trade-off between accuracy of estimations and required computation, related to the number of particles.[11]

[11] In our experimentation we will be using 100 particles; generating more particles leads to higher efficiency at the cost of complexity and computational overhead, which results in significantly longer simulations.

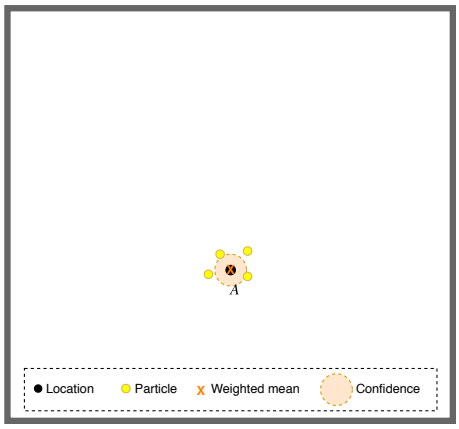


Figure 3.12 Illustration of the initialisation phase.

Prediction phase

When accurate information becomes unavailable, the location of a mobile device can only be estimated through pedestrian dead reckoning. This is done periodically. PDR results in increasing errors which render the confidence about the location lower. Although all particles are moved altogether according to the PDR readings, we model the confidence decrease by separately recording a radius of a circular area. This is subsequently used at collaborative localisation phase to calculate the covariance matrix that is exchanged between two devices upon encounter.

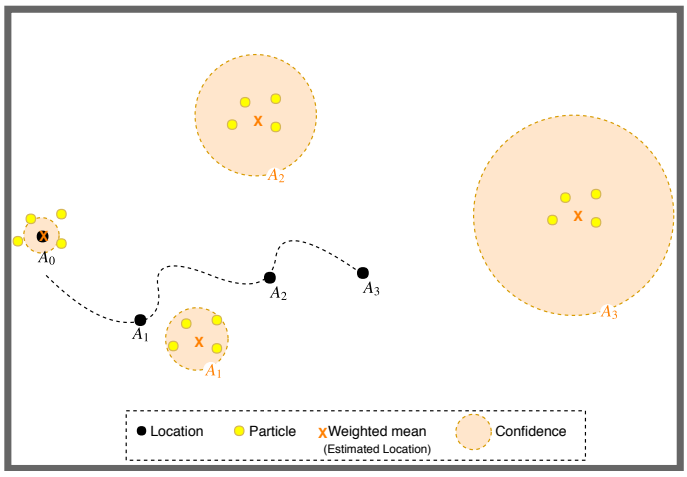
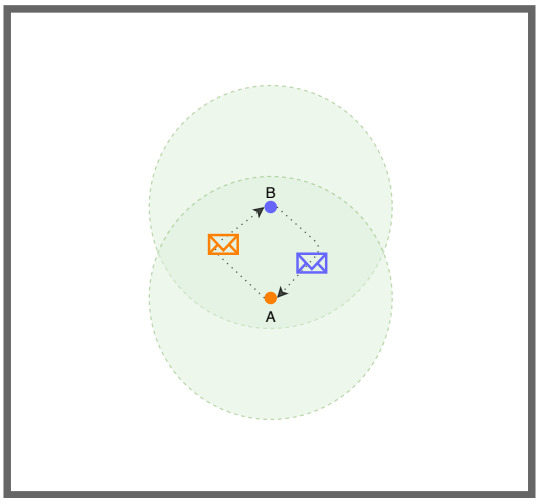


Figure 3.13 Illustration of error growth in the prediction phase.

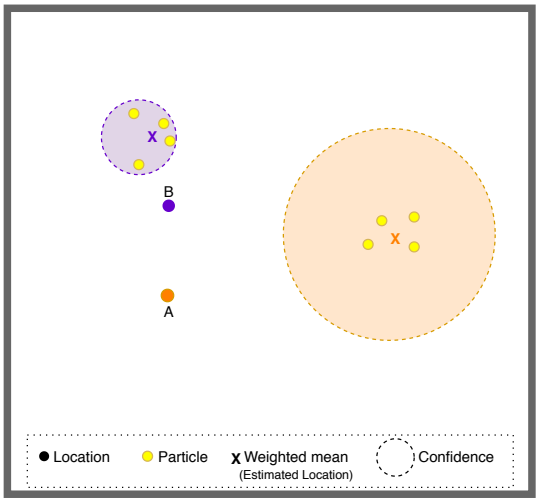
Figure 3.13 demonstrates this process. Node A is moving and PDR is employed at times t_1, t_2, t_3. A's actual location is shown as A_1, A_2, A_3. PDR estimations are shown with an X. The circular area in orange denotes the (ever reducing) confidence about the estimation. Note that although the radius is increasing the particle filters are all moving as PDR dictates.

Measurement phase

Upon encounter, mobile devices exchange their location information and the respective confidence (as shown in Figure3.14a). As mentioned in Section 3.4.2, to avoid unnecessary decrease in the location estimations due to multiple encounters in short timescales (e.g. when devices are roaming in the same area) [51], we only allow such exchange if devices have travelled at least 100m since their last encounter. Mobile devices may have different confidences for their location estimations, as shown in Figure 3.14b.



(a) Message/Information exchange.



(b) Illustration of location confidence of both encountered nodes A and B.

After location information is exchanged, mobile devices compute the mean and variance of their new estimated location (which is common for both devices), according to the formulae discussed in Section 3.4.2. The results of this computation are then used to

re-weight the particles. Particles positioned near the computed collaborative mean are weighted with a higher weight compared to distant ones. Weights are calculated using the Mahalanobis distance of the newly calculated location (co-location of the two encountered devices) and the particles. The Mahalanobis distance is a measure of the distance between a point and a distribution, which is ideal for our setup. It enables us to measure how similar our prediction is (i.e. the particle filters) to the actual value (i.e. the computed collaborative mean).

$$distance = \sqrt{\frac{(x_i - \mu_x)^2}{{\sigma^2}_x} + \frac{(y_i - \mu_y)^2}{{\sigma^2}_y}}$$

Note that the equation above (3.4.3) is a simplified version of the Mahalanobis distance in a two dimensional space with uncorrelated values for x and y; i.e. when their covariance is 0. An example of this calculation is shown in Figure 3.15. Node A encounters node B. B's confidence about its location is way stronger than A's. As shown in the figure, calculated collaborative mean is consequently very close to B's previous estimation. The figure illustrates A's perspective; A's particles are re-weighted so that the ones that are closer to the new collaboratively calculated mean are higher (thickness is shown using the turquoise rings in Figure3.15).

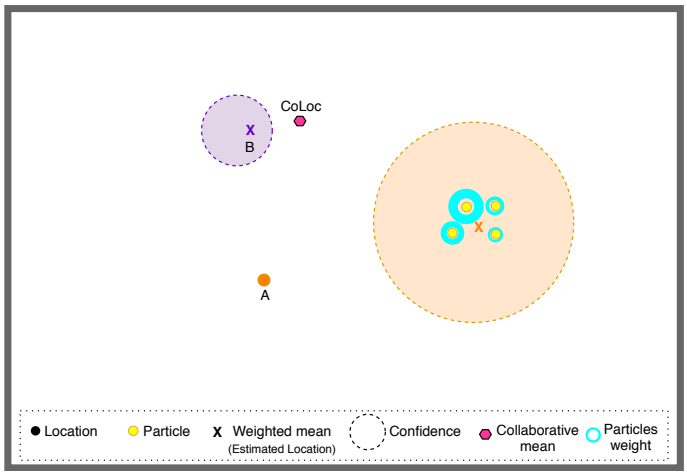


Figure 3.15 Illustration of re-weighting phase.

Subsequently, the computed weights are normalised. We introduce an additional step before the weight value is finalised, which is deciding how much the receiver should rely on the newly computed information versus the previous information. This is crucial to avoid dramatic decreases in the location accuracy when encountering nodes with severely inaccurate location estimations. We employ an Exponentially Weighted Moving Average (EWMA), as shown below:

$$new\_weight = (1-\alpha) * old\_weight + (\alpha) * new\_weight$$

where α is determined as follows:

$$\alpha = \frac{\sigma^2_{Old}}{\sigma^2_{Collaborative} + \sigma^2_{Old}}$$

The value of α ensures that more weighting is put to the new weight when the confidence for the old location estimate was very high (in comparison with the confidence for the new location) and vice versa.

Resampling

The final phase in the life-cycle of particles is the resampling one. New particles are generated around the ones with the highest weights (which are calculated as described above) and the associated weights are reset so that all particles have uniform weights. Particles with small weights are deleted. Resampling is used to avoid situations where all but one of the weights are close to zero.

The whole life-cycle is repeated after every encounter. If accurate information becomes available (e.g. outdoors or when GPS is enabled), the whole process is reset and initialisation takes place again based on the new accurate information.

Chapter 4

Experimental Evaluation and Analysis

In this chapter we present and discuss GeoHawk's extensive experimental evaluation, including performance comparison with the state-of-the-art protocols, namely Epidemic [112], PRoPHET [60] and WSR [2]. We have implemented GeoHawk and WSR in the ONE simulator, which we used to simulate different types of dense opportunistic networks and user mobility. The ONE simulator comes with models for the Epidemic and WSR protocols. In Section 4.1, we discuss the Key Performance Indicators for GeoHawk and how we measure them within the ONE simulator. In Section 4.2, we describe different parameters that influence the behaviour of GeoHawk; we explore the effect of said parameters in subsequent sections.

First, in Section 4.3, we explore GeoHawk's behaviour when varying the parameters discussed in Section 4.2. We experiment with a dense network with static users (Section 4.3.1) and a dense network with slow mobility (Section 4.3.2). Note that the expected mobility in the studied dense network environments is generally limited so the static network approximates the target networks fairly well. By eliminating the mobility factor, we can better explore and understand the behaviour of GeoHawk when varying a diverse set

of protocol parameters. For the mobility scenario, we have implemented specific mobility models in the ONE simulation that allow us to simulate crowded events, such as music festivals. In Section 4.5 we compare the performance of GeoHawk with well-studied opportunistic routing protocols, such as Epidemic, PRoPHET and WSR. In Section 4.5.3, we introduce uncertainty in the location of mobile devices emulating networking scenarios where GPS may be unavailable or prohibitively expensive in terms of energy consumption. Finally, we evaluate GeoHawk in conjunction with the proposed collaborative localisation approach.

4.1 Key Performance Indicators (KPIs)

The core KPIs for GeoHawk are the message delivery ratio, delivery latency and network overhead.

Delivery Ratio. The message delivery ratio captures the ability of a network protocol to deliver messages to their destination effectively and it can be affected by a number of complex network conditions (e.g. limited buffer space in the mobile devices, increased mobility, limited bandwidth, large network overhead etc.). The message delivery ratio is calculated as follows:

$$delivery\_ratio = \frac{\#delivered\_messages}{\#created\_messages} \tag{4.1}$$

Note that in the calculation above, we ignore all control messages and routing advertisements and focus only on data messages that we inject in the network. When appropriate, we report the number of disseminated advertisements separately.

Latency and Hop Count. These two metrics reflect the efficiency of the protocol in terms of the time and number of hops it takes to reach the destination device within the destination region. In order to get the full picture regarding GeoHawk's delivery behaviour and performance, we measure the latency and hop-count for both reaching the destination region (i.e. at the end of phase 1) and the destination device (i.e. at the end of phase 2). Delivery latency and hop count are linked; however, given the mobility in the network, along with the (resulting) opportunistic connectivity, shorter paths (in terms of hop count) may be slower e.g. because specific devices in the path have been disconnected for some time. Along the same lines, longer paths may be faster to traverse if all devices are connected.

Network Overhead. Delivering messages in crowded network environments is far from straightforward. Control messages and advertisements are exchanged, and data messages may be replicated or re-routed when information in the network is stale or inaccurate. For each message to be delivered to its destination the network needs to transmit data larger in size compared to the size of the message; this is the network overhead required to deliver messages. We assess the performance of the proposed protocol in terms of network overhead by separately counting the number of exchanged data and control (including advertisements) messages in the network.

To better understand the effectiveness of GeoHawk's first phase (i.e. how well GeoHawk forwards a message its destination region), we study the effect of the two thresholds that restrict GeoHawk from flooding a message to its destination region; i.e. when it is believed by a mobile device that the destination node resides within a region it currently resides. We define as a *True Positive (TP)*, a situation when a device floods a message in the destination region (where it also currently resides), and the destination node is in this

area. In the context of the simulation we can trivially retrieve the current location of any device. Respectively, a *False Positive (FP)* occurs when a device floods a message in the destination region (where it also currently resides), but the destination node is not in this area. True and False Negatives (TN and FN) are defined accordingly.

We define two threshold values that are used to eliminate false positives when flooding a message into its destination area; these values are for (1) the ratio r_m of the number of hops before the message was last biased over the total number of hops that the message took during the first phase (*path ratio threshold*) and (2) the probability derived when matching a device identifier to a weak Bloom filter (see Section 3.1) (*membership probability threshold*). We report on GeoHawk's behaviour when only taking into account the first threshold and when taking into account both thresholds. The latter case naturally results in lower true and false positives, compared to the former one.

4.2 Protocol Parameters and Mobility Patterns

A number of parameters affect GeoHawk's behaviour and performance; we have extensively experimented with different values in various network scenarios. Below, we group and summarise them.

4.2.1 Geographical Regions and Routing Tables

Routing entries in GeoHawk consist of a region specification (coordinates of centre and radius) along with a weak Bloom filter that represents the identifiers of the devices believed to reside within the region. Region information decays both spatially and temporally (see

Chapter 3.1). A number of parameters affect the decaying process and the underlying weak Bloom filters.

Bloom filters are characterised by their *size*; i.e. the number of bits used to aggregate information about device identifiers, the number of hash functions k used to insert and match an element in the Bloom filter. Along with the expected number of elements n, these parameters result in a specific *false positive probability*, the probability that an element is matched in the filter although it was never inserted. In our approach, we use decaying Bloom filters with the following related parameters; the *decay interval* defines how often devices check and decay stored regions; by default this value is set to the ONE simulators' update rate/interval as shown in Tables 4.1 and 4.2. At every check, a device will discard entries when the cardinality of the respective Bloom filter is below the *cardinality threshold*. When the regions' lifetime (region TTL) expires, temporal decaying is initiated. While a region is being temporally decayed (i.e. after it reached its maximum lifetime), the *decaying probability* defines the probability that a bit set to *1* in the weak Bloom filter will be flipped to *0*; the decay probability defines the Bloom filters' bit flipping rate (see default values in Tables 4.1 and 4.2). The larger the decaying probability is, the faster a weak Bloom filter is decayed, resulting in faster discarding of the respective routing entry.

Each node maintains routing information about the topology in the form of regions mapped to weak Bloom filters. Said routing information is checked against incoming messages which are biased if stronger information is found. The *size* of the routing table determines the maximum size of the routing table before having to drop entries, in order to store new ones. Larger routing tables require more memory from mobile devices and may end up storing stale information. On the other hand very small tables would be problematic

when no information could be found to route messages to their destinations (especially in large and dense networks). The replacement policy is First-In-First-Out.

4.2.2 Announcements

GeoHawk relies on announcements to disseminate routing information that refresh the distributed state of the network. Advertisements are forwarded towards specific directions. The *announcement interval* defines how often a node generates a new announcement that is propagated in the network using GPSR. This value affects the rate at which information is updated in the network and the induced (control) network overhead. The *announcement lifetime* (announcement TTL) defines the number of network hops an announcement can take before it is discarded. The value is configured taking into account the diameter of the network. Small values may result in announcements never reaching remote nodes, whereas large values of the lifetime may bring about unnecessary network overhead and result in stale information being disseminated.

4.2.3 Data Messages

GeoHawk's primary objective is to route and deliver data messages to their destination nodes. The protocol does so by biasing, redirecting, flooding and replicating these messages in an attempt to increase the delivery ratio. A GeoHawk message starts with a given number of tickets ($tickets \geq 0$). A message can only be biased (and not replicated) when $tickets = 0$. Increasing the value of *tickets* may improve the chances of delivering a data message, however at the cost of extra required bandwidth and buffer space at network devices with limited resources, which, overall, may have a negative effect on delivery ratio.

Deciding to flood a message or not if the current node resides in the message's destination region is determined by two thresholds; The *path ratio* threshold acts on the ratio r_m of the number of hops before the message was last biased over the total number of hops that the message took during the first phase. The *membership probability* threshold sets a limit on the weak Bloom filter matching process at the current node; when the matching results in a probability lower than the threshold, the current node will not flood the message, but redirect it to a new region.

4.2.4 Mobility Patterns

We explore three different yet generalized scenarios involving mobility to investigate a range of applications, as the ones mentioned in Chapter 1; (1) devices slowly roam within the whole area covered by the opportunistic network; (2) devices slowly roam within 2 sub-areas in the network (where events take place) and a subset of these move from event area to event area; (3) as in (2) but with 4 events and, therefore, smaller event areas. The mobility pattern for all the above mentioned scenarios is as follows. Initially, each mobile device is randomly assigned with an action which can be to stand within an event area, walk slowly within an event area or walk a bit faster towards another event area (if more exist). The duration of the action is selected uniformly are random from the range [300,600] seconds. At the end of this time period, a node is randomly assigned with a new action, as defined above.

4.3 Exploring GeoHawk's Behaviour

GeoHawk involves a number of parameters that affect its behaviour in complex and, at times, conflicting ways. Before proceeding with comparing its performance with the state-of-the-art, we systematically explore its performance and associated behaviours when varying these parameters. Such a systematic exploration will allow us to understand how sensitive GeoHawk is to specific parameters and, equally importantly, how to set their values for the various studied (simulated) deployments.

4.3.1 Static Network Topology

In this section we evaluate the performance of GeoHawk in a static network topology. The fact that mobile nodes are not moving, simplifies exploring the behaviour of the protocol. As mentioned above, it is also a fair approximation of our studied opportunistic network environments, where mobility is assumed to be low given that nodes are densely populated in a given area. The default values for all simulation parameters are shown in Table 4.1.

We have simulated an opportunistic network that spans an area of $41 * 41m^2$ [1] and contains 1533 [2] devices that were uniformly distributed in the simulated area . The transmission range and rate were set to 5 metres and 250Kbps, respectively [3]. The duration of each simulation was configured to be 1800. Time in the ONE simulator progresses through periodic ticks of the virtual clock; we set the period to be 0.5 seconds seconds[4].

[1] Running ONE simulations in larger areas and with the same density would not be practical.

[2] This density is selected based on hexagonal tessellation, which divides the simulated area in a way that ensures equal spacing and uniform distribution of the desired density.

[3] Although we use the same transmission rate as the one supported by Bluetooth, the transmission range had to be adjusted in order to simulate signal strengths within dense settings while making it proportional to the selected area size.

[4] Both the simulators' duration and refresh rate (periodic ticks) are also another limitation of the ONE simulator, which was designed for sparse environments that handle large datagrams; a larger refreshing

Table 4.1 Default values for static scenario.

Parameter	Value
Map size	41m * 41m
#hosts	1533 nodes
Transmission range	5m
Transmission rate	250Kbps
Simulation duration	1800s
Update interval	0.5
Cool down period	100s
Message size	100KB
Message TTL	60s
Message arrival rate	10 messages per 100 seconds
Buffer size	10M
Announcement frequency	10s
Announcement TTL	15 hops
Bloom filter size	4000bits
Table size	30 FIFO
Region TTL	30s
Decay radius rate	0 m/s
Decay probability	0
Cardinality threshold	1
Tickets	0
Path ratio threshold	0.9
Membership probability threshold	0.9

The last 100 seconds are ignored (cool down period) [5]. Data messages (100KB each with a TTL value of 60 seconds) are injected in mobile devices that are selected uniformly at random [6]; we inject 10 messages every 100 seconds. We have set the value of the buffer size at each mobile device and the message lifetime to 10MB and 60 seconds, respectively[7].

Mobile devices issue advertisements every 10 seconds and each advertisement can be forwarded for up to 15 hops. Bloom filters are 2000 bits long. Routing tables can store up to 30 routing entries and the replacing policy is FIFO. Regions' spatial decay is disabled (set to 0) to suite the topology of a static scenario. Thus, the regions' size will only be affected by region integration. Temporal decaying will start after 30 seconds but decaying is disabled given that the network is static, which means that routing entries are discarded immediately after temporal decaying is triggered. This is achieved by setting the cardinality ratio threshold to 1; i.e. an entry is discarded when fewer than 100% of the Bloom filter bits are zero (which is obviously always true)[8].

Finally, by default, replication is disabled (initial number of tickets is set to 0)[9] . The two thresholds that control flooding in destination regions, as described in Section 3.3.2, are set to 0.9 which is a strict choice [10] .

Before discussing the results for the static network topology, it is worth exploring what would be an ideal reported delivery ratio. A value of a 100% would not be possible for a

rate means that we lose bandwidth (the chance to send/exchange small data packets), while a smaller rate indicates more processing hence, slowing down the simulator.

[5]The cool down period considers the average latency and frequency of packets being generated.

[6]The size of packets and the frequency of generation are also constrained by the simulator's capabilities.

[7]We chose a large buffer size to assist us in evaluating the protocols performance without having to consider the limitations on resources. On the other hand, the message lifetime here reflects latency average.

[8]The default values of advert frequency and TTL, BF size, routing table size, region TTL and cardinality were selected based on the assessment of series of experiments, which considers the simulators' limitations and protocols' performance in the defined environment.

[9]Our analysis showed that packets can easily find their way in this static scenario. As a result, we decided to disable the ticketing mechanism.

[10]Strict thresholds are deployed to guard against false positives and spontaneous flooding.

number of reasons; (1) in the current static network topology, GPSR constructs planarised graphs with leaf nodes (i.e. nodes with a signal connectivity edge). This means that GPSR will likely forward messages that have been traversing the networks' external/outer face to one of these leaf nodes. In the absence of mobility, leaf nodes will keep messages until they expire, as GPSR will not forward a message to the node it received it from to avoid routing loops. (2) Bloom filters are prone to false positives [11], which means that, even in a static scenario, a message may be flooded in the wrong region; in turn, this means that the message will never make it to its destination node.

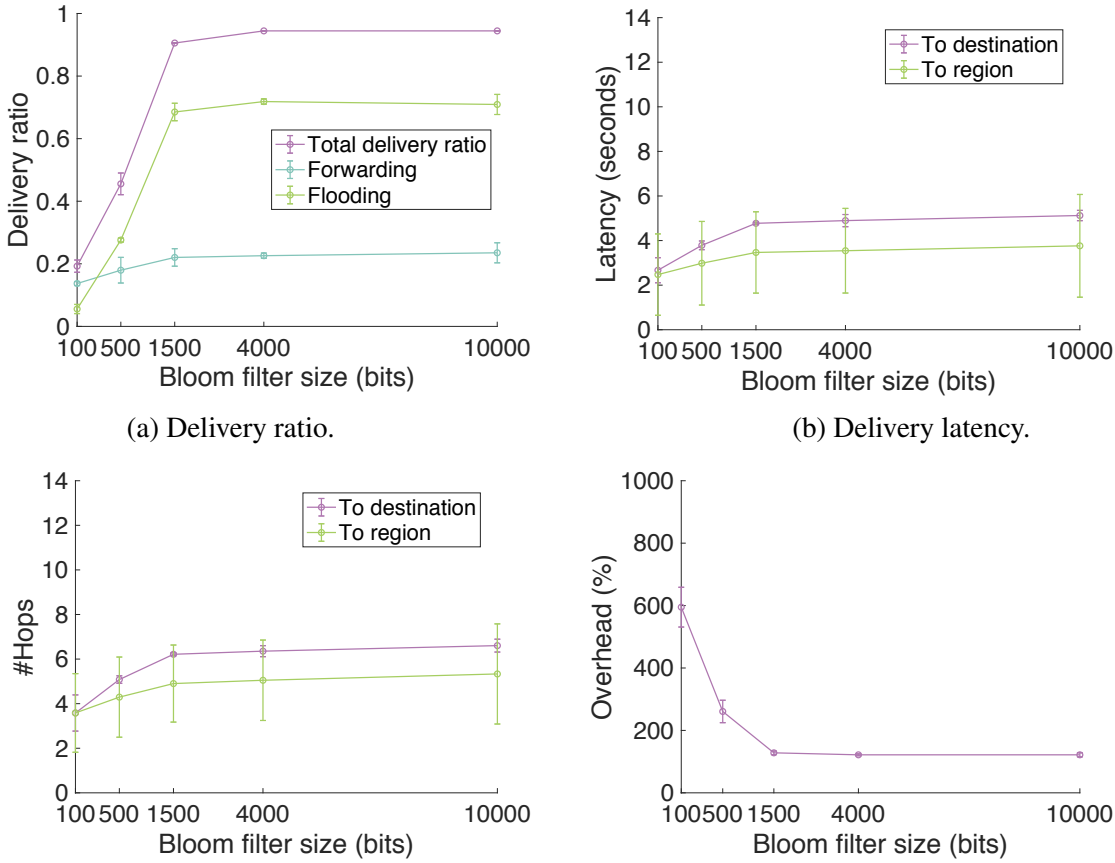
Bloom Filter Size

GeoHawk relies on Bloom filters for aggregating routing information in terms of identifiers of nodes that are known to be residing within a specific region. The size of the Bloom filter has significant impact on the false positive rate when matching the identifier of a destination node to locally stored routing state. Below we investigate the effect of the Bloom filter size on GeoHawk's performance.

In Figure 4.1a we see that the Bloom filter size significantly affects the delivery ratio; more specifically, for sizes smaller than 1000 bits, message delivery is very problematic due to a large amount of false positives. The delivery latency and associated number of hops to the destination device and region are smaller for smaller Bloom filters. This is again because of the large number of false positives which lead to flooding very quickly. This is apparent in Figure 4.1d where the induced network overhead is very high for small Bloom filter sizes (600%).

[11] False positives are one of the main characteristics of Bloom filters. However, the severity of this problem becomes more apparent with temporal decay.

Overall, we observe that GeoHawk performs very well for Bloom filter sizes that are greater than 1500 bits in all KPIs. Larger Bloom filters do not provide any further benefits for this setup. The Bloom filter size is a deployment specific parameter which relates to the number of mobile devices and the density of the network; i.e. given a desired false positive rate (a very small one for GeoHawk) and an expected number of identifiers in the Bloom filter, one could then calculate the optimal Bloom filter size.



(a) Delivery ratio.

(b) Delivery latency.

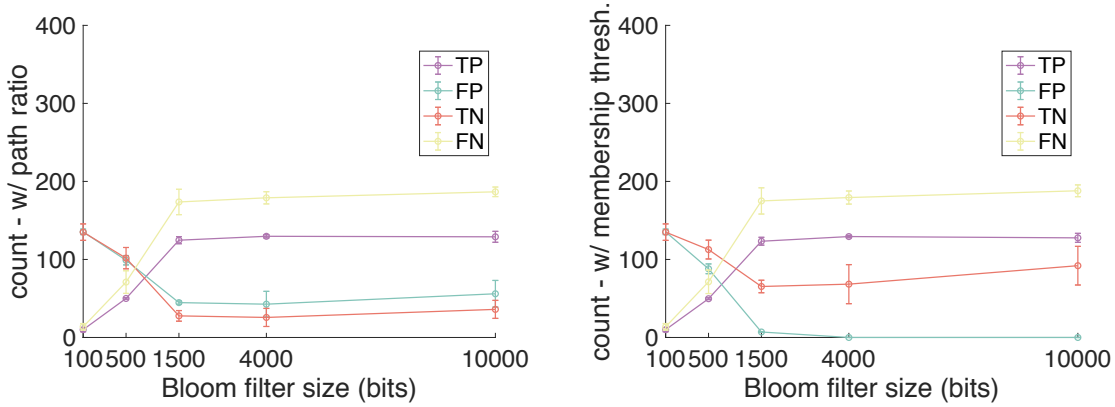
(c) Number of hops to destination region and device.

(d) Network overhead.

Figure 4.1 Performance analysis - varying the Bloom filter size (static scenario).

Figures 4.2a and 4.2b illustrate how GeoHawk thresholds (see Section 3.3.2) prevent spurious flooding (i.e. flooding when the destination device is not in the flooded region). Figure 4.2a illustrates the number of TPs, FPs, TNs and FNs when only the path ratio threshold is used to filter out spurious flooding. It is clear that for small Bloom filters

the number of FPs is very large compared to the number of TPs. When the membership threshold is enabled the number of FPs is eliminated for larger (usable) sizes



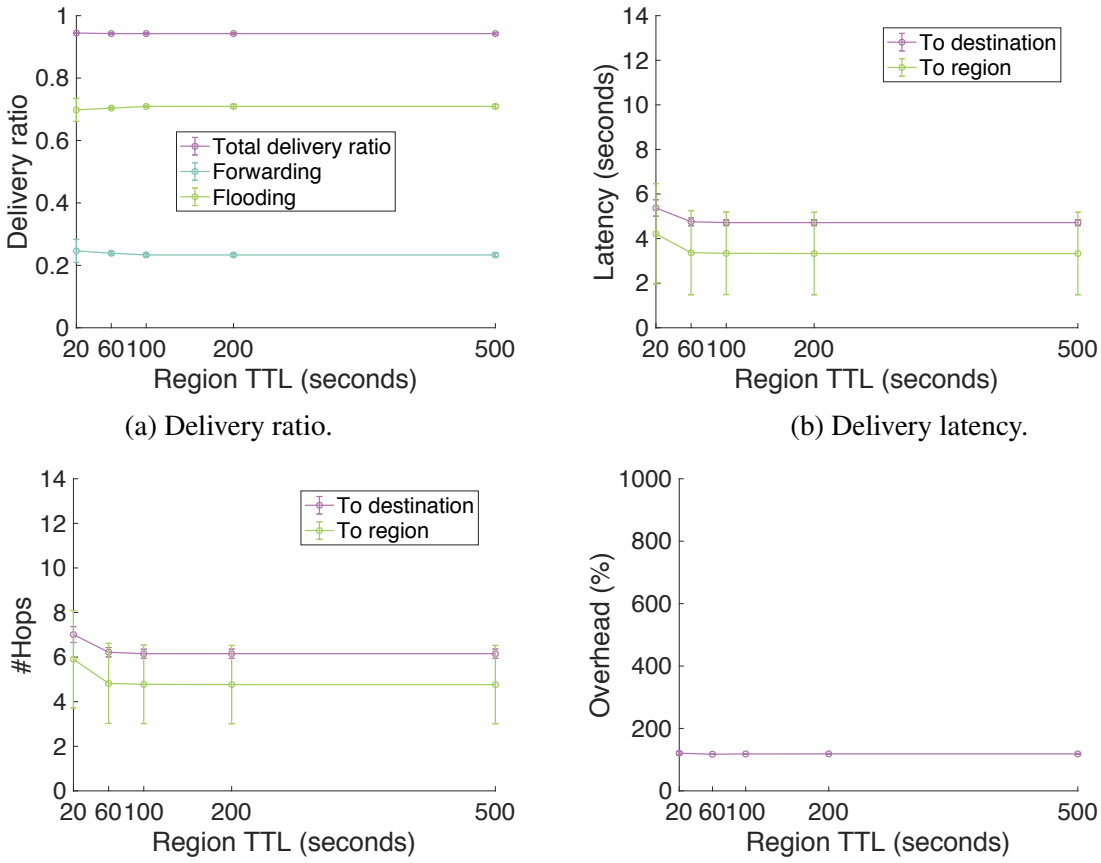
(a) (b)

Figure 4.2 Preventing spurious flooding - varying the Bloom filter size (static scenario).

Region TTL

GeoHawk nodes discard routing entries either to make space for new entries when the table is full or when the stored region is temporally decayed. Temporal decaying (which is disabled in this set of experiments) is triggered when spatial decaying is completed (see Section 3.2.1). Consequently, the region TTL is one of the parameters that affects the lifetime of the stored routing entries in GeoHawk routing tables. Below we experiment with different values for the maximum region lifetime (in seconds).

As shown in Figure 4.3, for the given network setup, only very small values of the region TTL affect GeoHawk's performance; i.e. higher delivery latency (Figures 4.3b and 4.3c) and higher overhead (Figure 4.3d). This is due to occasional lack of routing information which requires more redirections, as discussed below.



(a) Delivery ratio.

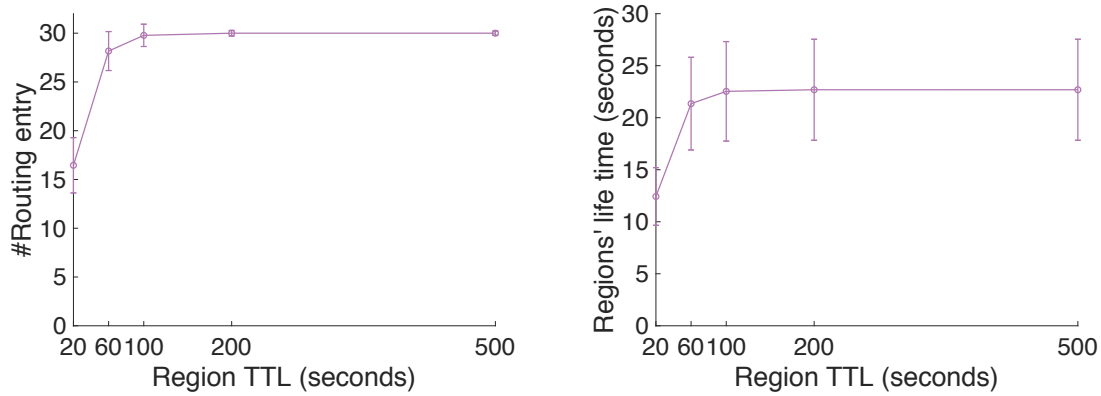
(b) Delivery latency.

(c) Number of hops to destination region and device.

(d) Network overhead.

Figure 4.3 Performance analysis - varying the region TTL (static scenario).

Figure 4.4 depicts the effect of the region TTL in the size of routing tables and the total lifetime of regions in routing tables. We clearly see that small values of the region TTL result in smaller routing tables, which as discussed above affect GeoHawk's performance. The total region lifetime is also related to the TTL, as shown in Figure 4.4b. More specifically, the region TTL puts an upper bound on the lifetime of a routing entry; for the smallest TTL value (20 seconds) we observe that the lifetime of a routing entry is significantly lower than when the TTL value is 100 or larger. For these values routing entries are replaced by new entries extracted from incoming advertisements before the TTL expires.



(a) Average table size.

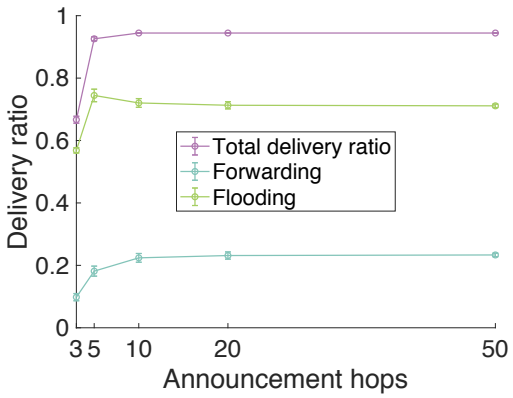
(b) Region lifetime.

Figure 4.4 Table size and region lifetime - varying the region TTL (static scenario).

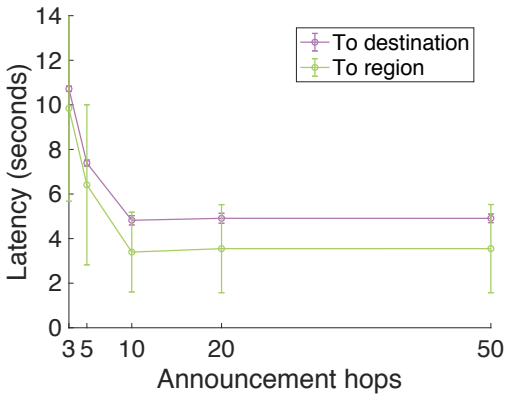
Announcement TTL and Frequency

GeoHawk relies on announcements to propagate aggregated routing information to mobile devices across the crowded network. It is therefore important to understand the relationship between the advertisement TTL (in hops) and generation frequency to GeoHawk's behaviour and the overall performance. Announcements are forwarded towards random directions using GPSR. For a given network diameter (in hops), which can be estimated for a specific network deployment, one could also estimate the average number of hops that GPSR will require to deliver an advertisement across the network.

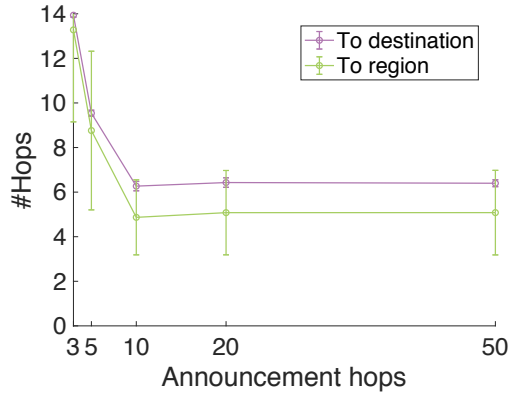
In Figure 4.5 we observe that when the advertisement TTL is small, routing performance is significantly degraded. This is because advertisements are only forwarded in their immediate neighbourhoods, which means that devices do not have routing information about devices that reside further in the network.



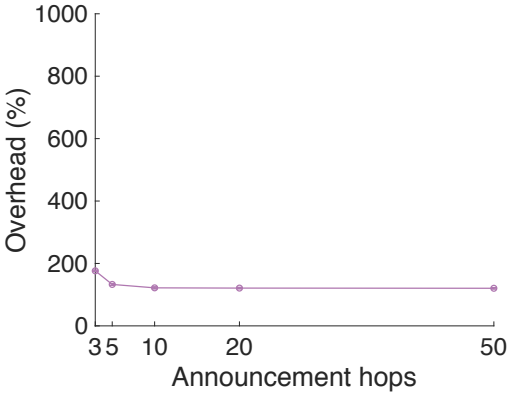
(a) Delivery ratio.



(b) Delivery latency.



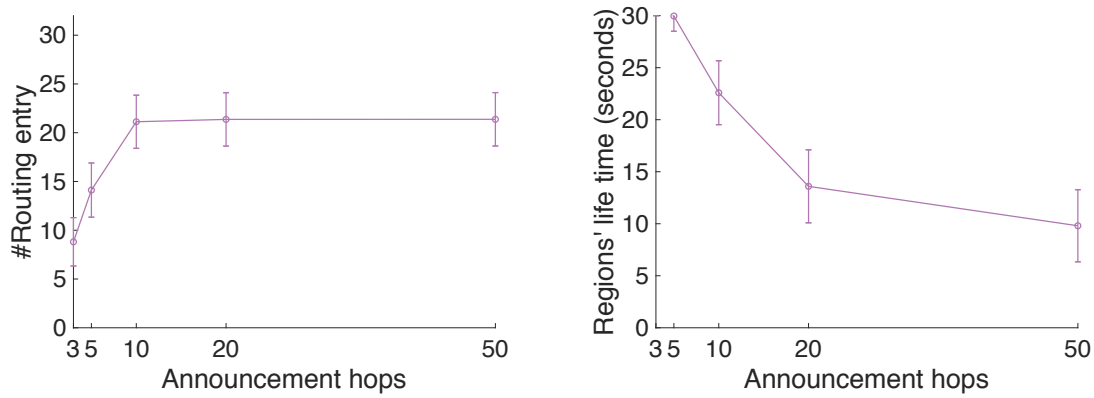
(c) Number of hops to destination region and device.



(d) Network overhead.

Figure 4.5 Performance analysis - varying the announcement TTL (static scenario).

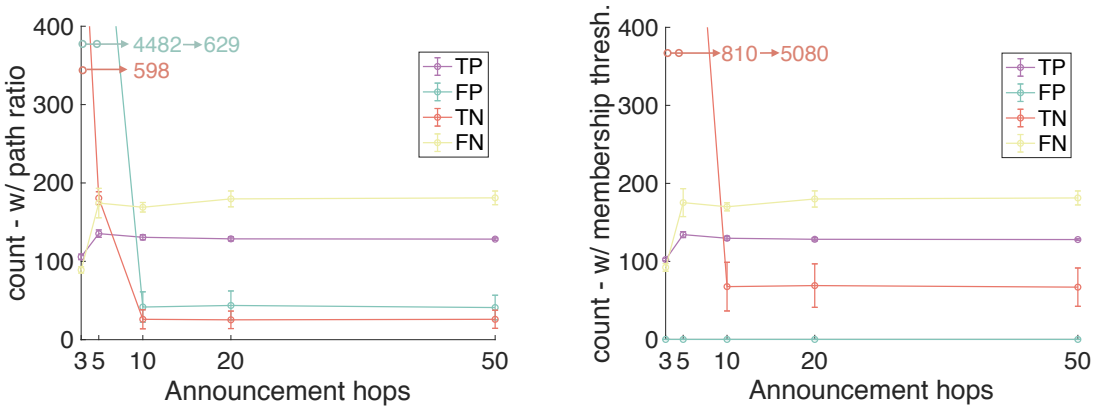
The underlying reason for the performance degradation when the advertisement TTL is small is shown in Figure 4.6. The average routing table size significantly smaller for very small TTL values, which is translated to poor routing decisions and, consequently, performance degradation. The overall lifetime of routing entries is larger as fewer advertisements are seen by nodes due to their limited forwarding range.



(a) Average table size. (b) Region lifetime.

Figure 4.6 Table size and region lifetime - varying the announcement TTL (static scenario).

Figure 4.7 illustrates the effect that poor (or lack of) routing state has in terms of finding and flooding a message in the correct destination region. We observe that the number of TNs is extremely high compared to the TPs when the announcement hop size is very small. This means that most messages are routed towards regions that are not the correct ones. GeoHawk thresholds do a good job preventing flooding the messages there (given the large number of TNs), however performance is still degraded.

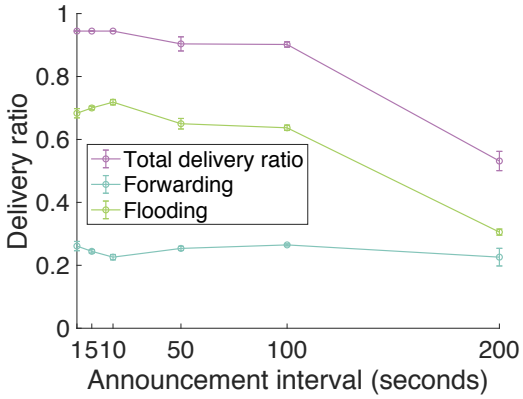


(a) (b)

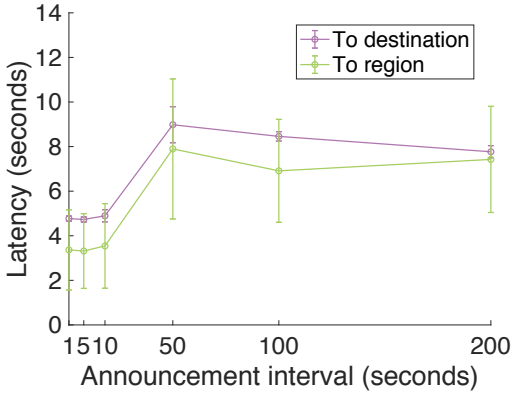
Figure 4.7 Preventing spurious flooding - varying the announcement TTL (static scenario).

Below we investigate the effect of the frequency of advertisement propagation into GeoHawk's performance. In Figure 4.8, it is clear that GeoHawk's performance heavily

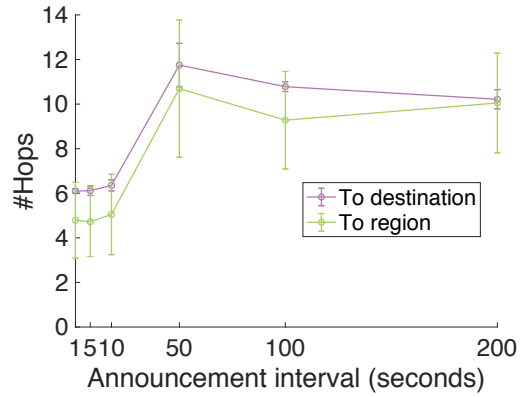
depends on the frequency of sending advertisements. When frequency is very low (e.g. less than 100 seconds), delivery ratio drops significantly, while latency and the hop number increase. Network overhead is not significantly affected as spurious flooding is prevented by GeoHawk's thresholds; this is illustrated in Figure 4.8a, where only the number of messages that are delivered through flooding decreases, whereas the number of messages that reach their destination device without flooding is the same.



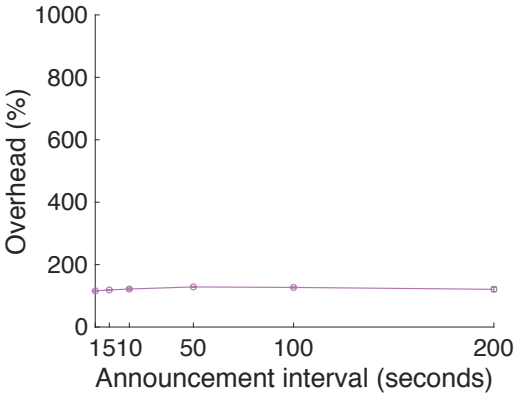
(a) Delivery ratio.



(b) Delivery latency.



(c) Number of hops to destination region and device.

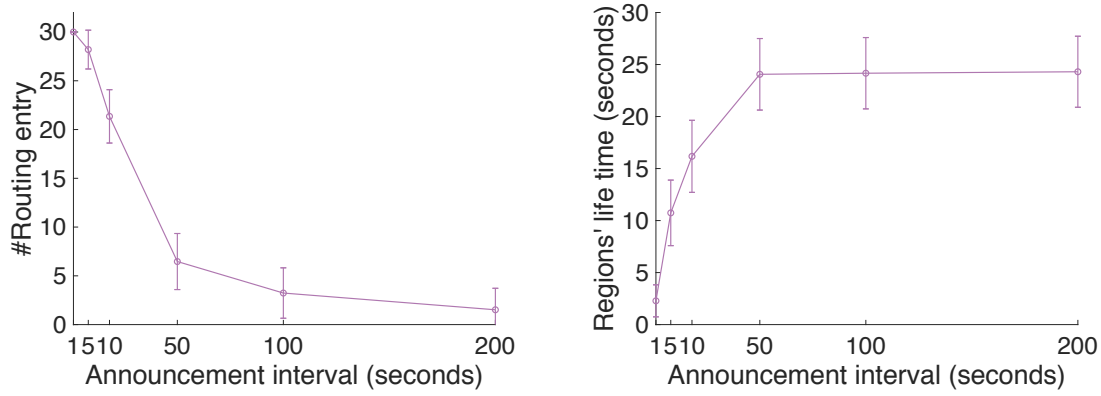


(d) Network overhead.

Figure 4.8 Performance analysis - varying the announcement frequency (static scenario).

The reason for performance degradation is the lack of routing information in devices' routing tables. This is illustrated in Figure 4.9a. It is worth noting that, even with (around) 5 entries (on average), GeoHawk performs acceptably well; this is the effect of routing state aggregation, which is propagated in advertisements. Sparser updates through

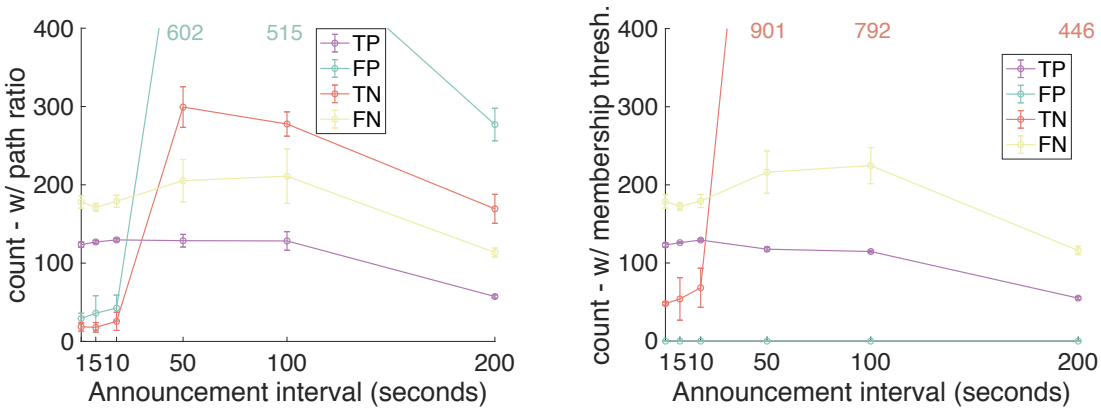
advertisements also result in longer state lifetime; in the majority of cases, routing state is evicted only because the region TTL expires (30 seconds in this simulation setup), given that advertisements are never replaced.



(a) Average table size. (b) Region lifetime.

Figure 4.9 Table size and region lifetime - varying the announcement frequency (static scenario).

As shown in Figure 4.10, infrequent advertisement propagation leads to a large number of FPs (4.10a), which are eliminated when both GeoHawk thresholds are in place (4.10b). In the latter case, there is a very large number of TNs when advertisements are sparse, which explains the serious performance degradation described above.



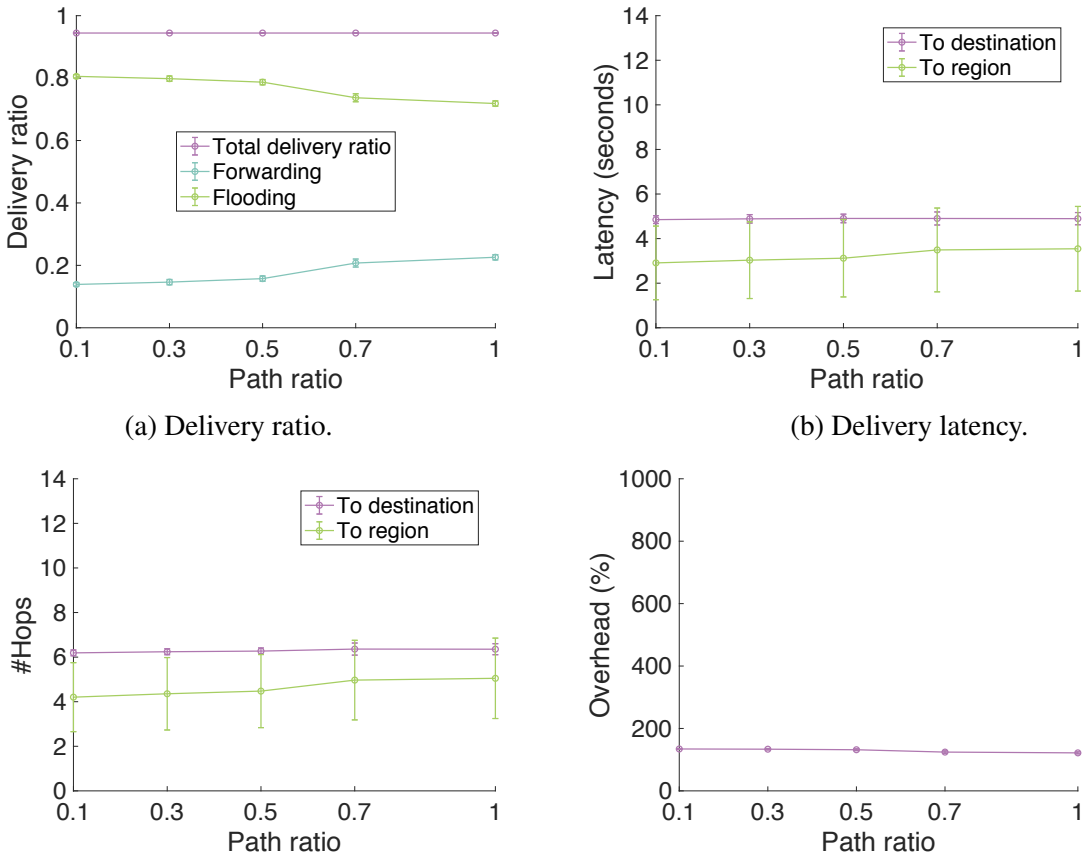
(a) (b)

Figure 4.10 Preventing spurious flooding - varying the announcement frequency (static scenario).

Preventing Spurious Flooding

GeoHawk employs two thresholds to prevent spurious flooding in destination regions [12] . If a message is flooded in a region where the device does not reside then (1) delivery ratio will be decreased as no further forwarding will be attempted for the flooded message (unless it was previously replicated) and (2) network overhead will be increased without increasing delivery ratio. Below we investigate GeoHawk's behaviour for various values of the *path ratio* and *membership* thresholds. In Figure 4.11, we observe that, for this simulated network setup, the path ratio threshold does not have any dramatic effect in GeoHawk's performance. It is worth noticing that as the threshold increases (i.e. GeoHawk gets stricter when it comes to deciding whether to flood or not in the current destination region), flooding is decreased and, instead, destination devices are reached during the first phase (Figure 4.11a). As shown, in Figure 4.11d, this naturally results in a slight decrease ($> 15\%$) of induced network overhead.

[12]Once a message reaches the best known region for a specific destination, the protocol then uses these two thresholds to make a decision on whether to flood this area or redirect towards a stronger one.



(a) Delivery ratio.

(b) Delivery latency.

(c) Number of hops to destination region and device.

(d) Network overhead.

Figure 4.11 Performance analysis - varying the path ratio threshold (static scenario).

GeoHawk's observed performance for different values of the path ration threshold can be explained by looking at Figure 4.12a. As the value of the threshold increases, the number of FPs slightly decreases and, respectively, the number of TNs increases. At the same time, the number of FNs increases, which is a natural consequence of applying a stricter threshold on a specific metric (path ratio). The respective values when the membership threshold is applied on top of the path ratio one (Figure 4.12b) are as expected; the FPs are eliminated and, consequently, the number of TNs significantly decreases, compared to Figure 4.12a.

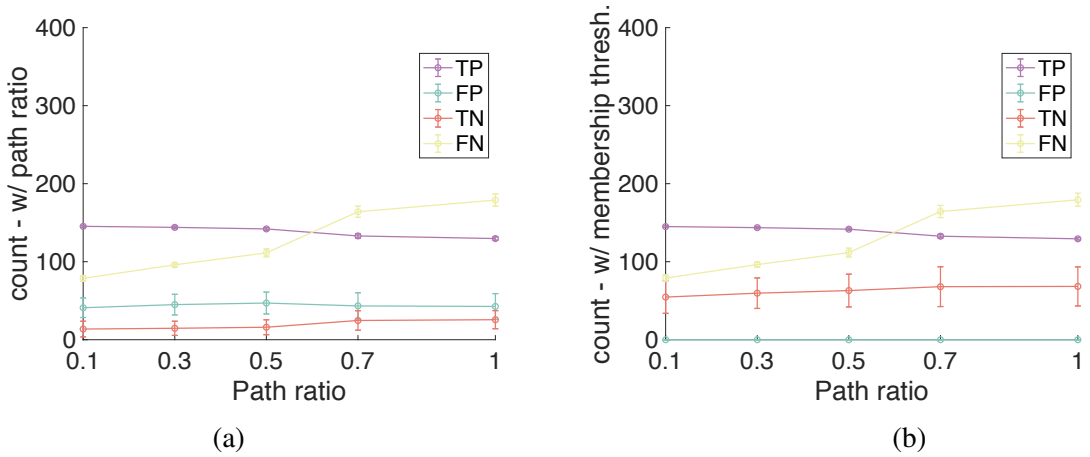
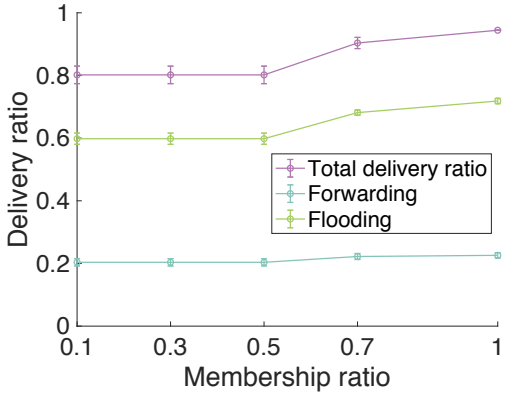
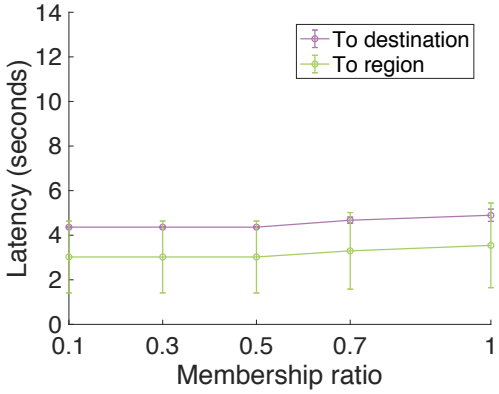


Figure 4.12 Preventing spurious flooding - varying the path ratio threshold (static scenario).

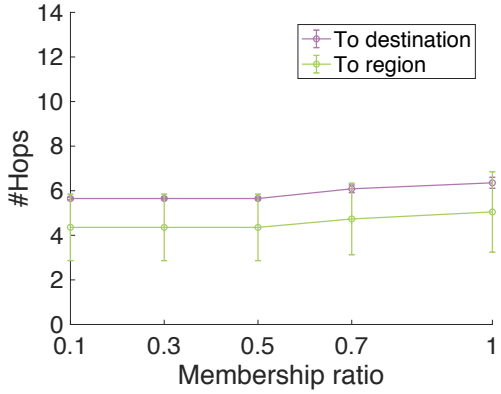
The membership threshold prevents flooding when the BF matching process results in a lower than desired probability. Allowing for values lower than 1 is important when temporal decaying is enabled (where mobility is present). In this network setup nodes are static which means that one should hardcode the value to 1; i.e. matching a BF with a device identifier should yield a value of 1 for the device to reside within the region and any value lower than 1 would be a FP. This is clearly illustrated in Figure 4.13. Delivery ratio is maximised and network overhead is minimised when the membership threshold is 1. The latency (and associated number of hops) appears to be increasing but this is the result of messages reaching more (and more distant) regions that would otherwise be discarded.



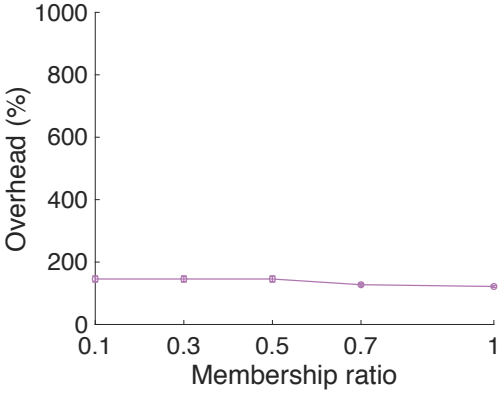
(a) Delivery ratio.



(b) Delivery latency.



(c) Number of hops to destination region and device.

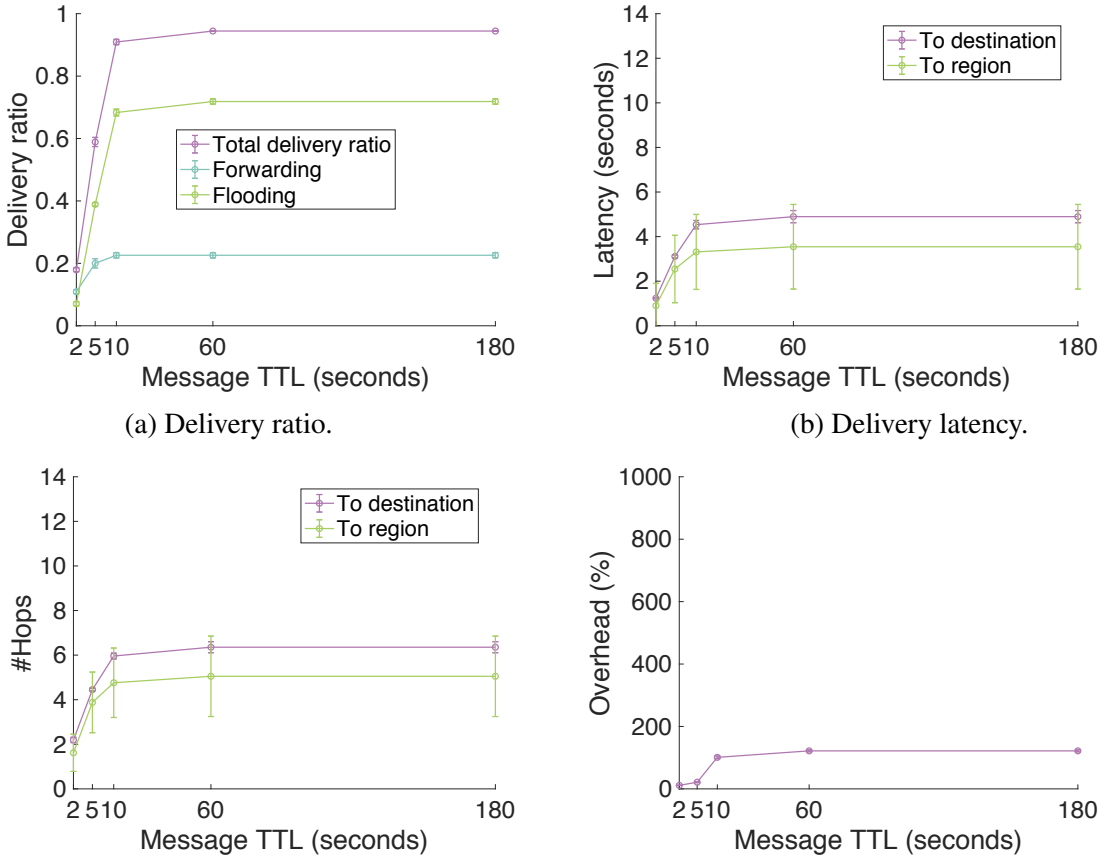


(d) Network overhead.

Figure 4.13 Performance analysis - varying the membership threshold (static scenario).

Message TTL

Each message carries a TTL value which is measured in time units. Messages are discarded when the TTL value hits zero. In Figure 4.14 we observe that GeoHawk's delivery ratio dramatically drops when the message TTL is smaller than 10 seconds. Messages that are destined to regions that are far away from the source node never make it to their destination because they expire and are discarded. Message latency and network overhead appear to be lower for smaller TTL values, but this is only a by-product of the fact that messages are discarded quickly, therefore the ones that make it to their destination do it quickly, minimising the induced overhead.



(a) Delivery ratio.

(b) Delivery latency.

(c) Number of hops to destination region and device

(d) Network overhead.

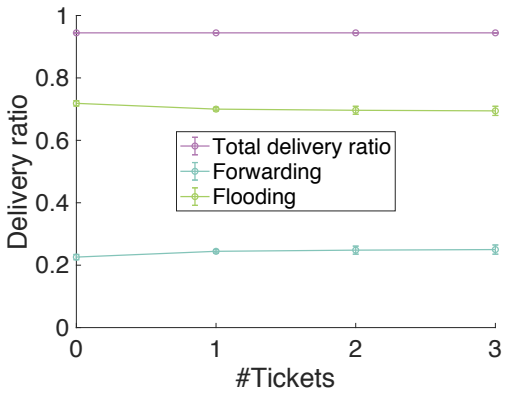
Figure 4.14 Performance analysis - varying the message TTL (static scenario).

Message Replication

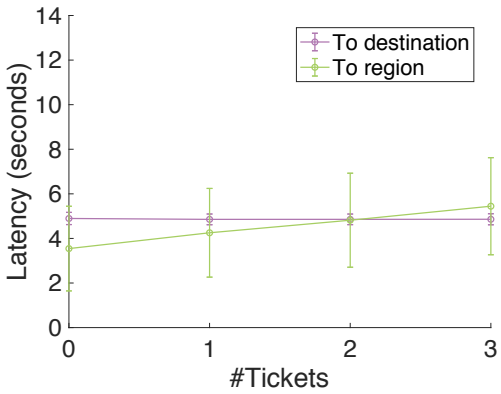
GeoHawk's ticketing mechanism is meant to allow exploring larger areas (with multiple messages) before flooding, when routing information in intermediate nodes is ambiguous but strong [13]. In the static scenario studied in this section (with the default settings described above), mobile nodes have accurate routing information. As a result, GeoHawk's ticketing mechanism does not provide any benefits. This is illustrated in Figure 4.15. Note

[13] Network nodes have no way to confirm the credibility of exchanged information be it from direct encounters or announcements; i.e. the only reliable knowledge is the fact that neighbours are within a communication range. What makes it more challenging is the integration process, which enables nodes to merge received information with existing ones enlarging them specially and modifying them temporally. Since a region can grow large and false positives is one of Bloom filters' limitations, uncertainty and ambiguity are amplified.

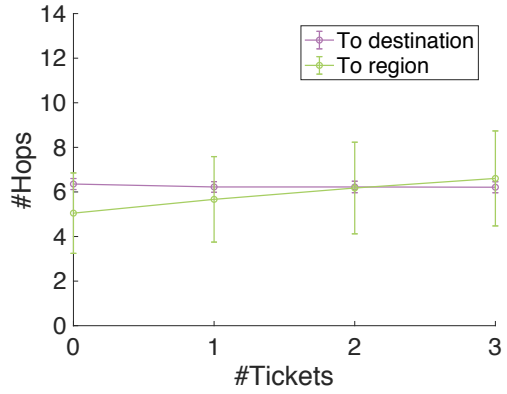
that the network overhead is not significantly increased either, which confirms the above statement; i.e. although ticketing is supported, messages are very rarely replicated because very strong (temporally and spatially) routing information is encountered very quickly.



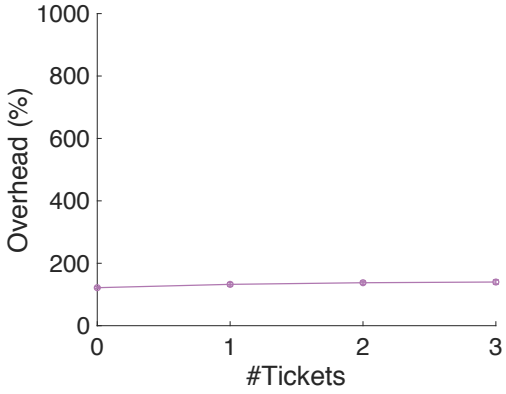
(a) Delivery ratio.



(b) Delivery latency.



(c) Number of hops to destination region and device



(d) Network overhead.

Figure 4.15 Performance analysis - varying the number of tickets (static scenario).

4.3.2 Single-Event Network with Mobility

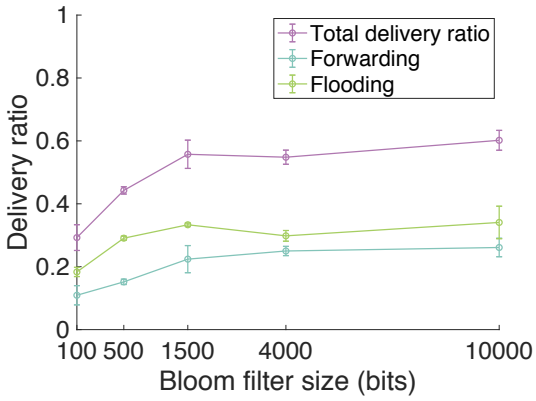
In this section we discuss GeoHawk's performance when devices move in a scenario where there is a single event taking place in the defined network area. Table 4.2 depicts GeoHawk's default values for this scenario. Note that temporal decaying is enabled in contrast to the static scenario where it was redundant.

Table 4.2 Default values for scenarios with movement.

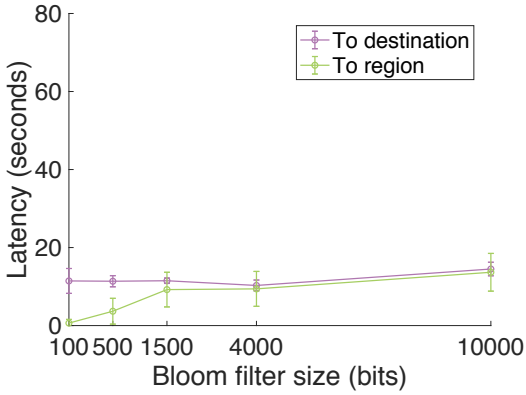
Parameter	Value
Map size	41m * 41m
#hosts	1500 nodes
Transmission range	5m
Transmission rate	250Kbps
Simulation duration	1800s
Update interval	0.5
Cool down period	100s
Message size	100KB
Message TTL	60s
Message arrival rate	10 messages per 100 seconds
Buffer size	10M
Announcement frequency	10s
Announcement TTL	15 hops
Bloom filter size	4000bits
Table size	50 FIFO
Region TTL	30s
Decay radius rate	0.3 m/s
Decay probability	0.001
Cardinality threshold	0.5
Tickets	0
Path ratio threshold	0.9
Membership probability threshold	0.9

Bloom Filter Size

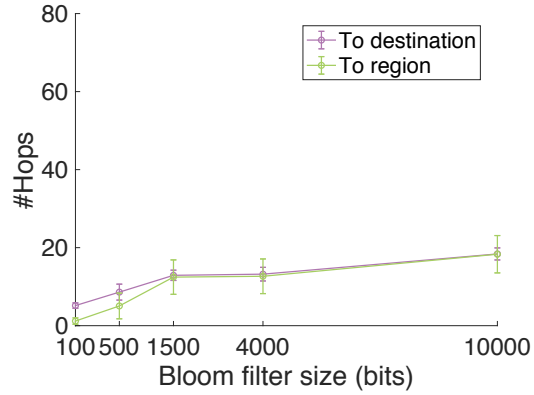
Figure 4.16 illustrates the effect of the Bloom filter size in GeoHawk's performance. As with the static network topology, small Bloom filters result in low performance in terms of delivery ratio and network overhead. Due to device mobility, region information (i.e. a Bloom filter) may be containing more devices compared to the static network scenario (see Section 3.2.1), therefore slightly larger Bloom filters are required to achieve the best-possible performance (given the rest of the default parameters presented in Table 4.2). This is clearly illustrated in Figure 4.16d, where the network overhead for smaller Bloom filters is high (higher compared to the static network scenario).



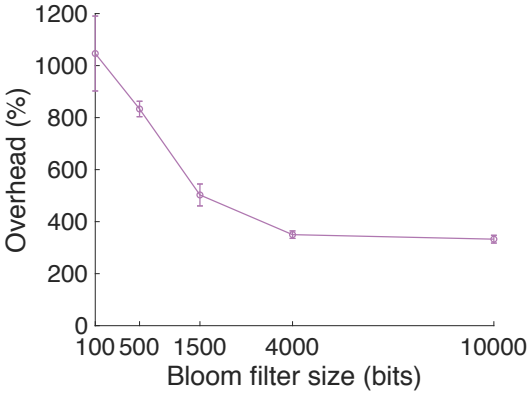
(a) Delivery ratio.



(b) Delivery latency.



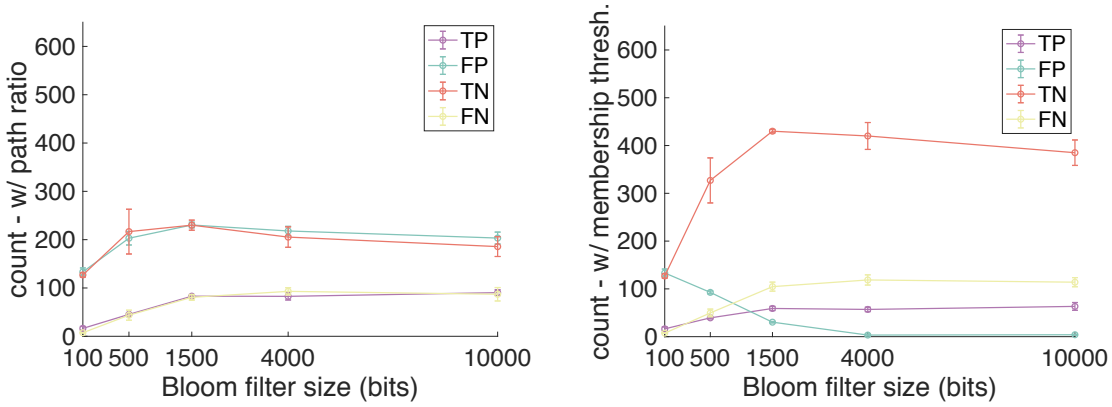
(c) Number of hops to destination region and device.



(d) Network overhead.

Figure 4.16 Performance analysis - varying the Bloom filter size (mobility).

Figure 4.16d explains the very high network overhead mentioned above; even when both thresholds are enabled, a large number of FPs is present for small Bloom filters. The number of TPs is lower compared to the static scenario, which explains the lower delivery ratio shown in Figure 4.16a.

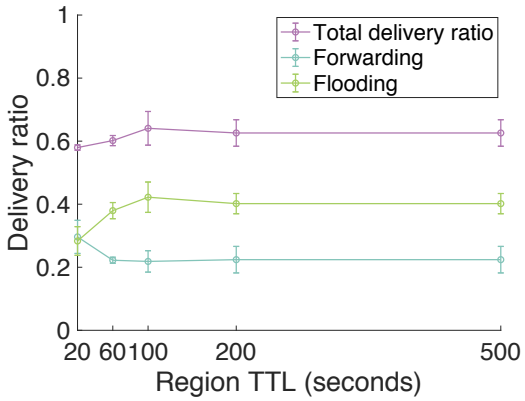


(a) (b)

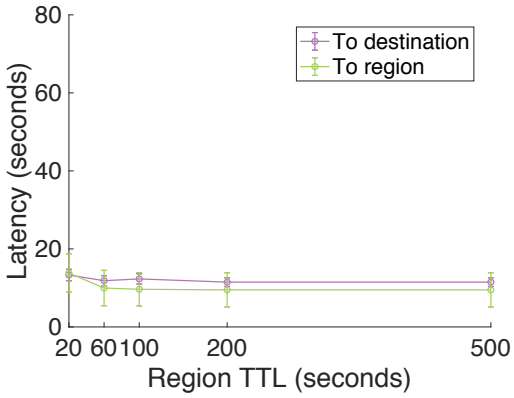
Figure 4.17 Preventing spurious flooding - varying the Bloom filter size (mobility).

Region TTL

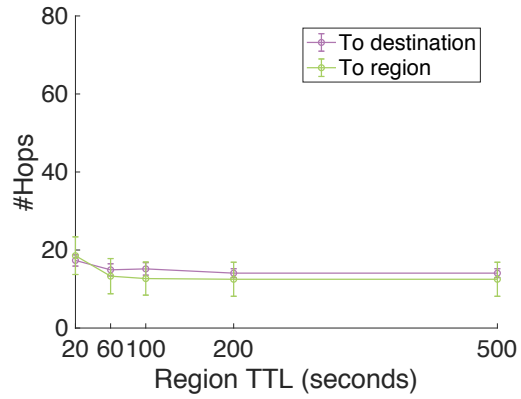
As with the static scenario, very small region TTL values result in performance degradation (Figure 4.18a) due to having less routing information in devices' routing tables (Figure 4.19a). This is because regions are quickly removed from routing tables after being temporally decayed, as shown in Figure 4.19b. The network overhead increases with the region TTL but this is only because more messages are forwarded and eventually flooded, which is necessary for delivering messages (hence the increase in the delivery ratio).



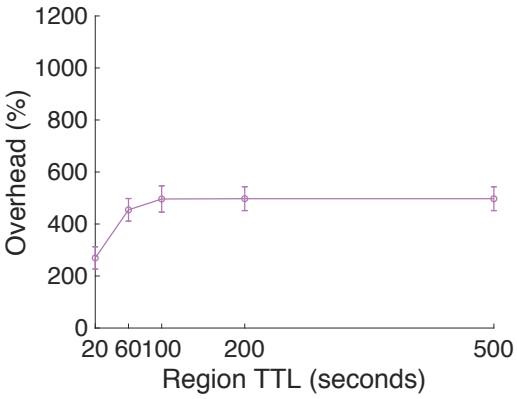
(a) Delivery ratio.



(b) Delivery latency.

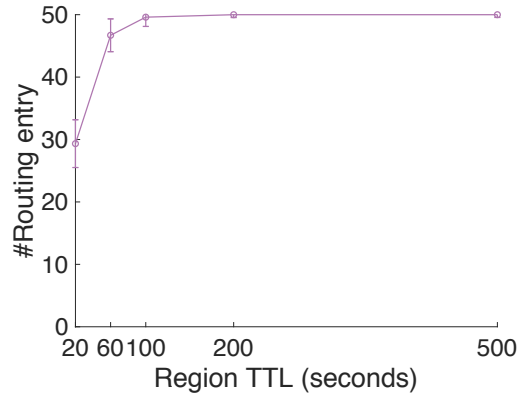


(c) Number of hops to destination region and device.

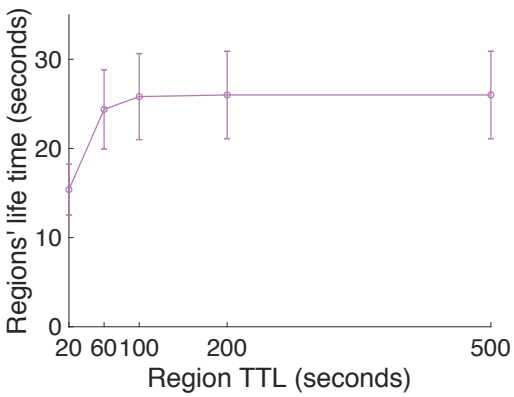


(d) Network overhead.

Figure 4.18 Performance analysis - varying the region TTL (mobility).



(a) Average table size.



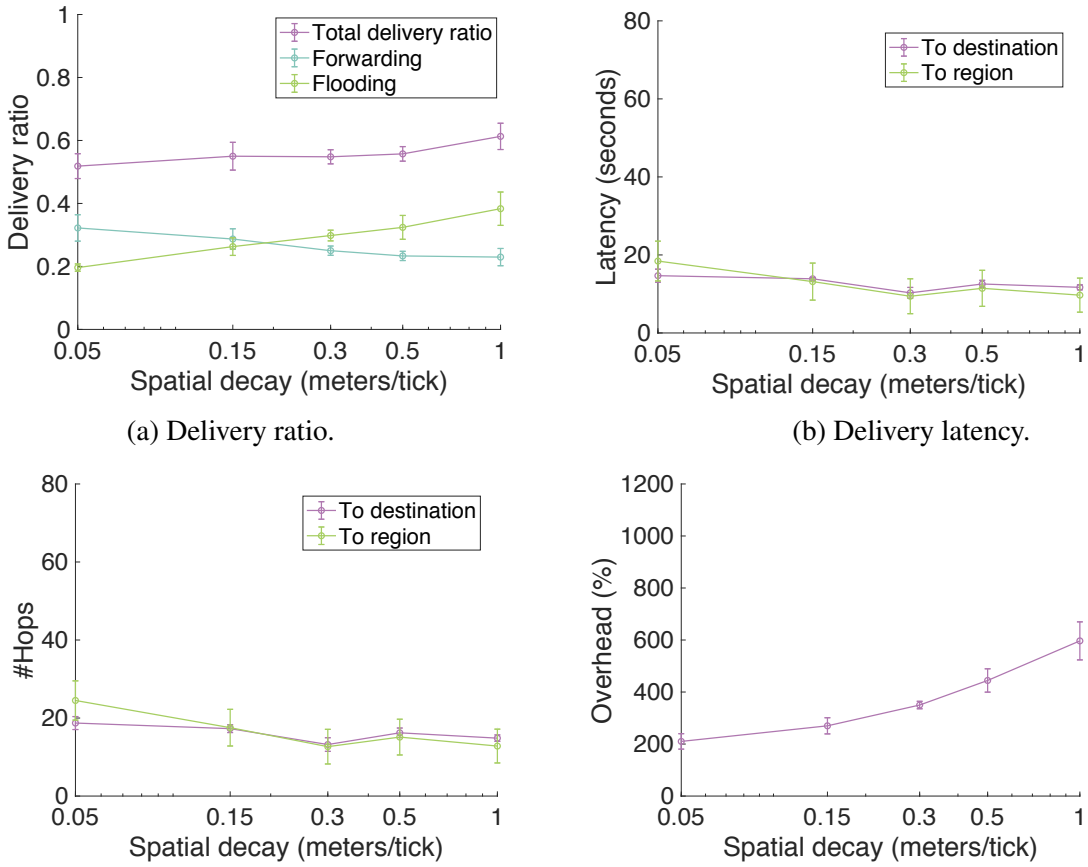
(b) Region lifetime.

Figure 4.19 Table size and region lifetime - varying the region TTL (mobility).

Spatial Decay

In this section we experiment with different values for spatially decaying a region by increasing its radius. By spatially weakening a region, we accommodate for node mobility by assuming that, as the region increases, a moving node will still be in it when a message arrives in the spatially decayed region. Without increasing the radius, flooding would either not happen or happen within a smaller region and the destination node would have moved outside this region. The values on the x axis are in meters per second [14]. As shown in Figure 4.20a, the delivery ratio increases as the spatial decay rate increases. This comes at a significant increase in the induced network overhead; spatially decayed regions are larger, therefore flooding is more extensive.

[14]Node mobility as well as nodes' velocity play a key role in determining the optimal spatial decay rate. Therefore, the evaluated values (rates) in this experiment also reflect the effect on the 1 event scenario on spatial decay rates' outcome and protocols' performance.



(a) Delivery ratio.

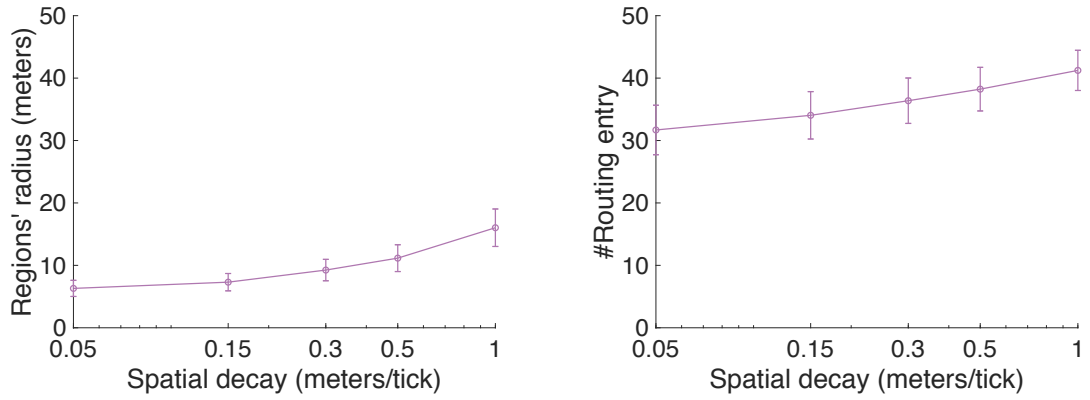
(b) Delivery latency.

(c) Number of hops to destination region and device.

(d) Network overhead.

Figure 4.20 Performance Analysis - varying the spatial decay rate (mobility).

Figure 4.21a illustrates the average radius of regions where messages are flooded. It can be clearly observed that as the spatial decaying gets more intense (i.e. meaning with higher spatial decay rates), the average radius increases, too. Figure 4.21b depicts the average table size for different values of spatial decaying. It is interesting to note that the average size increases with the spatial decaying rate. This is because larger regions can be difficult to merge with information received through advertisements or direct encounters (see Section 3.2.1), therefore new region information can only be added as new entries in the table.



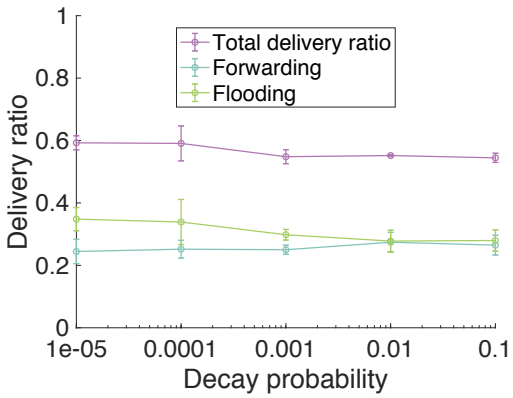
(a) Flooded region radius

(b) Average table size.

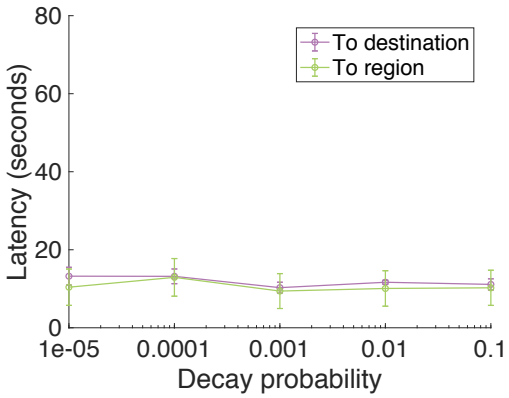
Figure 4.21 Flooded region radius and table size - varying the spatial decay rate (mobility).

Temporal Decay

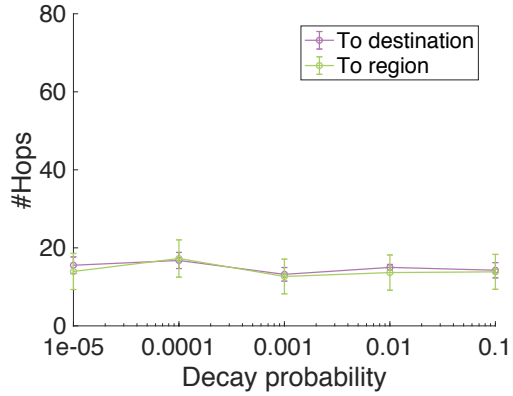
Here we experiment we different rates of temporal decaying by adjusting the probability of flipping bits every time a BF is decayed (see Section 3.2.1). In Figure 4.23 we observe that, for the current network setup, GeoHawk's performance is not significantly affected. As decaying gets faster (i.e. the respective decaying probability gets larger), the delivery ratio drops slightly. This is because regions' lifetime drops (shown in Figure 4.23), and, consequently, the average size of nodes' routing tables decreases (shown in Figure 4.23a). The latter is also dependent on the advertisement rate and the user speed. If advertisements were exchanged more frequently and users encountered other users more often, more regions would be added to the routing tables, countering the effect of reducing their lifetime.



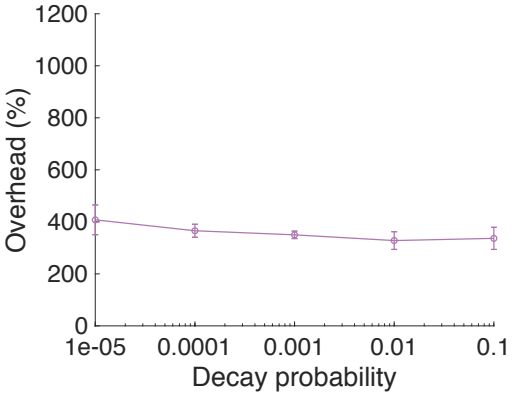
(a) Delivery ratio.



(b) Delivery latency.

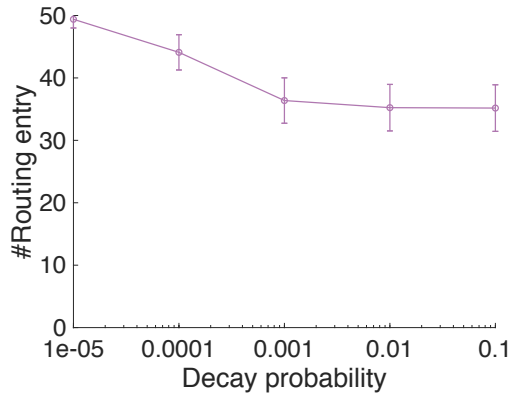


(c) Number of hops to destination region and device.

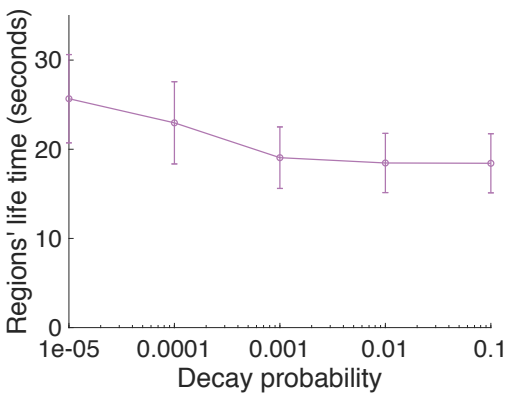


(d) Network overhead.

Figure 4.22 Performance analysis - varying the temporal decay rate (mobility).



(a) Average table size.

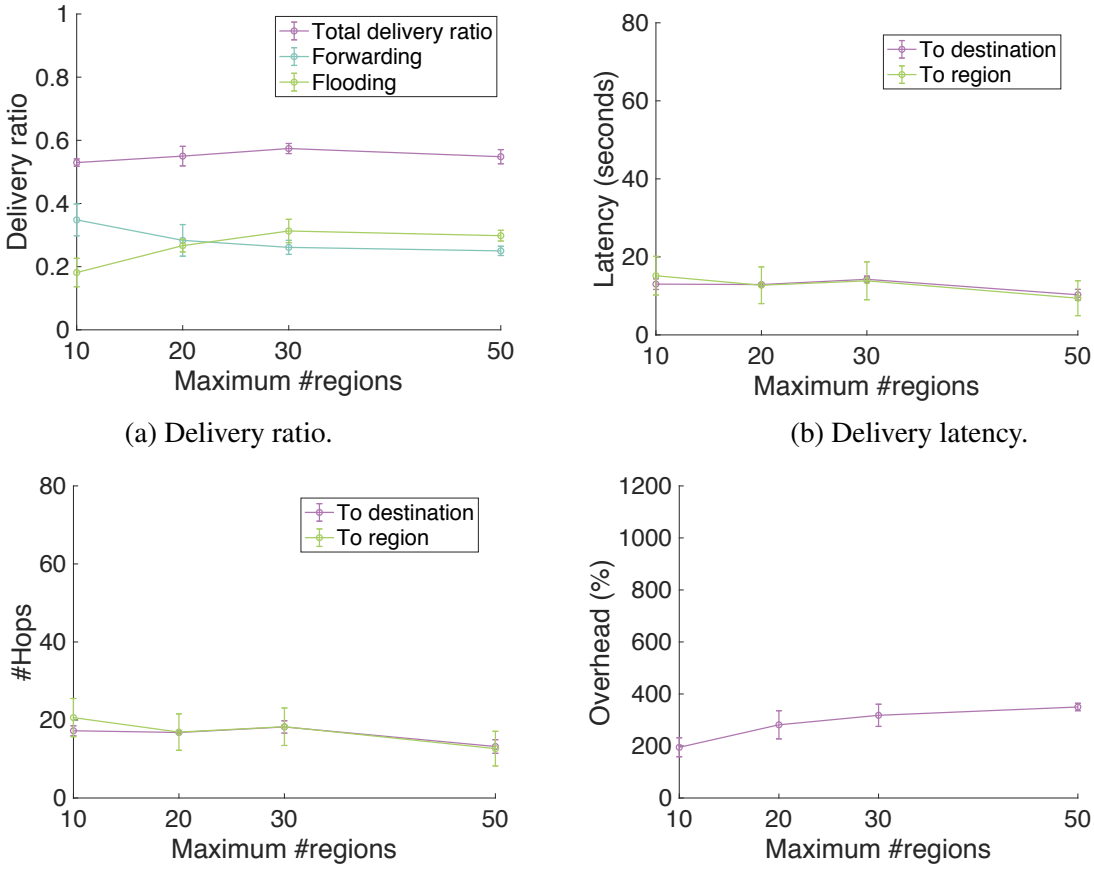


(b) Region lifetime.

Figure 4.23 Table size and region lifetime - varying the temporal decay rate (mobility).

Table Size

The size of the routing table, in combination with other factors affecting the lifetime of regions in tables, plays an important role in GeoHawk's performance. In this network setup, we show that increasing the table size does not affect the performance significantly, mainly because of the selected values for the advertisement frequency and region lifetime. In Figure 4.24 we do observe that the performance slightly improves, in terms of delivery latency, as the table size increases. At the same time the measured network overhead increases. This indicates more frequent flooding which also justifies the decrease in latency, because destination devices are more frequently reached by flooding than redirecting messages (which adds to the perceived latency).



(a) Delivery ratio.

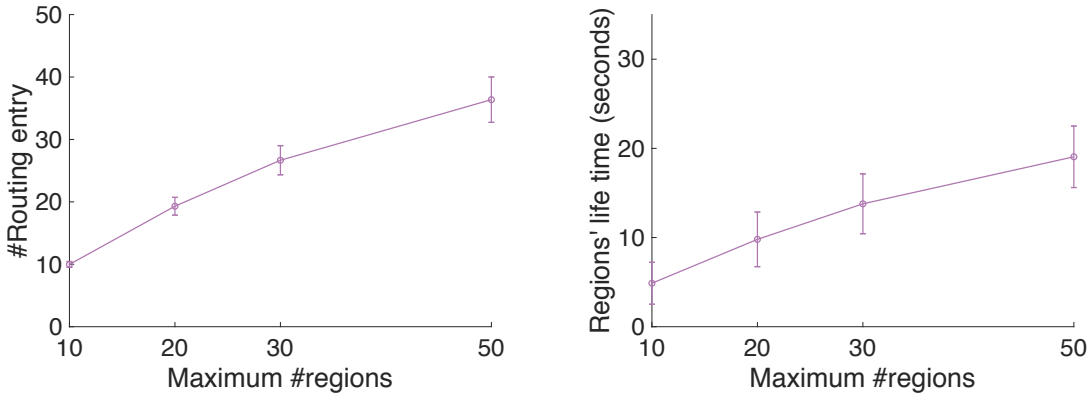
(b) Delivery latency.

(c) Number of hops to destination region and device.

(d) Network overhead.

Figure 4.24 Performance analysis - varying the routing table size (mobility).

In Figure 4.25a we observe that, as expected, the average table size increases as the maximum window size increases. However, when the maximum size is 50, the average size does not get above 35, which means that regions are deleted faster than created (through advertisements and direct encounters). We have observed larger performance variations when changing the maximum table size in scenarios where advertisements are sent more frequently; indeed, this led us to the selected value of maximum table size when comparing GeoHawk with the state of the art (see Section 4.5.2.)



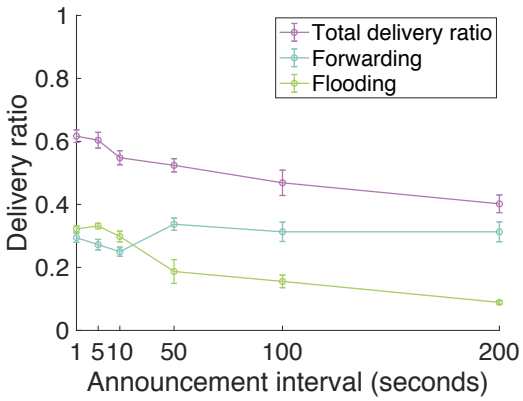
(a) Average table size.

(b) Region lifetime.

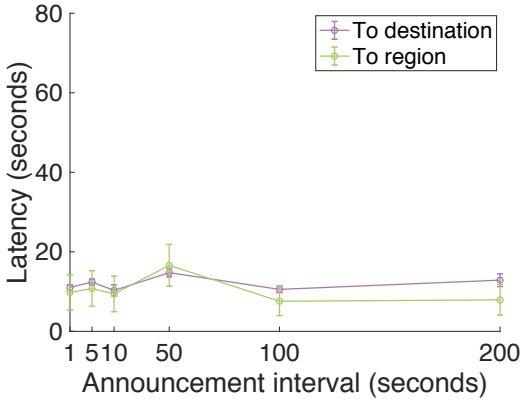
Figure 4.25 Table size and region lifetime - varying the routing table size (mobility).

Announcement Frequency

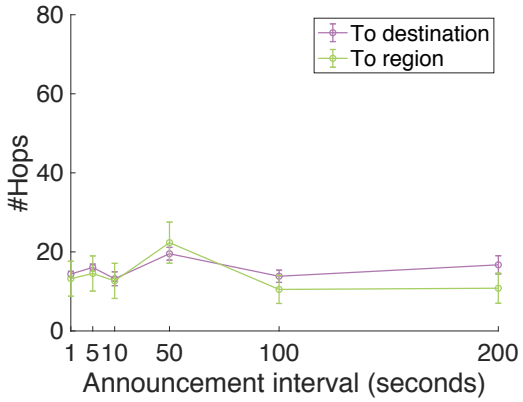
As mentioned above, the rate at which nodes generate new advertisements is crucial for the performance of GeoHawk. In Figure 4.26a we observe that GeoHawk's delivery ratio drops as the interval increases. Delivery latency is not significantly affected; fewer messages make it to the destination and the ones that do, make it in the roughly the same amount of time. Network overhead decreases but this is only because fewer messages make it to their destination (and flooded).



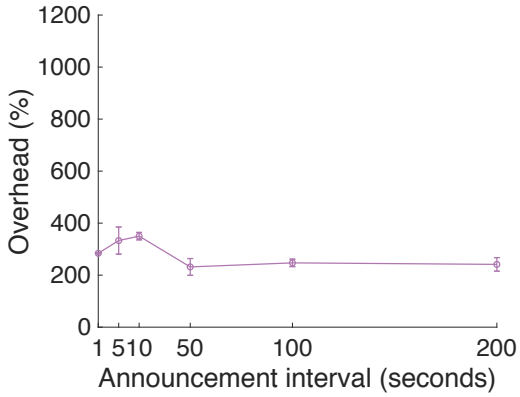
(a) Delivery ratio.



(b) Delivery latency.



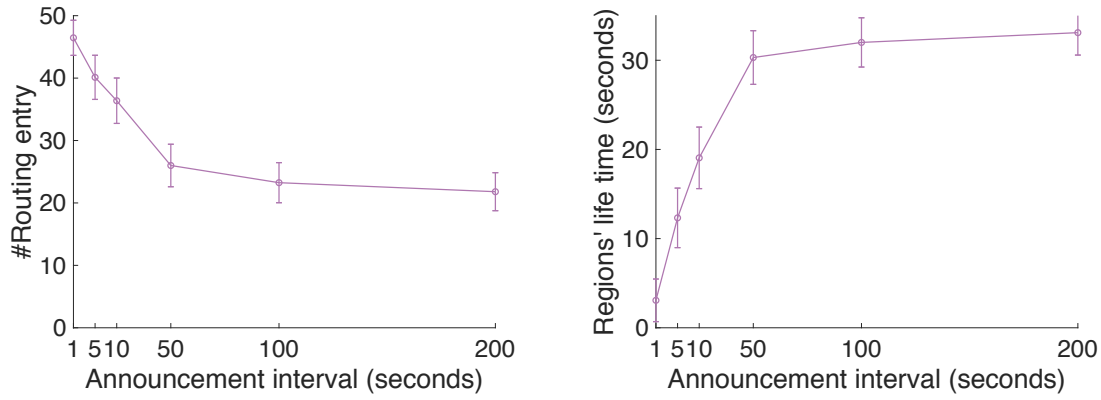
(c) Number of hops to destination region and device.



(d) Network overhead.

Figure 4.26 Performance analysis - varying the announcement frequency (mobility).

In Figure 4.27a, we observe that the average table size drops as the advertisement interval increases. This is because less routing information is being disseminated in the network. For this reason, regions stay longer in the routing tables until they reach their maximum lifetime, subsequently get temporally decayed, and discarded (Figure 4.27b). For short intervals (e.g. 1 advertisement every 1 or 5 seconds), their lifetime is very small, because they are quickly replaced by new routing information.



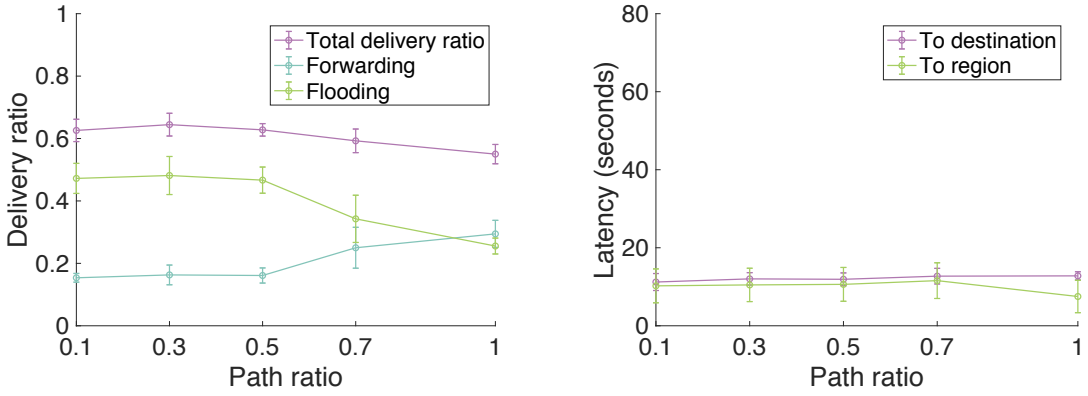
(a) Average table size.

(b) Region lifetime.

Figure 4.27 Table size and region lifetime - varying the announcement frequency (mobility).

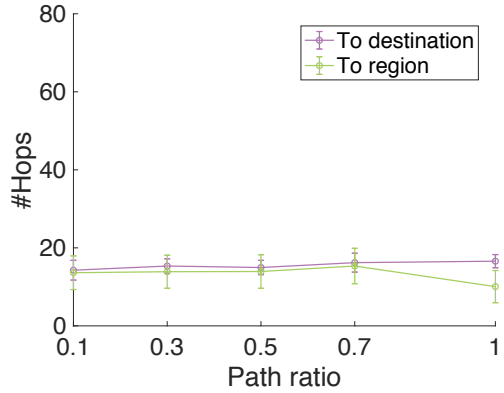
Preventing Spurious Flooding

GeoHawk's thresholds play a key role in minimising false positives, thus preventing spurious flooding. At the same if they are set too high, a large of FNs may result in significant performance degradation. In Figure 4.28a, we see that the delivery ratio drops as the value of the path ratio threshold increases. This is because the number of FNs increases, as shown in Figure 4.29. At the same time, the network overhead significantly decreases because the number of FPs also drops (and the respective number of TNs increases), as shown in Figure 4.29.

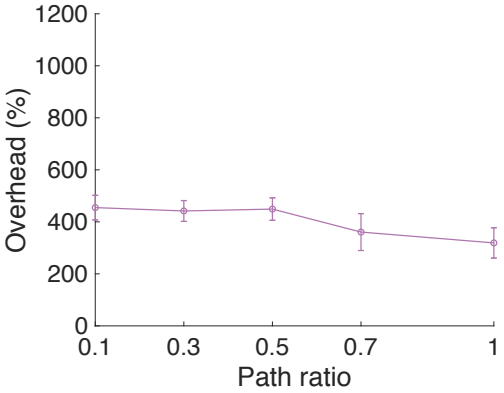


(a) Delivery ratio.

(b) Delivery latency.

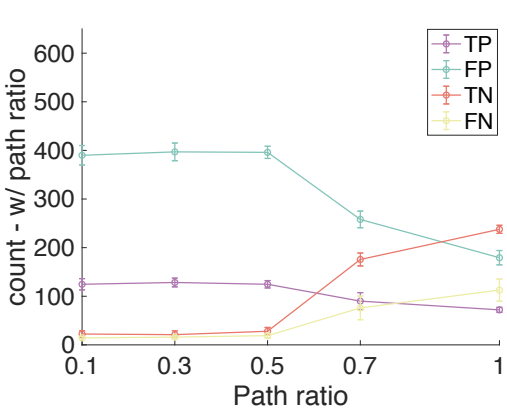


(c) Number of hops to destination region and device.

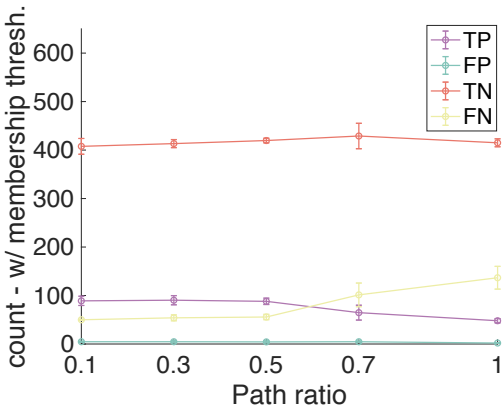


(d) Network overhead.

Figure 4.28 Performance analysis - varying the path ratio threshold (mobility).



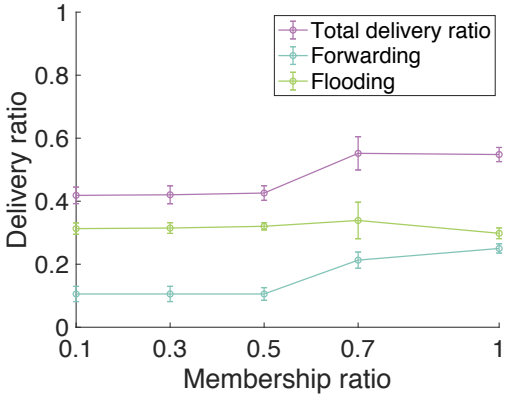
(a)



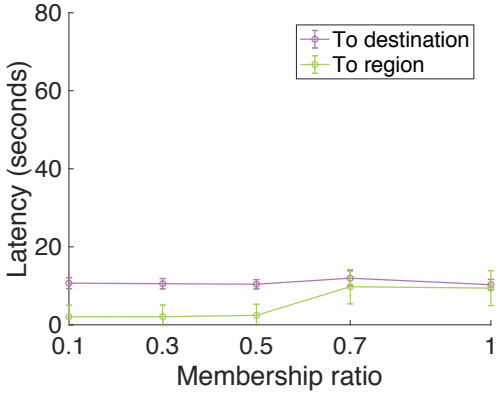
(b)

Figure 4.29 Preventing spurious flooding - varying the path ratio threshold (mobility).

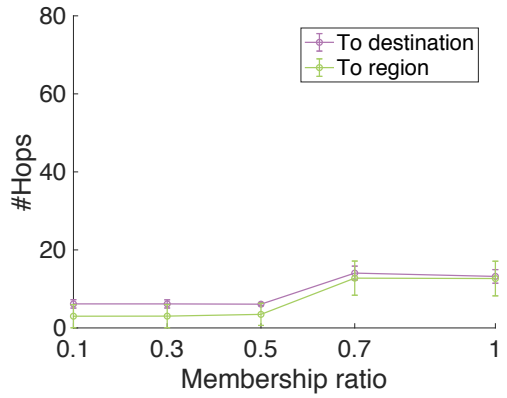
For the experiments below, we fixed the path ratio threshold to 0.9 which results in a very small number of FPs and a relatively large number of TNs. GeoHawk's performance when applying the membership threshold on top of the path ratio one is illustrated in Figure 4.30. Here, we observer that for small values of the threshold delivery ratio is small. This is because flooding is allowed even when the probability that a node matches the region's Bloom filter is low; this would be the result of matching the identifier of the destination node with a significantly weakened Bloom filter. The result of that is that messages are flooded in the wrong regions and never make it to their actual destination. This is also obvious by looking at Figure 4.30d where the network overhead is extremely high (due to spurious flooding) for small values of the membership threshold.



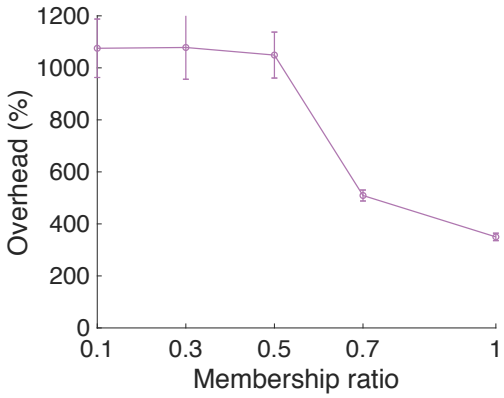
(a) Delivery ratio.



(b) Delivery latency.



(c) Number of hops to destination region and device.

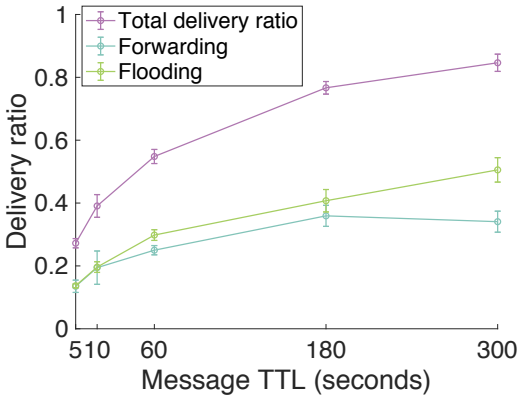


(d) Network overhead.

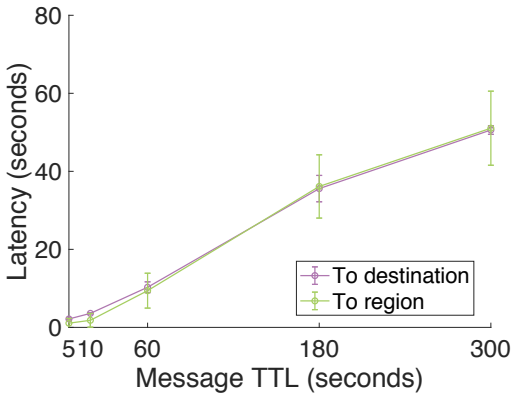
Figure 4.30 Performance analysis - varying the membership threshold (mobility).

Message TTL

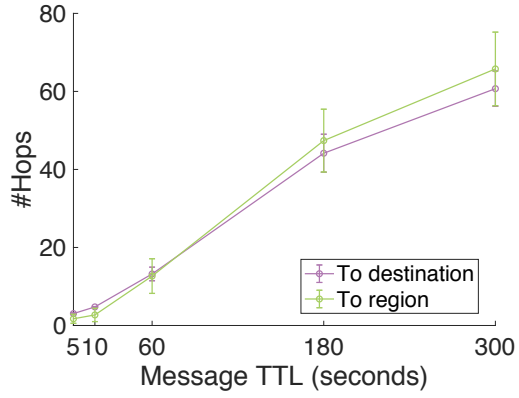
Below we experiment with the message TTL value and observe its (dramatic) impact on GeoHawk's performance. For very small values, GeoHawk's delivery ratio is poor as the majority of messages do not even have the chance to get to their destination region. At the same time, the measured message latency and network overhead are very low because the few messages that make it to their destination are nearby their source. As the TTL increases, the message delivery ratio increases along with the delivery latency and network overhead. For very large values of the TTL, over 80% of the messages make it to their destination, albeit with an average latency of around 50 seconds.



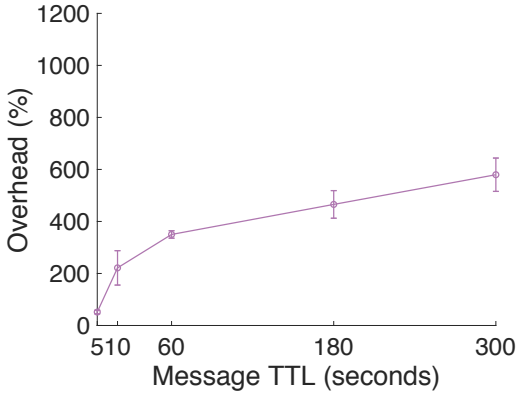
(a) Delivery ratio.



(b) Delivery latency.



(c) Number of hops to destination region and device



(d) Network overhead.

Figure 4.31 Performance analysis - varying the message TTL (mobility).

Figure 4.32 illustrates statistics about the regions where a message arrived. It is clear that as the TTL value increases significantly more messages are redirected and eventually make it to their destination.

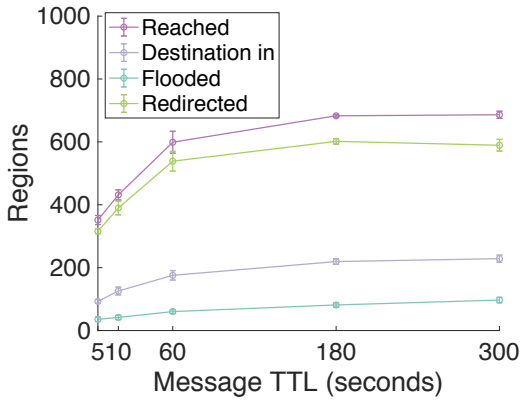
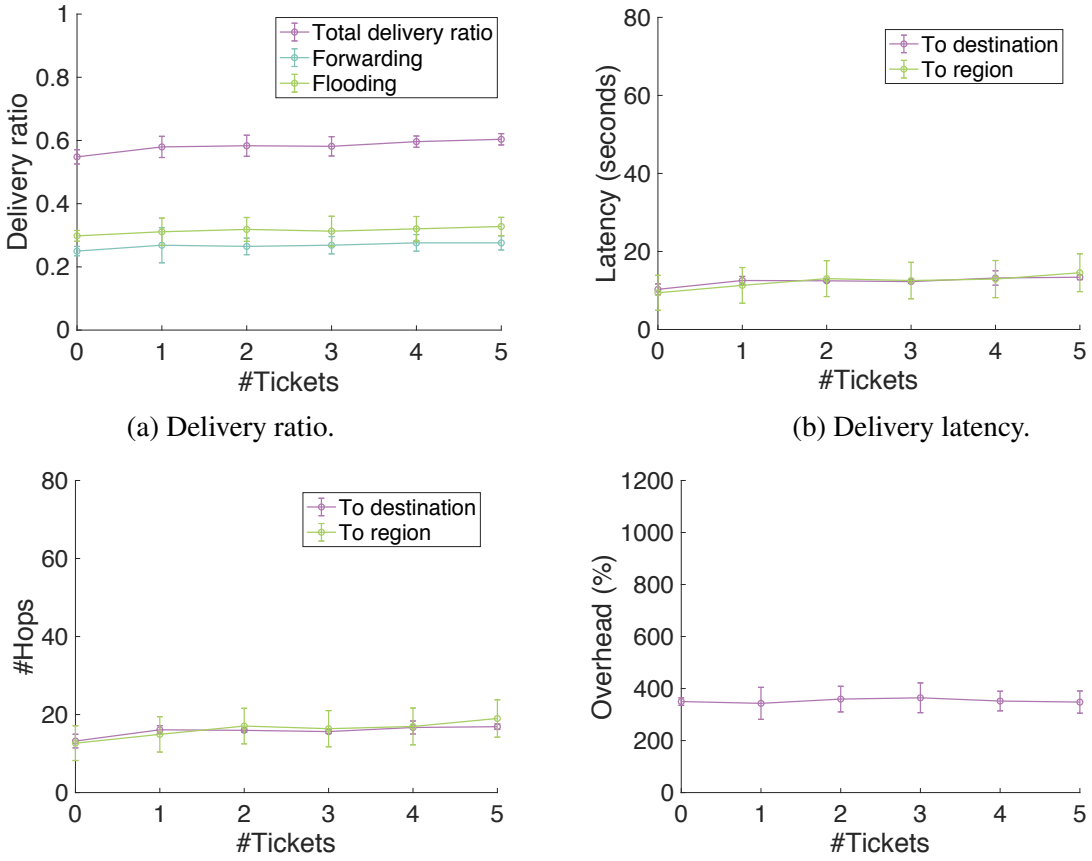


Figure 4.32 Region statistics - varying the message TTL (mobility).

It is clear that setting the value of the message TTL presents an important trade-off between the probability that the message will arrive to its destination and the time it will take to do so. Setting the TTL value can therefore be application specific; time-critical messages can be sent with a small TTL value whereas ensuring delivery can be done by setting the TTL to a relatively large value.

Message Replication

Below, we experiment with different values of the number of tickets assigned to a message. We observe that in this network setup, message replication plays a small positive role in GeoHawk's performance. In Figure 4.33, we observe that the average delivery ratio slightly increases along with the delivery latency. In this set of experiments, messages are not often replicated and when they do, they are eventually forwarded to the same network regions hitting similar regions.



(a) Delivery ratio.

(b) Delivery latency.

(c) Number of hops to destination region and device

(d) Network overhead.

Figure 4.33 Performance analysis - varying the number of tickets (mobility).

4.4 Discussion

Different application scenarios require different protocol configurations. However, from the extensively evaluation presented above, we observe trade-offs with respect to configuration parameters and GeoHawks' performance in terms of delivery rate, overhead and latency. We list bellow all parameters that noticeably affect GeoHAwk's performance:

Influence of parameters that prevent spurious flooding. Assigning the membership and path ratio thresholds with high (strict) values ($\geqslant 0.7$) plays a key role in minimising false positives, therefore preventing spurious flooding and resulting in significantly

lower network overhead. However, strict parameters can also lead to false negatives (i.e. when a node is prevented from flooding in areas where the device actually resides), attributing to the decline in delivery ratio. One can consider application needs and resources limitations in to drive a desirable balance.

Influence of routing entries' aging. Maintaining up-to-date routing tables is very important for GeoHawk's performance. Specific parameters affect how long a specific routing entry can exist in a device's routing table.

Spatial decaying results in expanding the radius of a routing entry (region) according to a pre-defined rate. This process is initiated from the time this entry is inserted into a node's routing table until its defined lifetime expires (region TTL). After that point, the region is temporally decayed, which weakens its semantic state by flipping the Bloom filter bits according to a pre-specified weakening interval.

A higher spatial decay will increase the regions' size at a fast rate, indicating that flooding events can increase delivery ratio at the same time it can also result in greater network overhead; i.e. flooding a large area will likely ensure that a message will make it to its destination. On the other hand, flooded messages will also reach a larger number of recipients consuming more network bandwidth. Assigning a small or zero value to the spatial decay rate, can adversely affect delivery ratio; i.e. by flooding in a very small region where the destination node have moved outside of its range.

The Region TTL parameter is related to spatial decay as it determines the extent of a regions' spatial growth. A large lifetime enables a region to grow large while a small value restricts growth. Thus, the same arguments described for spatial decay also hold for Region TTL.

A low temporal decay rate prolongs the lifetime of a routing entry, so that each and every node within a dense network will always have some routing information, which improves GeoHawk's performance. However, it can induce a large overhead as the quality of information cannot be guaranteed; i.e. as routing information is temporally decayed with a low rate, membership testing yields a probability that a specific node identifier has been inserted in the Bloom filter, which restls in flooding and the consequent increase of network overhead. In contrast, a high rate prevents the survival of stale routing information, which improves network overhead. Nevertheless, it can also lead to the rapid extinction of valuable information, which negatively affects delivery ratio and latency.

In order to set these values, one needs to study and understand the network deployment, especially the mobility characteristics, as well as the size and density of the network.

Influence of announcement transmission frequency and lifetime. Advertisements play a key role in acquiring and updating routing state, which improves the quality of routing decisions and, consequently, the rate of successful delivery. A higher transmission frequency ensures that nodes are provided with fresh state; a longer lifetime ensures that this information will reach a larger population of network devices. However, having a higher rate and a longer life expectancy cannot guarantee the quality of routing information and as a result can hurt delivery ratio and increase both network overhead and latency. High announcement rates and frequencies implies that buffer overflow is likely to take place, which is going to lead the protocol to lose valuable routing information, make poor decisions and reducing its delivery ratio and latency. On the other hand, low rates and shorter TTLs can contribute to stale information, poor

decisions and low delivery rates. Thus, determining a good value for these parameters requires understanding the network and mobility characteristics.

Influence of message lifetime. The message lifetime defines the "delivery interval" during which a message is valid. In crowded unicast scenarios, the delivery ratio increases with message lifetime; i.e. granting more time to deliver a message to its assigned destination within dense settings will likely contribute to the increase of delivery ratio. However, longer lifetimes imply the need for larger buffers to store (and carry) the messages and higher bandwidth to exchange them. Given the finite (and rather limited) nature of both of them, the increase in latency and network overhead is to be expected. Accordingly, limitations such as buffer availability should be considered, as the lack of buffer space can result in quick message extinction.

4.5 Comparing GeoHawk with the State-of-the-Art

In this section we compare Geohawk with existing protocols for routing in opportunistic networks. Our baseline protocol is Epidemic which floods messages in the network. Additionally, we experiment with WSR and PRoPHET. Given the very challenging network environment studied in this thesis we do not expect that any of these protocols would be able to perform well as the number of users and the resulting network density increases to realistic scales (e.g. thousands of devices in relatively small areas). Experimenting in such scales is problematic as it would require extremely long simulations that would not be possible to complete in time. Instead, we seek to "break" these protocols by decreasing the available buffer and increasing the number of circulated messages. The latter is done

implicitly by increasing the TTL value that messages carry, resulting in much longer message lifetimes.

Our evaluation focusses on the KPIs discussed in Section 4.1, namely delivery ratio, network overhead and message latency. All protocols are evaluated in static and mobile scenarios in networks spanning an area of $41 * 41m^2$. We experimented with different numbers of mobile devices (200, 500, 1000, 1500 and 2000) to examine the behaviour of the protocols when the density increases to realistic numbers (e.g. 1.2 $users/m^2$). As noted above, it is generally not possible to scale the whole network up to realistic numbers. The transmission range and rate were set to 5 metres and 250Kbps, respectively, for all experiments. The duration of each simulation was configured to be 1800 seconds. Time in the ONE simulator progresses through periodic ticks of the virtual clock; we set the period to be 0.5 seconds. The last 100 seconds of each simulation are ignored (cool down period). Data messages (100KB) are injected in mobile devices that are selected uniformly at random; we inject 10 messages every 100 seconds. Mobile devices issue advertisements every 10 seconds and each advertisement can be forwarded up to 15 hops.

4.5.1 Performance Comparison in a Static Network Topology

Here we evaluate the performance of GeoHawk in a static network scenario. We vary the total number of users in the network and assess how the protocols scale up as the network density increases. The values of GeoHawk parameters in the static scenario are derived from the default settings mentioned in Section 4.3.1. To maximise GeoHawk's performance (given the empirical evidence collected above), we increase the region TTL to 50s and decreased the cardinality to 0.5. We also relaxed the r_m threshold to 0, and increased the membership value to 1 as no form of spatial or temporal decaying occurs

in this scenario. Since GeoHawk and WSR both share similar design principles, it is necessary to demonstrate how differences in behaviour can affect a protocols' efficiency. Thus, as a rule of thumb, WSR parameters will always be assigned with same values assigned for GeoHawk. For PRoPHET we set the ageing frequency to 30 seconds. All parameters and values used in the static scenario are shown in Table 4.3.

In Figure 4.34 we see that GeoHawk performs very well in all KPIs for all studied network densities. More specifically, in Figure 4.34a we observe that GeoHawk maintains a high delivery ratio that is comparable to Epidemic. WSR's performance drops rapidly as the number of users in the network increases. This is because WSR does not employ any flooding, nor replication, mechanism in the network and only relies on advertisements to disseminate routing information. As a result, when the network is dense (i.e. 1.2 users per m^2 for 2000 users in total), less than half of the messages make it to their destination. PRoPHET[15] and Epidemic perform equally well to GeoHawk for the specific network setup. As discussed above (and shown in the next section), both protocols would suffer significant drops in their performance as the total size of the network increases, while being equally dense; (1) epidemic relies solely on flooding and as the number of users increases to the targeted numbers (i.e. 10s or 100s of thousands), routing would collapse due to (a) the limited amount of buffers ub mobile devices and (b) the finite and commonly limited network bandwidth; (2) PRoPHET relies on keeping delivery probabilities about all encountered nodes, therefore it would require extremely large routing tables to deal with the very large number of users in the network. On the other hand, GeoHawk strikes the right

[15]The low performance of PRoPHET for 500 users is a simulation artefact, because PRoPHET nodes build routing state (delivery probabilities for all nodes in the network) by exchanging messages with other nodes upon a direct encounter. In the static scenario such encounters happen only once at the beginning of the simulation. As a result, nodes do not get the chance to exchange information received by other nodes to their neighbours more than once. For 200 users the resulting routing state is incomplete and the performance is very low in the ONE simulator.

Table 4.3 Default values for performance comparison (static scenario).

Common for all protocols

Parameter	Value
#hosts	[200, 504, 1512, 2016] nodes
Transmission range	5m
Transmission rate	250Kbps
Simulation duration	1800s
Update interval	0.5
Cool down period	100s
Message size	100KB
Message TTL	30s
Buffer size	1M

PRoPHET

Parameter	Value
Aging delivery predictions frequency	30s

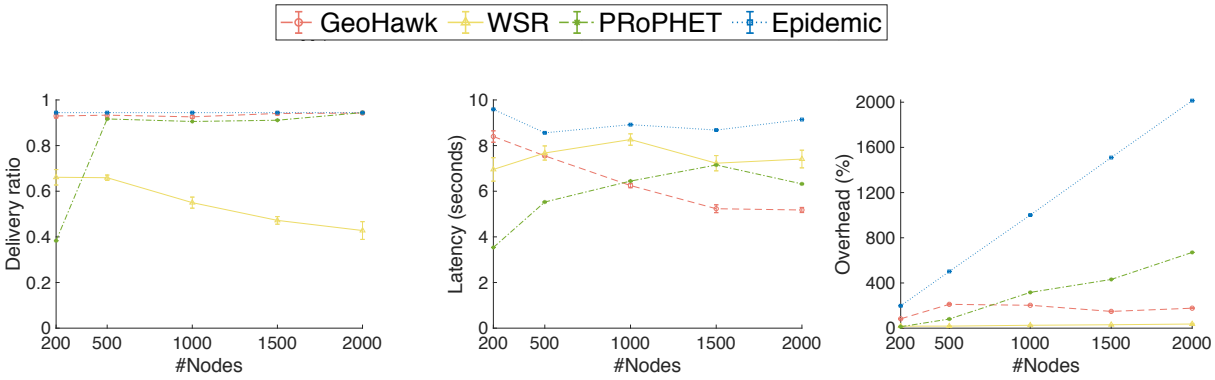
WSR

Parameter	Value
Announcement frequency	25s
Announcement TTL	15 hops
Bloom filter size	4000bits
Spatial decay	0.3 m/s
Decay probability	0.001
Cardinality threshold	5bits
Table size	30

GeoHawk (in addition to WSR values)

Parameter	Value
Table size	30 FIFO
Region TTL	50s
Spatial decay	0 m/s
Cardinality threshold	0.5
Tickets	0
Path ratio threshold	0
Membership probability threshold	1

balance between aggregating routing information and disseminating it in advertisements and relying on flooding when the message is close to the destination. As the number of nodes increases, the stress is only on the Bloom filters that aggregate information, therefore scaling GeoHawk would only require increasing the size of the Bloom filters. The associated network overhead would be negligible compared to the one induced by blind flooding (Epidemic) or having to constantly exchange routing information about all nodes in the network (PRoPHET).



(a) Delivery ratio. (b) Delivery latency. (c) Network overhead.

Figure 4.34 Performance comparison - varying network density (static scenario).

GeoHawk performs very well with respect to the induced overhead. Naturally, the overhead increases with the numbers of users in the network. Flooding a message in the same exact area would result in higher network overhead if the area is more densely populated, which is the case when the number of users increases. On the other hand, it is obvious that Epidemic performs very badly in terms of network overhead; for 2000 users the overhead is 2000%. As mentioned above (and investigated in the next section), this becomes a significant problem when the buffer size of devices decreases and the number of messages increases; Epidemic would never scale up to the targeted network sizes; unfortunately, it is currently computationally infeasible to run simulations at these scales. WSR's overhead is the lowest but, as discussed above, this comes at a significant cost in terms of delivery ratio. WSR does not flood nor replicate but, as a result of that,

cannot deliver messages to their destinations, especially as the network density increases. PRoPHET performs the best when it comes to network overhead because it does not flood messages; it only replicates them based on the information it has about the delivery probability for each node in the network. The comparison with GeoHawk here is unfair. As mentioned above, PRoPHET achieves such a high delivery ratio because nodes essentially have near-perfect information about all nodes in the network. In reality, this would be impossible in large-scale dense deployments given the limited memory and bandwidth.

In Figure 4.34b we observe that GeoHawk latency decreases as the network density increases. This is because more routing information is available to all network nodes through advertisement dissemination[16]. WSR will keep forwarding the message until the destination is reached, therefore as the network density increases it becomes more difficult to do so (lower delivery ratio) and when it happens it takes more time (higher latency). Epidemic performs the worst as the message will be flooded to all nodes in the network before reaching its destination. PRoPHET performs similarly to GeoHawk.

4.5.2 Performance Comparison in the Presence of Mobility

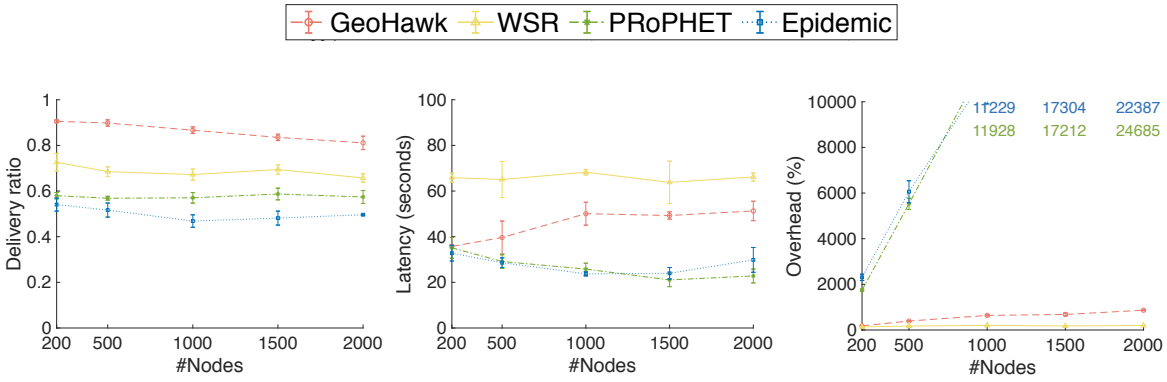
We have evaluated GeoHawk, WSR, PRoPHET and Epidemic performance in network scenarios with mobility that approximate the studied use cases. We have looked at scenarios with 1, 2 and 4 events, as described in Section 4.2.4. Nodes move with speed ranging between [0.26, 0.3 m/s] within events and [0.5,1m/s] when moving across events. Most parameter values are extracted from the default settings used in Section 4.3.2. Based on the collected empirical evidence, we increased message lifetime to 250s (opting in for higher delivery ratio over latency in GeoHawks) and reduced the buffer size to 1MB. We

[16]Note that we keep the advertisement interval fixed for all these experiment.

also extended the region TTL to 50s to ensure that all nodes have always some routing information. The Bloom filter size was increased to 6000bits and the r_m threshold was reduced to 0.7 in order to improve the delivery ratio. As mentioned previously WSR parameters are drawn from GeoHawk's ones. We set PRoPHET's ageing frequency to 30s. All parameters and values for environments with mobility are shown in Table 4.4.

Varying the Network Density

In the first set of experiments we compare the performance of GeoHawk with that of the baseline protocols when varying the network density. We do so by increasing the number of devices (network density) while keeping the network area fixed. Figure 4.35 illustrates the performance comparison when a single event is taking place in the network; i.e. users move around the whole network area.



(a) Delivery ratio. (b) Delivery latency. (c) Network overhead.

Figure 4.35 Performance comparison - varying network density (mobility - 1 events).

GeoHawk outperforms all other protocols with respect to delivery ratio. GeoHawk delivers around 90% of the messages in the network when the density is not extreme (i.e. 1 device per $3.3m^2$ or 500 devices) and over 80% of the messages when 2000 users roam the network area (i.e. 1.2 devices per m^2). WSR performs worse than GeoHawk as it tries to find the destination node directly. Our conjecture is that WSR would perform badly in real

Table 4.4 Default values for performance comparison (mobility).

Common for all protocols

Parameter	Value
#hosts	[200, 500, 1500, 2000] nodes
Transmission range	5m
Transmission rate	250Kbps
Simulation duration	1800s
Update interval	0.5
Cool down period	100s
Message size	100KB
Message TTL	250s
Buffer size	1M
Movement speed within Event areas	[0.26,0.3] m/s
Movement speed between Events	[0.5,1] m/s

PRoPHET

Parameter	Value
Aging delivery predictions frequency	30s

WSR

Parameter	Value
Announcement frequency	10s
Announcement TTL	15 hops
Bloom filter size	6000bits
Table size	50
Decay probability	0.001
Cardinality threshold	5bits

GeoHawk (in addition to WSR values)

Parameter	Value
Table size	50 FIFO
Spatial decay	0.28 m/s
Region TTL	50s
Cardinality threshold	0.5
Tickets	0
Path ratio threshold	0.7
Membership probability threshold	0.9

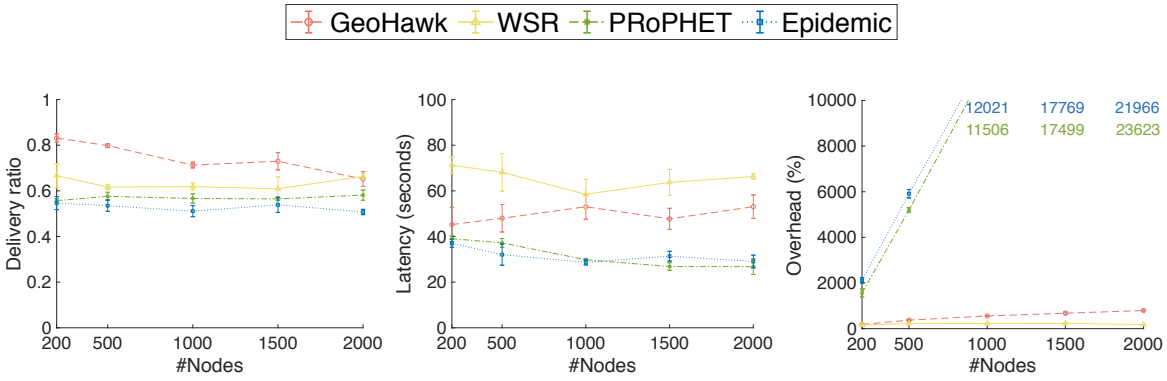
large-scale events with 10s or 100s of thousands of users in large areas, because it would then be impossible to find specific destinations without flooding. As discussed above, it would unfortunately be impossible to simulate such large networks with the ONE simulator. Epidemic and PRoPHET perform poorly delivering less than 60% of the messages to their destinations. As discussed above and shown in the next section, this is because of the relatively large message TTL value we have used in these comparisons. Messages stay in the network for longer even after one of these has been delivered to the destination. This, in combination with the relatively small buffer in mobile devices, results in a very large number of messages that need to be stored and exchanged upon every encounter between mobile devices. Note that PRoPHET used replication and Epidemic merely floods the messages. This is clear evidence that both of these protocols would perform very poorly in large-scale scenarios.

Figure 4.35c depicts the measured delivery latency for all protocols. The delivery ratio - latency trade-off is apparent in the figure. WSR relies on unicast forwarding until it hits the destination node, which can only work if the message TTL is large. GeoHawk behaves much better as it only needs to get close to where the destination is believed to reside. Then it floods the message paying a small price in terms of network overhead. If the message is routed to a region where the probability that the node is in it is low, the message is redirected. That is why larger message TTL values also benefit GeoHawk. Epidemic and PRoPHET perform better but the recorded latencies are only for the delivered messages which are significantly fewer compared to the ones delivered by GeoHawk.

Figure 4.36 illustrates the network overhead for all different protocols as the density of the network increases. As shown in the figure, both Epidemic and PRoPHET are just not the right protocols for dense network environments. Epidemic floods all messages whereas

PRoPHET relies on replication. The reason why PRoPHET performs so poorly in terms of network overhead is because of the large message TTL which results in messages being in the network for a long time period. Unsurprisingly, WSR performs the best compared to all other protocols, as it does not flood nor replicate messages; a single message will be forwarded again and again until it reaches its destination. GeoHawk strikes the right balance between delivery ratio, latency and network overhead. It delivers most messages, while keeping the overhead at acceptable levels.

Figure 4.35b illustrates the performance comparison when two events are taking place in the network; i.e. users move within an event area but also across events. This is a more challenging environment because inter-event message advertisement dissemination depends on the limited amount of devices moving across events in time.

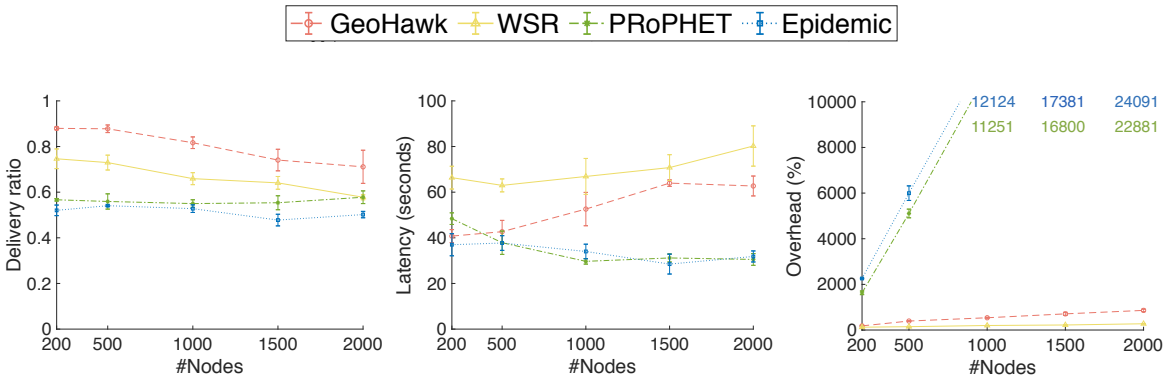


(a) Delivery ratio. (b) Delivery latency. (c) Network overhead.

Figure 4.36 Performance comparison - varying network density (mobility - 2 events).

As shown in Figure 4.36a, the delivery ratio for all studied protocols drops for the aforementioned reasons. At the same time, delivery latency also increases; region information is of lower quality due to the multiple events, therefore messages take longer to get to their destination. Network overhead (Figure 4.36c) is not significantly affected by the number of events and the cross-event mobility.

Finally, in Figure 4.37 we present the results when 4 events are concurrently taking place in the network. In comparison to the delivery ratio for two events (see Figure 4.36), GeoHawk performs better as shown in Figure 4.37 . This is because of the way events are distributed within a map. We keep the size of the event are constant, therefore in the 4-event scenario the area in-between event areas is much smaller compared to the 2-event scenario. As a result, it is easier to disseminate routing information and data messages across events. In all figures we see the same trends as above; delivery ratio, latency and network overhead are all comparable to the behaviour and performance of 1 Event, where delivery ratio is decreased, latency is increased and network overhead remains at very similar levels as the network density increases.



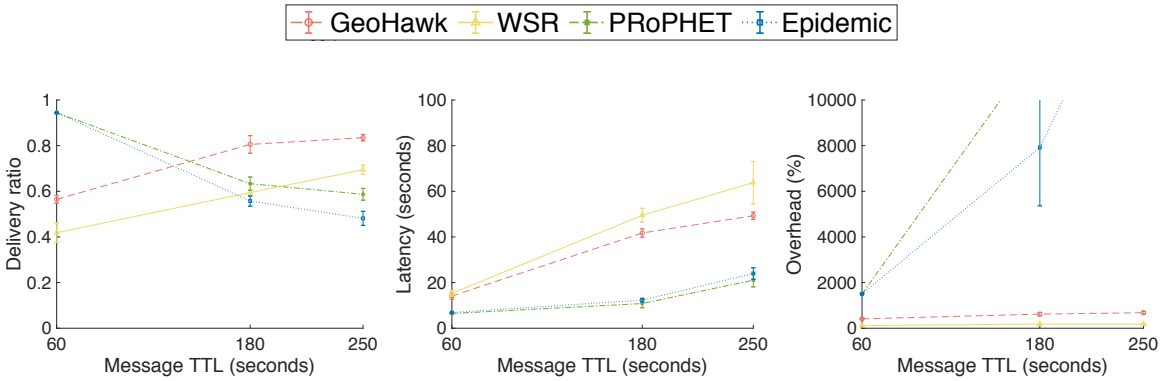
(a) Delivery ratio. (b) Delivery latency. (c) Network overhead.

Figure 4.37 Performance comparison - varying network density (mobility - 4 events).

Varying the Message TTL

In the previous sections, we have discussed the trade-off between delivery ratio and latency when adjusting the message TTL. At the same time, we have showed that with relatively large values of the TTL both Epidemic and PRoPHET perform poorly. This emulates well a large-scale deployment with many messages and small device buffers. In this section,

we investigate this trade-off by experimenting in the presence of mobility (and a varying number of events) while changing the value of message TTL. Figure 4.38 illustrates the performance comparison when a single event is taking place covering the whole network area.



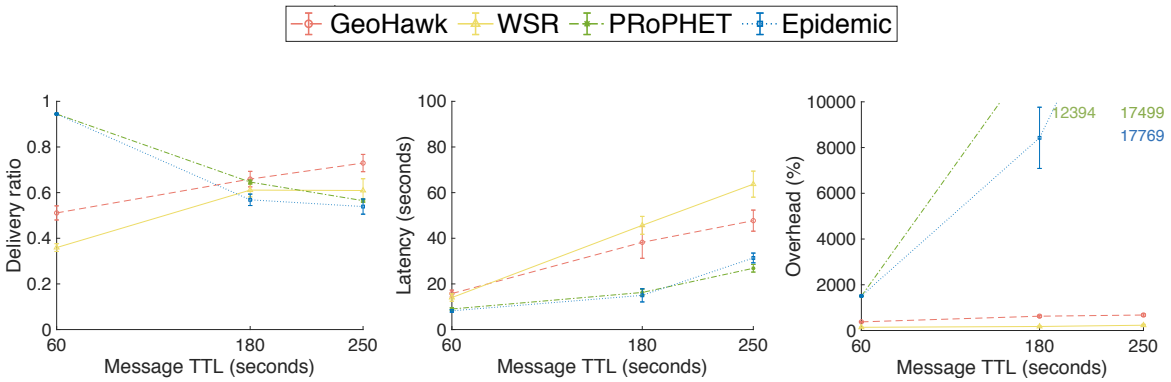
(a) Delivery ratio. (b) Delivery latency. (c) Network overhead.

Figure 4.38 Performance comparison - varying the message TTL (mobility - 1 event).

We observe that GeoHawk's achieved delivery ratio increases with the message TTL.When messages reach regions and flooding is prevented, they are redirected and the chances that they will (1) reach the correct region and (2) get flooded increase. For larger values of TTL multiple such redirections are possible, hence the increase in delivery ratio in Figure 4.38a and latency in Figure 4.38a. WSR behaves in a similar way, although the performance is always worse compared to GeoHawk. Epidemic delivers almost all messages to their destination when the TTL value is small (at the expense of significantly large network overhead). However, as the TTL value increases and more (identical) messages exist in the network, Epidemic performance drops dramatically. We anticipate that Epidemic would perform very poorly in a large-scale network because of the presence of a large amount of messages, small device buffers and finite bandwidth. PRoPHET's performance drops as well for the same reasons; none of these protocols is meant to scale up to very dense opportunistic networks. Figure 4.38c depicts the induced network overhead in the network due to message delivery for all studied protocols. As mentioned above,

Epidemic achieves very high delivery ratio for small TTL values at the cost of transmitting a very large amount of messages. This would not work in a large-scale deployment and be extremely energy hungry. PRoPHET's network overhead is also unacceptable high due to to replication in the network.

Figure4.39 illustrates the performance comparison when two events are taking place in the network. As mentioned previously, this type of environment is more challenging as stability of network topology and advertisement dissemination depends on the limited amount of devices moving across events.

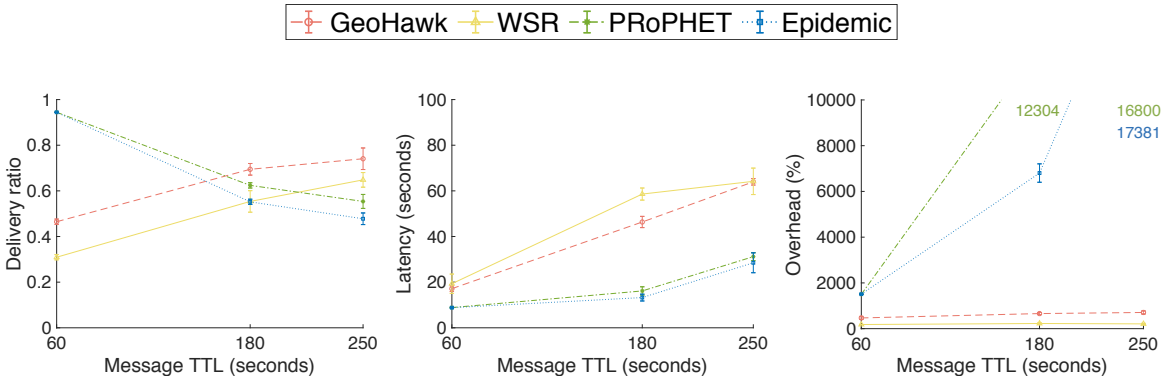


(a) Delivery ratio. (b) Delivery latency. (c) Network overhead.

Figure 4.39 Performance comparison - varying the message TTL (mobility - 2 events).

In comparison with the 1-event scenario, we observe in Figure 4.39a a drop in the overall performance of all studied protocols for the aforementioned reasons. As with the 1-event scenario, the message lifetime significantly affects the delivery ratio, latency and network overhead. GeoHawk and WSR delivery ratio increases with the message TTL, while Epidemic and PRoPHETs' decreases. Likewise, latency for all studied protocols increases with larger message TTL (Figure 4.39b). Network overhead on the other hand significantly increases with large values for Epidemic and PRoPHET, while GeoHawk and WSR maintain a similar yet better performance in terms of overhead (see Figure 4.39b).

Finally, in Figure 4.40 we present the results when 4 events are concurrently taking place in the network. The overall performance for all studied protocols is similar to the performance measured for the 1 event scenario. This was discussed above. In Figure 4.40 we observe that the large TTL value increases and more (identical) messages exist in the network. Thus, Epidemic's performance drops. On the contrary, Large TTL values provide an opportunity for GeoHawk and WSR to reach the correct region.



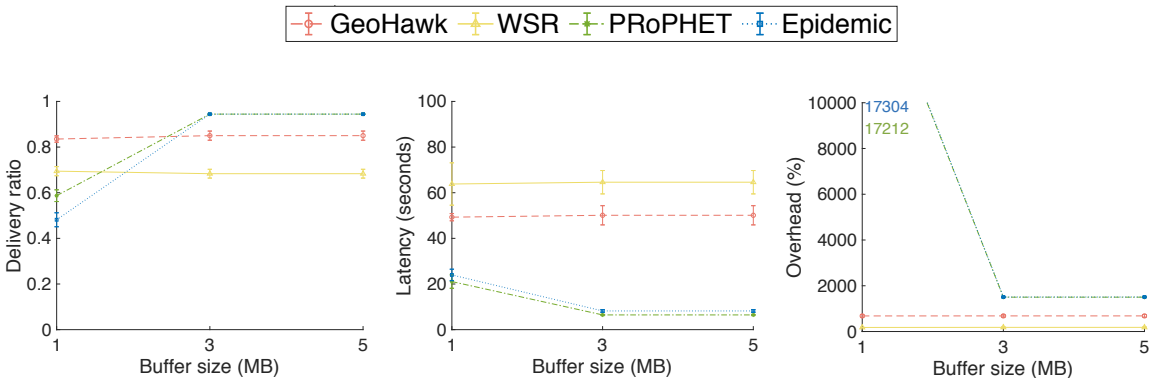
(a) Delivery ratio. (b) Delivery latency. (c) Network overhead.

Figure 4.40 Performance comparison - varying the message TTL (mobility - 4 events).

Varying the Size of the Device Buffer

In this subsection we explore how the devices' buffer size affects the performance of the studied protocols. Protocols that heavily rely on extensive flooding (Epidemic) and replication (PRoPHET) make heavy use of device buffers. As network density and messages increases, more and more buffers would be required. Increasing the size of the buffer is not panacea as the transmission rate in the network is limited and nodes move. In Figure 4.41 we see that the delivery ratio achieved by GeoHawk is not affected by the available device buffer. The illustrated results are for the 1-event scenario. This is an important advantage of our protocol; even with large numbers of devices there is no need to excessively store

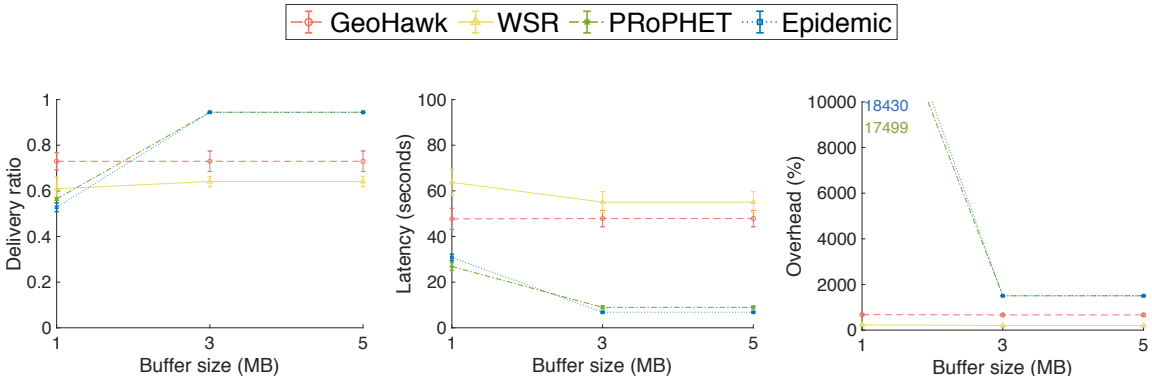
messages as these are forwarded and deleted from the buffer of the source device and, eventually, flooded in a confined region. In turn, this means that our protocol would scale very well with the size and density of the network. The same goes for WSR, although it performs worse than GeoHawk for reasons explained above. On the other hand, we see that both Epidemic and PRoPHET heavily depend on large buffers. This dependence increases with the number of devices and messages in the network, which makes them poor choices for our studied environment. GeoHawk delivers more than 80% of the messages to their destinations for all different buffer sizes. On the other hand, both Epidemic and PRoPHET perform very poorly when the buffer size is 1MB. This can be easily explained by looking at Figure 4.41c. They both induce an extremely high amount of network overhead (over 15,000%). When the buffer size is small then nodes keep exchanging messages among each other but these messages are quickly deleted from the buffer to make space for a new message. This keeps happening throughout the whole duration of the simulation and only around 50% - 60% of the messages end up being delivered. Latency follows a similar trend. GeoHawk and WSR are not affected by the buffer size (although latency is high due to the large message TTL value - see Section 4.5.2), whereas latency increases when the buffer size is 1MB for both Epidemic and PRoPHET.



(a) Delivery ratio. (b) Delivery latency. (c) Network overhead.

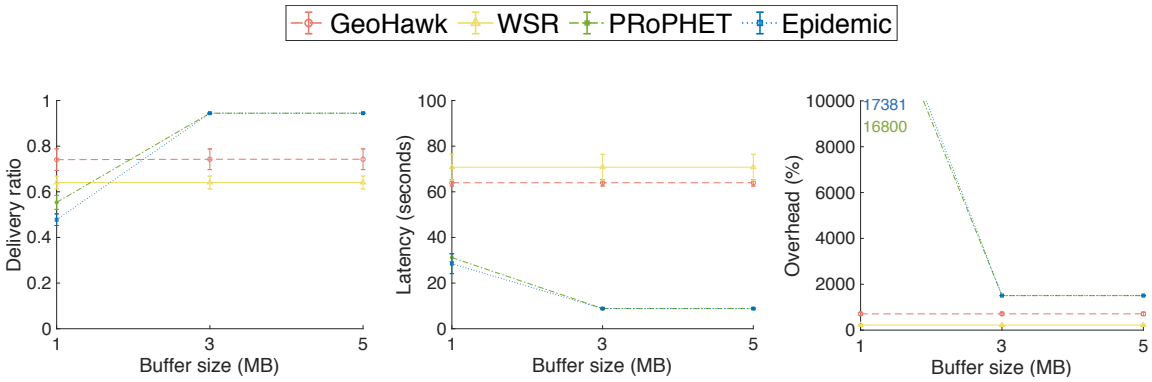
Figure 4.41 Performance comparison - varying the size of the device buffer (mobility - 1 event).

Figures 4.42 and 4.43 illustrate the measured performance when varying the size of the device buffer in the 2 and 4-event scenarios. Results follow the same trends; performance is slightly lower compared to the 1-event scenario; GeoHawk's performance is not affected by the buffer size. Epidemic and PRoPHET produce unacceptable large amounts of network overhead (especially when the buffer size is 1MB) and would not be able to support routing in large-scale deployments with a very large number of users and messages.



(a) Delivery ratio. (b) Delivery latency. (c) Network overhead.

Figure 4.42 Performance comparison - varying the size of the device buffer (mobility - 2 event).



(a) Delivery ratio. (b) Delivery latency. (c) Network overhead.

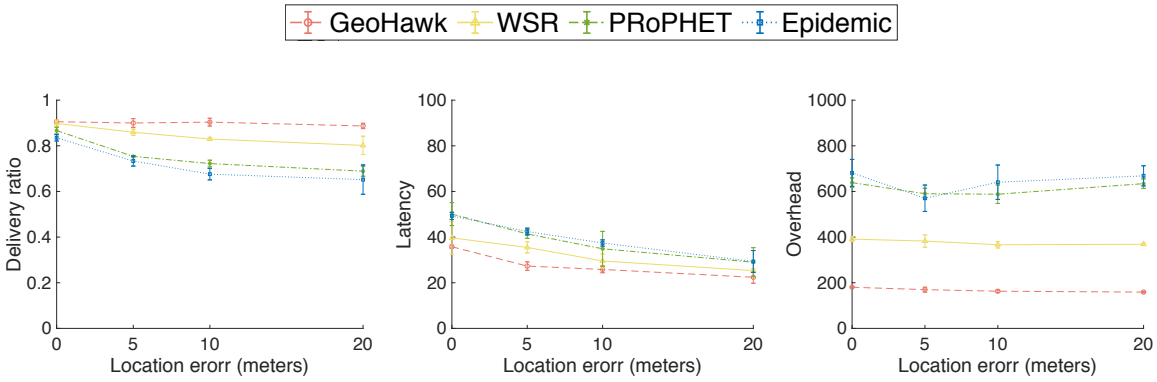
Figure 4.43 Performance comparison - varying the size of the device buffer (mobility - 4 event).

4.5.3 GeoHawk's Performance Under Location Uncertainty

In the previous sections we investigated GeoHawk's performance in 1 event scenarios, assuming that each device had always had access to its own location. As discussed in Section 3.4, devices may not always have access to very accurate estimates of their own location (e.g. through GPS). This may be because part of the network area is indoors or because GPS is disabled (or accessed relatively infrequently) when battery need to be preserved for a long time period (e.g. for a multi-day music festival). We have introduced a collaborative localisation mechanism that would enable GeoHawk users/devices to operate with estimates of their location, and, in this section we aim at assessing its suitability. Evaluating such a collaborative localisation mechanism using the ONE simulator has proven a challenging task. We developed a model for introducing PDR error which enables devices to collaboratively retrieve better estimates of their location.

Before exploring the collaborative localisation mechanism, we investigate GeoHawk's performance when combined with a simple simulation model that introduces (simulated) errors in the device's location. Location estimation in this simple model is done as follows. Every time a device needs access to its location (i.e. to generate a new announcement or to exchange location/region information with an encountered device) it draws its location from a bi-variate normal distribution; its mean is the actual device location in the two dimensional space (which we can retrieve from the simulator) and the coordinates are uncorrelated, which means that the covariance matrix is of the form $\begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$. We adjust the covariance matrix to induce different amounts of errors in the simulated system. The confidence of a node about its location (which is translated to a region radius) is drawn uniformly at random from $(0, \sigma]$.

Figure 4.44 illustrate GeoHawks' performance in a single event when varying the amount of (simulated) error. We have applied the same settings investigated in Section 4.5.2. The results show that GeoHawk can maintain good performance even with localisations errors. In Figure 4.44a we see that the delivery ratio is affected by large errors. This is expected as routing decisions rely on the accuracy of information propagated within the network. However, GeoHawk takes into account the uncertainty of the location of all nodes that are being aggregated (see Section 3.2), enabling it to sustain moderate localisation errors. In Figure 4.44b we observe that GeoHawks' performance improves, in terms of delivery latency, as error increases. This is because the size of our environment ($41 * 41m$) is proportional to the large amount of errors (10 and $20m$); an error of 20m is equivalent to half of our network size. As a result, a large radius will end up covering large proportion of the network, leading to lowering the delivery latency and at the same time increasing network overhead (see Figures 4.44b and 4.44c)

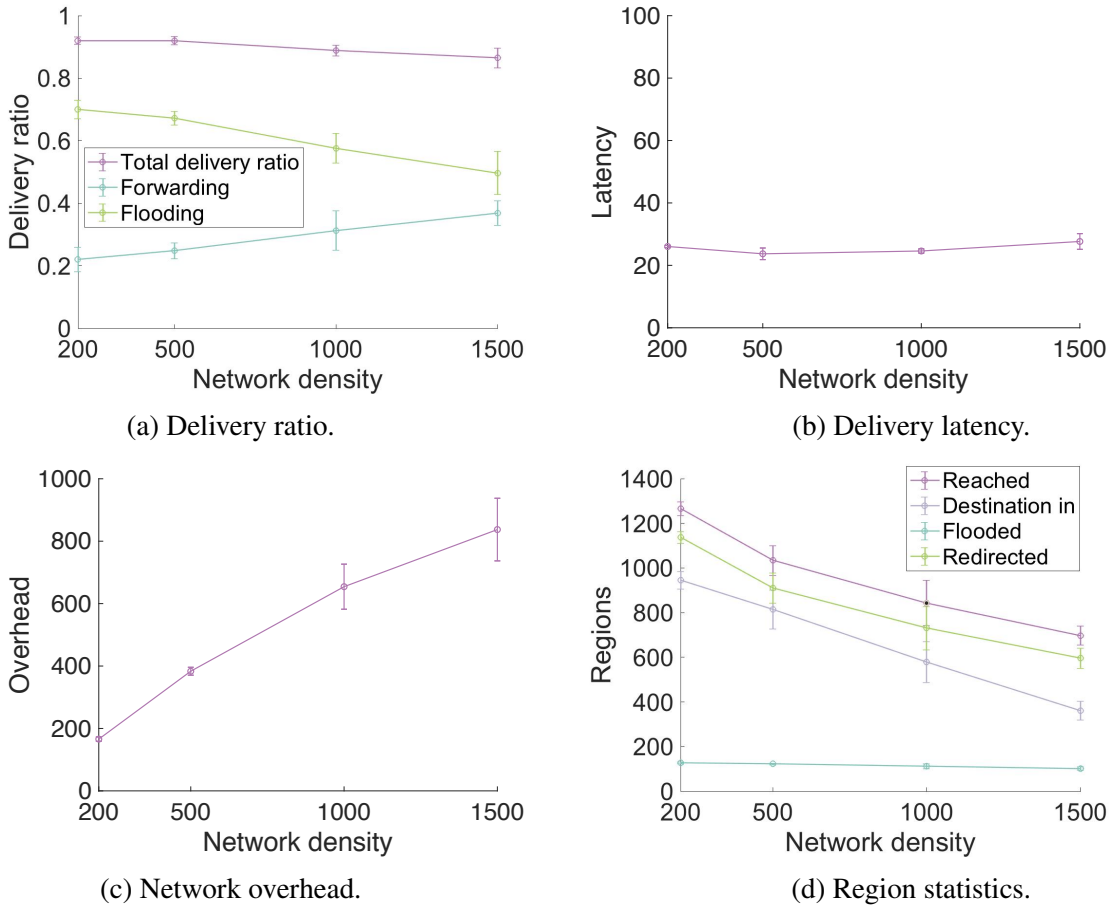


(a) Delivery ratio. (b) Delivery latency. (c) Network overhead.

Figure 4.44 Performance analysis - varying simulated error (mobility - 1 event).

Next, we evaluate GeoHawk's performance with the proposed collaborative localisation mechanism. We have used the settings discussed in Section 4.5.2, including a message TTL value of $250s$. In Section 4.5.2 we saw that in such a network setup GeoHawk performs very well, trading network latency for a very high delivery ratio. Smaller values of the message TTL resulted in lower delivery ratio values but lower latency, too. Figure 4.45

illustrates the delivery ratio when varying the density of the network. We observe that the performance of GeoHawk drops as the network density increases. For all densities, the delivery ratio is comparable to the one measured for the scenario where accurate location was always available. For all different densities, the majority of messages get to their destinations by flooding. A smaller percentage of messages get to their destination during GeoHawk's first phase. The delivery latency is also comparable with the scenario where location estimation was accurate and is generally stable with the density of the network. The measured network overhead increases with network density. However, it is slightly higher compared to the accurate location scenario. This is because errors affect nodes' confidence, which is represented by larger radii of regions. Figure 4.45d depicts statistics about the regions reached throughout the simulations. We observe that as discussed above flooding slightly decreases as the density of the network increases. As the network gets denser, more routing information is present, therefore messages are not redirected as much as in sparser simulated deployments.



(a) Delivery ratio.

(b) Delivery latency.

(c) Network overhead.

(d) Region statistics.

Figure 4.45 Performance comparison - collaborative localisation (mobility - 1 events, message TTL 250s).

Figure 4.46 illustrates the average location error throughout the whole simulation. We observe that as the network gets denser, the error increases very slowly with time, in contrast to the sparse simulated deployments. The key takeaway message here is that the proposed collaborative localisation system provides accurate enough location estimates to mobile devices for GeoHawk to operate at high performance levels in dense networks. This is because mobility is generally limited and devices are at times stationary. Stationary devices with initially accurate estimates of their location act as beacons for passing by devices, correcting PDR errors. In turn, these devices may stop moving later on, becoming beacons for other devices. In a real, large-scale network deployment devices could periodically (but infrequently) get access to accurate location estimates (e.g. using

GPS when being outdoors or some indoors based localisation mechanism) to correct their errors. Subsequently, such corrections would 'propagate' in the network through our collaborative localisation mechanisms as moving devices pass by stationary ones. Our approach performs much better compared to the simulated model discussed above where errors could not be smoothed by particle filters and corrected through collaborative localisation. We expect that any approach that would accumulate errors (e.g. a naive PDR localisation mechanism) would result in unacceptably large errors that would render GeoHawk and any other routing protocol that requires knowing devices' locations useless.

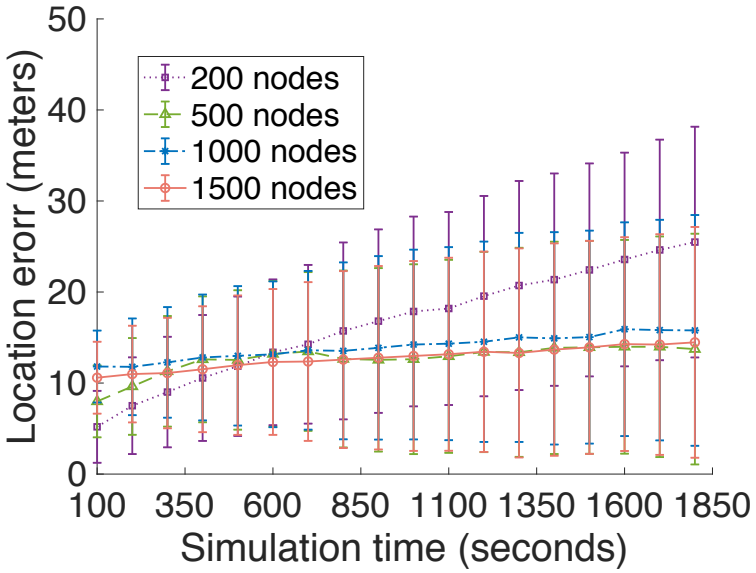


Figure 4.46 Average location error.

Chapter 5

Conclusion and Future Directions

In this thesis we discussed about the importance and the applications of the position-based protocols in opportunistic networks (Chapter 1). The state of the art in opportunistic network (i.e. Epidemic,PRoPHET, WSR) along with the collaborative localisation technique has been introduced and discussed in Chapter 2. We also extensively presented the major differences between the proposed work in topology- and position-based protocols designed for opportunistic networks. GeoHawk have been presented as an efficient and flexible protocol for geographical unicast in opportunistic networks. We highlighted significant challenges in position-based protocols in the context of dense, slow mobility opportunistic networks and described how GeoHawk deals with these challenges, overcoming limitations of existing approaches.

We also, comprehensively described the design of GeoHawk routing protocol in Chapter 3, proposing an improved approach that uses the context of users to aid routing decisions and collaboratively localising nodes in the network; without sharing the information with other users and compromising their privacy. Designing new ways of utilising more user related information without a potential data leak could be a subject to the future work.

The results presented in Chapter 4 investigates the behaviour of GeoHawk in detail and the impact of buffer capacity, message lifetime and transmission range for all state of the art protocols within crowded scenarios. We discussed about the importance of reserving network resources in such settings and trade-offs.

GeoHawk managed to deliver up to $\sim 80\%$ of the recipients, proving to be significantly better in comparison to existing approaches in several different dense scenarios. Achieving higher performance may be possible with some compromises; e.g. a solution which has better performance in crowded networks may affect other factors (e.g. network overhead and latency). The mechanism of aggregating announced information (*presented* in Chapter 3) enabled the maintenance of networks' soft state, which heavily contributed to this favourable outcome(i.e. better performance).

However, we believe that GeoHawk design is also open to further improvements that can restrict network overhead and increase or maintain delivery rates; e.g. dynamically allocating parameters values such as adjusting temporal decays' pace according the users' average speed and elevating ticketing mechanism by eliminating replication towards previously selected directions can lower network overhead.

In addition our findings shows that GPSR isn't the best way to distribute information within a dense network, as the distance metric in GPSR restricts the number of nodes that are able to receive an advert along a selected direction; in most cases, nodes hold overlapping regions mapped to a specific part of the network instead of spreading the entire area. Thus, we plan to investigate various approaches that enables our protocol to appropriately distribute information, where all nodes along a selected direction are able to receive its' designated announcement.

Moreover, in the research beyond this work, it would be interesting to look towards the special scenarios for particular use cases (e.g. realistically larger denser settings with longer durations and timely events) to study how close it is to the optimal performance. We also look forward to evaluate collaborative system design in 2.3 and report the protocols' behaviour with different localisation errors. However, the ONE simulator comes with many shortcomings and is not tailored for congested environments; the ONE is a single threaded program, which cannot exchange messages in a bidirectional manner, exchanging small sized messages leads to un-utilized throughput and its only able to process node by node. These issues will likely limit the range of experimentations and evaluation choices, jeopardizing our future progress. Thus, we aim to reconsider the possibility of reconstructing the ONE simulator using OMNeT++.

Denial-of-Service attacks on the other hand may affect GeoHawk (same as all the other routing protocols in opportunistic networks), as there are possibilities of a node intentionally reporting wrong copy-tickets or wrong movement directions in the network. Exploring such issue, was beyond the scope of the work in this thesis. Research on solutions about DoS attacks in geographical routing protocols is a topic for future work.

Bibliography

[1] Interplanetary networking special interest group (ipnsig). http://www.ipnsig.org/.

[2] Utku Günay Acer, Shivkumar Kalyanaraman, and Alhussein A Abouzeid. Weak state routing for large-scale dynamic networks. *IEEE/ACM Transactions on Networking*, 18(5):1450–1463, 2010.

[3] Michael D Addlesee, Alan Jones, Finnbar Livesey, and Ferdinando Samaria. The orl active floor [sensor system]. *IEEE Personal Communications*, 4(5):35–41, 1997.

[4] Wael Saleh Al-Halabi and Ibtesam Mohammed Abu-Sulyman. Analysis of activities in large spaces: Western piazza of the holy mosque as a sample research. *International Journal of Applied Physics and Mathematics*, 4(2):144, 2014.

[5] Yasir S Ali, Basim Zafar, and Mohammed Simsim. Estimation of density levels in the holy mosque from a network of cameras. In *Traffic and Granular Flow'15*, pages 27–34. Springer, 2016.

[6] Paramvir Bahl and Venkata N Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775–784. Ieee, 2000.

[7] Boaz Ben-Moshe, Paz Carmi, Moti Shani, and Nir Shvalb. Efficient model for indoor radio paths computation. *Simulation Modelling Practice and Theory*, 29:163–172, 2012.

[8] Nirupama Bulusu, John Heidemann, and Deborah Estrin. Gps-less low-cost outdoor localization for very small devices. *IEEE personal communications*, 7(5):28–34, 2000.

[9] N Buniyamin, WAJ Wan Ngah, N Sariff, and Z Mohamad. A simple local path planning algorithm for autonomous mobile robots. *International journal of systems applications, Engineering & development*, 5(2):151–159, 2011.

[10] Yue Cao and Zhili Sun. Routing in delay/disruption tolerant networks: A taxonomy, survey and challenges. *IEEE Communications surveys & tutorials*, 15(2):654–677, 2013.

[11] Yue Cao and Zhili Sun. Routing in delay/disruption tolerant networks: A taxonomy, survey and challenges. *IEEE Communications surveys & tutorials*, 15(2):654–677, 2013.

[12] Yue Cao, Zhili Sun, Haitham Cruickshank, and Fang Yao. Approach-and-roam (aar): a geographic routing scheme for delay/disruption tolerant networks. *IEEE transactions on Vehicular Technology*, 63(1):266–281, 2014.

[13] Yue Cao, Zhili Sun, Ning Wang, Haitham Cruickshank, and Naveed Ahmad. A reliable and efficient geographic routing scheme for delay/disruption tolerant networks. *IEEE Wireless Communications Letters*, 2(6):603–606, 2013.

[14] Yue Cao, Zhili Sun, Ning Wang, Maryam Riaz, Haitham Cruickshank, and Xiulei Liu. Geographic-based spray-and-relay (gsar): an efficient routing scheme for dtns. *IEEE Transactions on Vehicular Technology*, 64(4):1548–1564, 2015.

[15] Srdjan Čapkun, Maher Hamdi, and Jean-Pierre Hubaux. Gps-free positioning in mobile ad hoc networks. *Cluster Computing*, 5(2):157–167, 2002.

[16] Vinton Cerf, Scott Burleigh, Adrian Hooke, Leigh Torgerson, Robert Durst, Keith Scott, Kevin Fall, and Howard Weiss. Delay-tolerant networking architecture. Technical report, 2007.

[17] Dan Chalmers. *Sensing and systems in pervasive computing: Engineering context aware systems*. Springer Science & Business Media, 2011.

[18] Li-wei Chan, Ji-rung Chiang, Yi-chao Chen, Chia-nan Ke, Jane Hsu, and Hao-hua Chu. Collaborative localization: Enhancing wifi-based position estimation with neighborhood links in clusters. In *International Conference on Pervasive Computing*, pages 50–66. Springer, 2006.

[19] Ling-Jyh Chen, Chen-Hung Yu, Tony Sun, Yung-Chih Chen, and Hao-hua Chu. A hybrid routing approach for opportunistic networks. In *Proceedings of the 2006 SIGCOMM workshop on Challenged networks*, pages 213–220. ACM, 2006.

[20] Marco Conti and Silvia Giordano. Multihop ad hoc networking: The theory. *IEEE Communications Magazine*, 45(4), 2007.

[21] Marco Conti and Mohan Kumar. Opportunities in opportunistic computing. *Computer*, 43(1):42–50, 2010.

[22] James A Davis, Andrew H Fagg, and Brian N Levine. Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks. In *iswc*, page 141. IEEE, 2001.

[23] Ahmed El-Rabbany. *Introduction to GPS: the global positioning system*. Artech house, 2002.

[24] Jeffrey Erman and Kadangode K Ramakrishnan. Understanding the super-sized traffic of the super bowl. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 353–360. ACM, 2013.

[25] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34. ACM, 2003.

[26] Kevin Fall and Stephen Farrell. Dtn: an architectural retrospective. *IEEE Journal on Selected areas in communications*, 26(5), 2008.

[27] Emad Felemban, Adil A Sheikh, and Faisal Karim Shaikh. Mmapflow: a crowd-sourcing based approach for mapping mass pedestrian flow. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 311–317. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014.

[28] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous robots*, 8(3):325–344, 2000.

[29] V Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello. Bayesian filtering for location estimation. *IEEE pervasive computing*, 2(3):24–33, 2003.

[30] Atsushi Fujimura, Soon Y Oh, and Mario Gerla. Network coding vs. erasure coding: Reliable multicast in ad hoc networks. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1–7. IEEE, 2008.

[31] Matthias Grossglauser and D.N.C. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Transactions on Networking*, 10(4):477–486, 2002.

[32] Bo Han, Pan Hui, V.S. Anil Kumar, Madhav V. Marathe, Guanhong Pei, and Aravind Srinivasan. Cellular traffic offloading through opportunistic communications: A case study. In *ACM CHANTS*, 2010.

[33] Tian He, Chengdu Huang, Brian M Blum, John A Stankovic, and Tarek Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 81–95. ACM, 2003.

[34] Hong-Yu Huang, Pei-En Luo, Minglu Li, Da Li, Xu Li, Wei Shu, and Min-You Wu. Performance evaluation of suvnet with real-time traffic data. *IEEE Transactions on Vehicular Technology*, 56(6):3381–3396, 2007.

[35] ITV. Incredible aerial pictures show just how massive the glastonbury tent city is, 2015.

[36] Philippe Jacquet, Paul Muhlethaler, Thomas Clausen, Anis Laouiti, Amir Qayyum, and Laurent Viennot. Optimized link state routing protocol for ad hoc networks. In *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, pages 62–68. IEEE, 2001.

[37] P. Jassal, K. Yadav, A. Kumar, V. Naik, V. Narwal, and A. Singh. Unity: Collaborative downloading content using co-located socially connected peers. In *PERCOM Workshops*, 2013.

[38] Yunye Jin, Hong-Song Toh, Wee-Seng Soh, and Wai-Choong Wong. A robust dead-reckoning pedestrian tracking system with low cost sensors. In *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, pages 222–230. IEEE, 2011.

[39] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile computing*, pages 153–181. Springer, 1996.

[40] Evan PC Jones, Lily Li, Jakub K Schmidtke, and Paul AS Ward. Practical routing in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 6(8):943–959, 2007.

[41] Kamol Kaemarungsi and Prashant Krishnamurthy. Modeling of indoor positioning systems based on location fingerprinting. In *INFOCOM 2004. Twenty-third Annual-Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1012–1022. IEEE, 2004.

[42] George Kampis, Jan W Kantelhardt, Kamil Kloch, and Paul Lukowicz. Analytical and simulation models for collaborative localization. *Journal of Computational Science*, 6:1–10, 2015.

[43] George Kampis and Paul Lukowicz. Collaborative localization as a paradigm for incremental knowledge fusion. In *Cognitive Infocommunications (CogInfoCom), 2014 5th IEEE Conference on*, pages 327–331. IEEE, 2014.

[44] Hyunwoo Kang and Dongkyun Kim. Vector routing for delay tolerant networks. In *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, pages 1–5. IEEE, 2008.

[45] Hyunwoo Kang and Dongkyun Kim. Hvr: History-based vector routing for delay tolerant networks. In *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on*, pages 1–6. IEEE, 2009.

[46] Holger Karl and Andreas Willig. *Protocols and architectures for wireless sensor networks*. John Wiley & Sons, 2007.

[47] Brad Karp and Hsiang-Tsung Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254. ACM, 2000.

[48] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The one simulator for dtn protocol evaluation. In *Proceedings of the 2nd international conference on simulation tools and techniques*, page 55. ICST (Institute for Computer Sciences, Social-Informatics and . . . , 2009.

[49] Maurice J Khabbaz, Chadi M Assi, and Wissam F Fawaz. Disruption-tolerant networking: A comprehensive survey on recent developments and persisting challenges. *IEEE Communications Surveys & Tutorials*, 14(2):607–640, 2012.

[50] Kamil Kloch, Paul Lukowicz, and Carl Fischer. Collaborative pdr localisation with mobile phones. In *Wearable Computers (ISWC), 2011 15th Annual International Symposium on*, pages 37–40. IEEE, 2011.

[51] Kamil Kloch, Paul Lukowicz, and Carl Fischer. Collaborative pdr localisation with mobile phones. In *Wearable Computers (ISWC), 2011 15th Annual International Symposium on*, pages 37–40. IEEE, 2011.

[52] John Krumm. *Ubiquitous computer Fundamentals*. CRC Press, 2016.

[53] Erik Kuiper and Simin Nadjm-Tehrani. Geographical routing with location service in intermittently connected manets. *IEEE Transactions on Vehicular Technology*, 60(2):592–604, 2011.

[54] BDS Lakmali, WHMP Wijesinghe, KUM De Silva, KG Liyanagama, and SAD Dias. Design, implementation & testing of positioning techniques in mobile networks. In *Information and Automation for Sustainability, 2007. ICIAFS 2007. Third International Conference on*, pages 94–99. IEEE, 2007.

[55] Anthony LaMarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Timothy Sohn, James Howard, Jeff Hughes, Fred Potter, et al. Place lab: Device positioning using radio beacons in the wild. In *International Conference on Pervasive Computing*, pages 116–133. Springer, 2005.

[56] Ilias Leontiadis and Cecilia Mascolo. Geopps: Geographical opportunistic routing for vehicular networks. In *2007 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–6. Ieee, 2007.

[57] Wenzhong Li, Yuefei Hu, Xiaoming Fu, Sanglu Lu, and Daoxu Chen. Cooperative positioning and tracking in disruption tolerant networks. *IEEE Transactions on Parallel and Distributed Systems*, 26(2):382–391, 2015.

[58] Xu Li, Wei Shu, Minglu Li, Hongyu Huang, and Min-You Wu. Dtn routing in vehicular sensor networks. In *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference*, pages 1–5. IEEE, 2008.

[59] Yunfeng Lin, Ben Liang, and Baochun Li. Performance modeling of network coding in epidemic routing. In *Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*, pages 67–74. ACM, 2007.

[60] Anders Lindgren, Avri Doria, and Olov Schelen. Probabilistic routing in intermittently connected networks. In *Service assurance with partial and intermittent resources*, pages 239–254. Springer, 2004.

[61] Jó Agila Bitsch Link, Daniel Schmitz, and Klaus Wehrle. Geodtn: Geographic routing in disruption tolerant networks. In *2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*, pages 1–5. IEEE, 2011.

[62] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1067–1080, 2007.

[63] Haiyun Luo, Ramachandran Ramjee, Prasun Sinha, Li Erran Li, and Songwu Lu. Ucan: a unified cellular and ad-hoc network architecture. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 353–367. ACM, 2003.

[64] Ren C Luo, Ogst Chen, and Shi H Pan. Mobile user localization in wireless sensor network using grey prediction method. In *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*, pages 6–pp. IEEE, 2005.

[65] Daily Mail. Tents, tiaras and a few tears on the streets of london as a million turn out for wills and kate, 2011.

[66] Syed Haani Masood, Syed Ali Raza, and Mark Coates. Content distribution strategies in opportunistic networks. *CoRR*, abs/1308.0786, 2013.

[67] Martin Mauve, Jorg Widmer, and Hannes Hartenstein. A survey on position-based routing in mobile ad hoc networks. *IEEE network*, 15(6):30–39, 2001.

[68] Syed Atif Mehdi and Karsten Berns. Ordering of robotic navigational tasks in home environment. In *FIRA RoboWorld Congress*, pages 242–249. Springer, 2010.

[69] Ziad A Memish, Alimuddin Zumla, Rafat F Alhakeem, Abdullah Assiri, Abdulhafeez Turkestani, Khalid D Al Harby, Mohamed Alyemni, Khalid Dhafar, Philippe Gautret, Maurizio Barbeschi, et al. Hajj: infectious disease surveillance and control. *The Lancet*, 383(9934):2073–2082, 2014.

[70] Luis Mengual, Oscar Marbán, and Santiago Eibe. Clustering-based location in wireless networks. *Expert Systems with Applications*, 37(9):6165–6175, 2010.

[71] RT Question More. Over 1mn s. koreans protest embattled president park, opposition seeks impeachment (photos, videos), 2016.

[72] Mirco Musolesi, Stephen Hailes, and Cecilia Mascolo. Adaptive routing for intermittently connected mobile ad hoc networks. In *World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a*, pages 183–189. IEEE, 2005.

[73] Delphine Nain, Noshirwan Petigara, and Hari Balakrishnan. Integrated routing and storage for messaging applications in mobile ad hoc networks. *Mobile Networks and Applications*, 9(6):595–604, 2004.

[74] Asis Nasipuri and Kai Li. A directionality based location discovery scheme for wireless sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 105–111. ACM, 2002.

[75] BBC News. Hong kong: Democracy rally 'draws 510,000 protesters', 2014.

[76] BBC News. Why india's huge 'spiritual' festival has run into trouble, 2016.

[77] Dragos Niculescu and Badri Nath. Ad hoc positioning system (aps). In *Global Telecommunications Conference, 2001. GLOBECOM'01. IEEE*, volume 5, pages 2926–2931. IEEE, 2001.

[78] Salim Outemzabet and Chahe Nerguizian. Accuracy enhancement of an indoor ann-based fingerprinting location system using particle filtering and a low-cost sensor. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 2750–2754. IEEE, 2008.

[79] George Parisis, Vasilis Sourlas, Konstantinos V Katsaros, Wei Koong Chai, George Pavlou, and Ian Wakeman. Efficient content delivery through fountain coding in opportunistic information-centric networks. *Computer Communications*, 100:118–128, 2017.

[80] Shwetak N Patel, Matthew S Reynolds, and Gregory D Abowd. Detecting human movement by differential air pressure sensing in hvac system ductwork: An exploration in infrastructure mediated sensing. In *International Conference on Pervasive Computing*, pages 1–18. Springer, 2008.

[81] Shwetak N Patel, Khai N Truong, and Gregory D Abowd. Powerline positioning: A practical sub-room-level indoor location system for domestic use. In *International Conference on Ubiquitous Computing*, pages 441–458. Springer, 2006.

[82] Luciana Pelusi, Andrea Passarella, and Marco Conti. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *IEEE communications Magazine*, 44(11):134–141, 2006.

[83] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. Ad hoc on-demand distance vector (aodv) routing. Technical report, 2003.

[84] Kevin Peters, Abdul Jabbar, Egemen K Cetinkaya, and James PG Sterbenz. A geographical routing protocol for highly-dynamic aeronautical networks. In *2011 IEEE Wireless Communications and Networking Conference*, pages 492–497. IEEE, 2011.

[85] Nissanka B Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 32–43. ACM, 2000.

[86] Amanda Prorok, Alexander Bahr, and Alcherio Martinoli. Low-cost collaborative localization for large-scale multi-robot systems. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4236–4241. Ieee, 2012.

[87] Ioannis Psaras, Vasilis Sourlas, Denis Shtefan, Sergi Rene, Mayutan Arumaithurai, Dirk Kutscher, and George Pavlou. On the feasibility of a user-operated mobile content distribution network. In *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2017 IEEE 18th International Symposium on*, pages 1–9. IEEE, 2017.

[88] Aydin Rajaei, Dan Chalmers, Ian Wakeman, and George Parisis. Gsaf: Efficient and flexible geocasting for opportunistic networks. In *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–9. IEEE, 2016.

[89] Cliff Randell, Chris Djiallis, and Henk Muller. Personal position measurement using dead reckoning. In *null*, page 166. IEEE, 2003.

[90] Ioannis M Rekleitis, Gregory Dudek, and Evangelos E Milios. Multi-robot cooperative localization: a study of trade-offs between efficiency and accuracy. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2690–2695. IEEE, 2002.

[91] Valérie Renaudin, Melania Susi, and Gérard Lachapelle. Step length estimation using handheld inertial sensors. *Sensors*, 12(7):8507–8525, 2012.

[92] Jihoon Ryoo, Hyunjun Choi, and Hwangnam Kim. Sequential monte carlo filtering for location estimation in indoor wireless environments. In *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, pages 1–2. IEEE, 2010.

[93] Siamak Sarmady, Fazilah Haron, and Abdullah Zawawi Talib. A cellular automata model for circular movements of pedestrians during tawaf. *Simulation Modelling Practice and Theory*, 19(3):969–985, 2011.

[94] Andreas Savvides, Chih-Chieh Han, and Mani B Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 166–179. ACM, 2001.

[95] Andreas Schmidt-Dannert and Seminar Master Module SNET. Positioning technologies and mechanisms for mobile devices. In *Seminar Master Module SNET2 TU-Berlin*, 2010.

[96] Andreas Schmidt-Dannert and Seminar Master Module SNET. Positioning technologies and mechanisms for mobile devices. In *Seminar Master Module SNET2 TU-Berlin*, 2010.

[97] K Scott and S Burleigh. Bundle protocol specification (rfc 5050). *The IETF Trust*, 2007.

[98] Muhammad Zubair Shafiq, Lusheng Ji, Alex X Liu, Jeffrey Pang, Shobha Venkataraman, and Jia Wang. A first look at cellular network performance during crowded events. In *ACM SIGMETRICS Performance Evaluation Review*, volume 41, pages 17–28. ACM, 2013.

[99] Tara Small and Zygmunt J Haas. The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way). In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 233–244. ACM, 2003.

[100] CC Sobin, Vaskar Raychoudhury, Gustavo Marfia, and Ankita Singla. A survey of routing and data dissemination in delay tolerant networks. *Journal of Network and Computer Applications*, 67:128–146, 2016.

[101] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259. ACM, 2005.

[102] Ulrich Steinhoff and Bernt Schiele. Dead reckoning from the pocket-an experimental study. In *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*, pages 162–170. IEEE, 2010.

[103] Ken Sugawara, Toshiya Kazama, and Toshinori Watanabe. Foraging behavior of interacting robots with virtual pheromone. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 3074–3079. IEEE, 2004.

[104] Fabrice Tchakountio and Ram Ramanathan. Tracking highly mobile endpoints. In *Proceedings of the 4th ACM international workshop on Wireless mobile multimedia*, pages 83–94. ACM, 2001.

[105] Qinglin Tian, Zoran Salcic, I Kevin, Kai Wang, and Yun Pan. An enhanced pedestrian dead reckoning approach for pedestrian tracking using smartphones. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on*, pages 1–6. IEEE, 2015.

[106] Qinglin Tian, Zoran Salcic, I Kevin, Kai Wang, and Yun Pan. An enhanced pedestrian dead reckoning approach for pedestrian tracking using smartphones. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on*, pages 1–6. IEEE, 2015.

[107] The New York Times. Crowd scientists say women's march in washington had 3 times as many people as trump's inauguration, 2017.

[108] Sacha Trifunovic, Bernhard Distl, Dominik Schatzmann, and Franck Legendre. WiFi-Opp: Ad-hoc-less opportunistic networking. In *Proc. of ACM CHANTS*, 2011.

[109] Cenk Tunasar. Analytics driven master planning for mecca: Increasing the capacity while maintaining the spiritual context of hajj pilgrimage. In *Simulation Conference (WSC), 2013 Winter*, pages 241–251. IEEE, 2013.

[110] Okan Turkes, Hans Scholten, and Paul Havinga. Roro-lt: social routing with next-place prediction from self-assessment of spatiotemporal routines. In *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, pages 201–208. IEEE, 2013.

[111] Shweta Ubnare, Balram Yadav, and Bharti Chourasiya. Survey of localization techniques based on mobile beacon in wireless sensor network. *International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering*, 5(5), 2017.

[112] Amin Vahdat, David Becker, et al. Epidemic routing for partially connected ad hoc networks. 2000.

[113] Badri N Vellambi, Ramanan Subramanian, Faramarz Fekri, and Mostafa Ammar. Reliable and efficient message delivery in delay tolerant networks using rateless codes. In *Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*, pages 91–98. ACM, 2007.

[114] Vladimir Vukadinović and Gunnar Karlsson. Spectral efficiency of mobility-assisted podcasting in cellular networks. In *MobiOpp*, 2010.

[115] Ian Wakeman, Stephen Naicken, Jon Rimmer, Dan Chalmers, and Ciaran Fisher. The fans united will always be connected: Building a practical DTN in a football stadium. In *ADHOCNETS*, 2013.

[116] Hui Wang, Henning Lenz, Andrei Szabo, Joachim Bamberger, and Uwe D Hanebeck. Wlan-based pedestrian tracking using particle filters and low-cost mems sensors. In

Positioning, Navigation and Communication, 2007. WPNC'07. 4th Workshop on, pages 1–7. IEEE, 2007.

[117] Jing Wang, Ratan K Ghosh, and Sajal K Das. A survey on sensor localization. *Journal of Control Theory and Applications*, 8(1):2–11, 2010.

[118] Tong Wang, Yue Cao, Yongzhe Zhou, and Pengcheng Li. A survey on geographic routing protocols in delay/disruption tolerant networks. *International Journal of Distributed Sensor Networks*, 12(2):3174670, 2016.

[119] Yong Wang, Sushant Jain, Margaret Martonosi, and Kevin Fall. Erasure-coding based routing for opportunistic networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 229–236. ACM, 2005.

[120] Yufeng Wang, Hongxu Zhang, Chiu C. Tan, and Xiaojiang D. Sheng. WiGroup: A lightweight cellular-assisted device-to-device network formation framework. In *UIC*, 2015.

[121] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, 1992.

[122] Welovedc. 9/12 rally crowd estimates: Two million, 2009.

[123] Wikipedia. Great mosque of mecca, 2017. [Online; accessed 01-July-2017].

[124] Mohammad Yamin and Moteb Ayesh Albugami. An architecture for improving hajj management. In *ICISO*, pages 187–196, 2014.

[125] Pei Zhang and Margaret Martonosi. Locale: Collaborative localization estimation for sparse mobile sensor networks. In *Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on*, pages 195–206. IEEE, 2008.

[126] Zhensheng Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges. *IEEE Communications Surveys & Tutorials*, 8(1):24–37, 2006.

[127] Zhensheng Zhang and Qian Zhang. Delay/disruption tolerant mobile ad hoc networks: latest developments. *Wireless Communications and Mobile Computing*, 7(10):1219–1232, 2007.

[128] Jian Zhu and Gregory D Durgin. Indoor/outdoor location of cellular handsets based on received signal strength. *Electronics letters*, 41(1):24–26, 2005.

[129] Jinqi Zhu, Jiannong Cao, Ming Liu, Yuan Zheng, Haigang Gong, and Guihai Chen. A mobility prediction-based adaptive data gathering protocol for delay tolerant mobile sensor network. In *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference*, pages 1–5. IEEE, 2008.