Ain't Graph Bad: Empathic Information Dissemination for Autonomous Peer-to-Peer Clustering

Utilising Empathic Clustering to Reduce Disseminated Information

Thor M. K. Høgås (thor@roht.no)

Thesis submitted for the degree of
Master in Informatics: Programming and System Architecture
60 credits

Department of Informatics
Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2020

Ain’t Graph Bad: Empathic Information Dissemination for Autonomous Peer-to-Peer Clustering

Utilising Empathic Clustering to Reduce Disseminated Information

Thor M. K. Høgås (thor@roht.no)

Abstract

As consumer Wi-Fi density continues to expand in line with the increase of wireless devices in homes and urban settings, improving connectivity requires finding more innovative solutions to the resulting challenges of distributed sensing within dynamic radio resource management. While research within distributed and cooperative sensing has attempted to solve issues related to radio spectrum scarcity, these solutions either have their own associated challenges in application or target other environments. In order to solve these issues in residential settings we must have sufficiently efficient and optimised communication and clustering mechanisms in order to perform distributed channel allocation.

In this thesis, we utilise a distributed and autonomous minmax clustering algorithm together with a push-based information dissemination approach. This combination has only previously existed in an ongoing doctorate thesis.

For this clustering algorithm, we define two new pruning mechanisms, projective and sympathetic, before using OMNeT++ to create different, discrete event-based simulation models. We then compare these simulations and their results in order to evaluate their performance and suitability for use in residential distributed channel allocation.

We find that a flooding approach with both pruning mechanisms is affordable for cluster sizes of 100 nodes, and establish that these can be used in a distributed solution that maintains a low degree of complexity. One of the pruning mechanisms successfully reduces the amount of information disseminated for high-density topologies with a connectedness of $p > 15$. Combining self-pruning and sympathetic pruning reduced total traffic by approx. 50%, subsequently reducing the total payload transferred from 1.4 GB to 0.7 GB. This allows for larger cooperative clusters that remain within the upper bounds of the other mechanism, with a caveat of a minor complexity trade-off.

Finally, we briefly examine possible future work for the project.

Preface

This thesis has been developed as part of the project *EmpathicWiFi*, and this thesis would not exist if not for the assistance, work, discussions and meetings in the project.

Many thanks to my fellow academic-in-crime Nicolas Harlem Eide for his support, flexible Windows solutions and pleasant murderous sunlight on rainy days.

To the project members of EmpathicWiFi. Thanks to Magnus Skjegstad for his on-point communication expertise, attention to detail and suggestions for improving the results. His knowledge in the area of gossiping has been a continual source of inspiration.

Thanks to Madeleine Rønning for her long video calls during a time in which physical meetings were no longer feasible. Without her, this thesis would not be possible, either in content or in quality.

Thanks to Torleiv Maseng, my supervisor and the father of GSM. His expertise and continual encouragement have been fundamental to the progress of my work.

Lastly, I would like to thank my partner-in-crime Jess the Viking: First of Her Name, Stranger in a Foreign Land, Uncompromising Remainer and Proud Gryffindor for the daily support, motivation and necessary ass-kicking. Thank you.

— Thor Høgås,
Summer 2020

Contents

List of Tables

List of Figures

Part I.

Introduction & Background

Introduction 1.

1.1. Background Motivation

Since their introduction in 2007, the IEEE 802.11 standards for wireless network communication have been a global success. New features have been continuously added, resulting in higher rates and better services. However, these standards use the ISM frequency bands, which are the most crowded radio bands in the world, and are used daily by almost everyone with a wireless device. Higher density access point (AP) installations and continued urbanisation put the radio spectrum availability under constant pressure. The number of wireless access networks are increasing, increasing the number of connected Wi-Fi devices as well as the traffic usage. According to Barnett, Jain, Andra *et al.* [1], half (51%) of the global IP traffic in 2022 will be driven by Wi-Fi. This would mean a final compound annual growth rate of 18% for Wi-Fi-only devices and 53% for mobile devices in the period 2017–2022.

Definition 1.1.1 (Access point) *Networking device that connects wireless devices on a Wireless Local Area Network (WLAN) to a wired Local Area Network (LAN), and to any connected Wide Area Network (WAN), typically the Internet.*

This lack of sufficient radio resources is particularly evident in higher density areas such as apartment blocks or multi-dwelling units (MDUs), where there are more base stations (BSs) or APs than there are available frequencies or channels [2]. In South Korea, and likely other locations with high-density Wi-Fi installations where people live in close proximity to one another, this interference between Wi-Fi networks is a significant problem [3]. The issue is further exacerbated by *overlapping* radio transmissions, which may have a greater overlap due to APs with a higher transmission effect than necessary.

This is often caused by consumers who want to increase their wireless network range and think that they can achieve this by increasing the AP transmission power, without considering the strength of their mobile devices' transmissions. The issue is made worse by manufacturers catering to consumers' requests, causing neighbouring users more harm at no increased benefit to the purchaser, or third-party firmware such as DD-WRT, OpenWRT, or Tomato, that surface these configuration parameters.

Nodes attempt to mitigate frequency scarcity by scanning the local area for neighbouring devices and activity to find the local optimal channel, preferably one that is unused or the least used.

If such a channel is not available, APs may traditionally utilise a more dynamic approach such as least-congested channel search (LCCS), where the AP moves if the traffic volume is greater than a calculated threshold [4]. As with other local BS-based approaches, LCCS is weak to hidden nodes [5]. APs must receive information from either clients or other APs to optimise frequency assignment for multiple users and WLANs.

While commercial APs limit their effect to meet the set regulatory limitations, and seldom limit it beyond these requirements, overlapping AP areas cannot be solved solely by further limitation. The signal coverage of areas is heavily affected by building layouts and construction materials, as well as other items and interference. Even for an AP that can optimally cover its area with its signal strength, its channel and air time is still likely to be shared with other networks, as its coverage will extend beyond the particular confines of the device owner's apartment or building area and into other inhabitants' radio space.

Ideally, APs aim and users of a WLANs wish to entirely avoid overlap on the same frequency. This is known as the frequency assignment problem (FAP), and it is a problem reducible to the graph colouring problem [6]. While there are 3–4 non-overlapping channels in the 2.4GHz band in IEEE 802.11g-2003 (also known as Wi-Fi 3) networks—depending on the local regulatory constraints—this is too low to avoid collisions entirely in an urban setting. Therefore, there are still advantages to be had from optimising and reducing the collisions [2].

The 5GHz band in 802.11g initially provides 9 non-overlapping 20 MHz channels, with bonding provided by 802.11n and additional channels provided by 802.11ac. These are expanded more in later standards, providing upwards of 19 non-overlapping 20MHz channels. However, the channels may be banded together in bulks of 40MHz, 80MHz and 160MHz. Bonding reduces the number to 10 non-overlapping 40MHz channels, or 5 non-overlapping 80MHz channels.

As devices support increased throughput and new radio spectrums, usage does not decrease but increases until it reaches a bottleneck, similar to the way that increasing capacity on transport road networks allows for increased usage until capacity is back to a premium. Furthermore, introducing mesh APs in IEEE 802.11s (mesh networking) and repeaters may also increase the value of the project as the radio spectrum becomes more crowded. Therefore, it is our belief that this project is equally applicable to and useful for networks utilising current technology, such as the 5GHz band, as well as new wireless standards still under development, such as the 6GHz band, which was approved by the FCC for Wi-Fi usage as recently as April 2020.

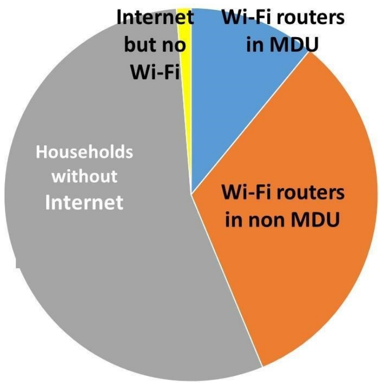


Figure 1.1.: Globally, around 2 billion Wi-Fi routers are located in apartments, MDUs and offices that interfere with each other due to a lack of bandwidth forcing them to share the same channel (blue piece).

The EmpathicWiFi Project

Network equipment manufacturers and Internet Service Providers (ISPs) have partial solutions for improving channel allocation [4], [7]. However, as we discuss later, these either do not cooperate across devices or do use a centralised architecture. The latest IEEE 802.11ax standard has spatial frequency reuse and adaptive power control built into the standard with LCCS, as introduced earlier, which selects the channel with the least traffic [4]. This is an unstable, individual-centric solution because it assumes that the neighbors who may have their performance reduced will do nothing to counteract it. As Zehl, Zubow and Wolisz [8] state in 'Practical Distributed Channel Assignment in Home Wi-Fi Networks':

> The efficient management of radio resources in today's home or residential Wi-Fi networks is still an open research question. Due to the chaotic and unplanned deployment of APs and the fact that all APs are managed individually by their owners, home Wi-Fi networks suffer from performance degradation due to contention and interference. [8]

Therefore, other solutions must be found that can work in tandem with the already existing infrastructure and installed base, and that will optimise throughput and decrease latency while targeting a heterogeneous base of devices.

The *EmpathicWiFi* project aims to reduce avoidable frequency overlap for clusters of access points [2]. It is both a research project and a commercial venture, which is working to implement a cooperative and distributed dynamic radio resource management system that is empathic in terms of how the cluster members cooperate.

We will demonstrate a completely new architecture for Wi-Fi systems which make them cooperate, rather than compete. Cooperation is done inside large clusters in which networks disturb each other. The finite cluster size make the system scale and make it possible for the microprocessor in each AP, to run our system. Our solution will result in better services and less interruptions for all Wi-Fi networks. It will enable more traffic to be handled by the networks.

Figure 1.2.: The EmpathicWiFi project objectives as described by Maseng [2]

To our knowledge, no practical method has yet been proposed that is suited for cities or countries without the use of tailored infrastructure servers that are operated by a third-party. Therefore, we have formulated the project objectives shown in fig. 1.2.

Empathic nodes establish backhaul connections to their neighbouring APs, enabling direct communication via the Internet [2], [9]. Subsequently, gathered wireless usage and interference information is shared with other devices, forming a distributed cluster. Current tests have shown an improvement potential of

Definition 1.1.2 (Backhaul) *A secondary communication channel established established in order to reduce costs of communication. In this context backhaul represents a communication between two nodes over WAN.*

upwards of 20% in terms of wireless transmission performance and packet loss [2].

Enabling Efficient Clustering and Optimisations

Definition 1.1.3 (Information dissemination) *The systematic process of ensuring those who need information in order to make informed decisions has the correct information made available to them within the given time constraints.*

The distributed clustering algorithm itself follows certain deterministic information dissemination paths in the course of converging towards results [10]. These paths and assumptions have been made to support the goals of this master's thesis, rather than to evaluate the ideal method of information dissemination in a cluster. Our motivation is to investigate how a model that utilises a form of empathic clustering can work, and furthermore how we can optimally acquire the information required for the clustering algorithm from our neighbouring APs, as well as provide it to them.

It is our belief that different approaches to information dissemination will yield different performance results based on metrics such as the total number of neighbour reports[1] before converging, the number of bytes transferred, and the time it takes to do this. In addition, we presume that adopting an evaluated empathic approach to dissemination in our network will support participants who are considering an actual implementation of the system in end-user consumer hardware. Different dissemination yields different behaviours when combined with other neighbouring nodes that are not empathic, known as *rogues*. For our purposes, rogues are inconsequential; interference between rogue nodes and empathic nodes are used in the process of calculating an optimal channel allocation even though they are not a partition member.

1: See section 5.1 for more on neighbour reports.

Rogue nodes are not included in the information dissemination simulations. Any rogue node observed does not contribute to spreading information, but its information will be spread.

This thesis aims to briefly introduce the reader to some of the concepts and approaches related to radio resource management, before delving into the practical challenges faced when the radio spectrum usage is shared from and throughout its surrounding access points. This establishes a background which we build upon to describe the alternative methods of information dissemination available.

1.2. Problem Statement

In order to further the implementation of *EmpathicWiFi*, we need to evaluate the different approaches to information dissemination for the purposes of clustering and, by extension, the channel allocation algorithm.

It is our goal to establish which of our approaches to information dissemination yield the best result within their limited scope, while fitting with the requirements of a hypothesised real-life implementation. We wish to see how information dissemination approaches coupled with the properties of the clustering algorithm allow for improvements in transmission costs. However, there are no hardware experiments in this thesis, and as we will delve deeper into in part II, a defined approach for our simulations and results.

Initial Questions

To begin with the author had the following questions:

Which form of information dissemination best suits EmpathicWiFi?
Is there one particular system that is more suitable than any others? Are there settings or environments in which EmpathicWiFi is used that make a system more suitable for certain dissemination approaches?

Is there a simple information dissemination approach that works better for a simulated variant of EmpathicWiFi?
Does a flood-based approach work better than one that utilises gossiping? How about an approach that incorperates the two?

Are there possible benefits to using distributed hash tables (DHTs) for information dissemination in EmpathicWifi?
Can DHTs be a suitable solution to both information dissemination and communication between nodes in an overlay network?

Can the clustering algorithm be exploited for gains in relatively simple information dissemination approaches?
In other words, can the clustering mechanism be used in the information dissemination approach to reduce the time it takes to complete the clustering, the information disseminated, or the amount of traffic?

How can simple information disseminantion work for EmpathicWiFi?
Closely related to previous issue: how might simple approaches work for disseminantion of clustering data for its requirements?

Can we simulate the clustering algorithm in a simulation framework?
Which simulation frameworks are most relevant for this type of simulation? Does it depend on what we want to simulate? Indeed, we assume that to be the case.

Research Questions

How effectively can flood-based information dissemination be used in real-world scenarios for EmpathicWiFi clustering per the metrics?
Considering the relevant metrics, what is the impact of flood-based information dissemination for EmpathicWiFi in a real-world implementation?

How effectively can a flood-delay based information dissemination be used in real-world scenarios for EmpathicWiFi clustering per the metrics?
Considering the relevant metrics, what is the impact of a modified flood-based information dissemination for EmpathicWiFi in a real-world implementation?

Can we prune information from forwarding when flooding that is not relevant for our own clustering?
If a given node does not use received information to reach its own clustering, can we ignore it?

Can we prune information from forwarding when it is not relevant to a neighbouring node's clustering?
If a neighbouring node does not need our received information to reach a clustering, can we ignore it on their behalf?

How much memory does it take us to reduce the number of messages in the different dissemination approaches?
Each method has its own associated mechanism and theoretical space complexity.

How much time complexity does it take us to reduce the number of messages in the different dissemination approaches?
Each method has its own associated mechanism and associated time complexity for performing the steps required to save on transmission costs. Are they worth it?

Hypotheses

1. The combination of backhaul communications and exploitation of minmax clustering properties in the information dissemination approach reduces the need to optimise for flooding.

   While the author believes that this is the case, to the point that even inefficient information dissemination is still sufficient to improve wireless connectivity for empathic nodes, we will not be able to confirm or reject this in this thesis. However, any reduction to the metrics we wish to optimise is a reduction in our optimisation needs, and may thus confirm or reject the hypothesis.

2. We can spread information using flood-based dissemination and perform clustering within the suggested interval of five minutes.

3. We can spread information using a flood-delay based dissemination and perform clustering within the suggested interval of five minutes.

4. We can spread information using a combined flood-delay and empathic dissemination approach that transmits messages in intervals to all empathic neighbours and performs clustering within the suggested interval of five minutes.

5. It is not disruptive to the network to use flood-based methods to spread information in networks if the cluster size is 100 or less.

6. It is possible to reduce the number of reports with a flood-based dissemination approach to that of a gossip-based dissemination approach.

7. We can exploit the minmax clustering to further reduce which pieces of information are candidates for dissemination, as well as to prune information.

8. We can exploit the minmax clustering to prune information in a way which is compatible with a flood-based dissemination approach.

9. We can reduce the number of transmitted reports by more than 5% within the 95% confidence interval by using empathic dissemination rather than the baseline flood-based dissemination approach.

10. An empathic dissemination approach will use more processing time than a baseline flood-based dissemination approach.

11. An empathic dissemination approach will require more memory than a baseline flood-based dissemination approach.

1.3. Outline

Part I starts with chapter 2, where we introduce background theory that helps us in our approach, including the field of radio resource management.

In chapter 3, we contextualise the small body of related work, the existing work on EmpathicWiFi, including an our understanding of minmax clustering in section 3.3.

Part II starts with chapter 4, where we start to define the strategy to our method, including our approach with regards to technology, measurements of interest, and the simulation framework we utilise. This includes our interpretation of requirements for EmpathicWiFi in section 4.2, and our primary choice of information dissemination in section 4.3, where we outline the information dissemination approach and its candidates.

In chapter 5, we build on the introduction to suggest specific empathic terminology and associated empathic mechanisms that we use in our work, along with rationalising its scale and behaviour in theory.

In chapter 6 we elaborate on all of the simulation models and their implementations, including common behaviour in section 6.2. section 6.3 contains our first approach wherein we elaborate blind flooding. We then examine the staged flooding approach in section 6.4, before implementing two models utilising our empathic mechanisms We then implement two models utilising the empathic mechanisms to lower transmission costs in section 6.5.

Part III is the final part, where we present the results with their trade-offs as simulated, and discussed, in chapter 7.

We conclude the thesis in chapter 8 by summarise interesting areas for future work and a timeline in section 8.1, and present our conclusion in section 8.2.

Background Theory | 2.

In this chapter we start by establishing the background material of the thesis. This includes an introduction to the basics of radio resource management.

2.1. The Basics of Radio Resource Management

Radio transmissions on the same frequency are considered shared mediums. As a physically shared medium, they are also a finite resource. Efficiently managing these resources is an ongoing research interest, and is referred to as radio resource management (RRM). The term functions as an umbrella for both small and large-scale applications [8], [9], its various approaches, and spectrum sensing [11]–[13]. When the frequency spectrum's capacity is not utilised to its full potential, spectrum holes occur, as a result of heterogeneous consumer devices limited to independently utilising sub-optimal heuristics and algorithms. Consequently, spectrum holes are radio areas left unused by primary a current primary user or users [14, p. 201].

Cooperation between APs in known to result in better spectrum utilisation [15], [16]. Simulations, analysis, plans and patents on how to do this, have been published [7], [17]–[21]. This work extends to improved measurement methods for allocating channels to APs with three high-performing algorithms [22], and experiments measuring traffic using a utility function with multiple APs connected to a server [23]. Central channel allocation schemes that include rogue APs have also been proposed, where channel utilisation is estimated based on received beacon signals [24].

Information from the clients are available from the AP when using IEEE 802.11k. In the Wi-5 EU project [25], Radio Resource Management was addressed and in particular AP selection and Vertical Handover using local controller was dealt with. On 1 April 2020 the EU project Smart-WiFi started using artificial intelligence to organize and optimize the World's Wi-Fi networks [26]. All the previous projects and articles agree on that coordination will result in better performance, but not how to do it on a large scale and certainly not how to do it without central coordination, even if 10 years ago, Chieochan, Hossain and Diamond [21] concluded that a decentralized system should be used for large networks to make them scale.

Cognitive radio The dynamic response to spectrum availability by collecting, evaluating and controlling the radio usage is referred to as *cognitive radio* [14], [27], [28].[1] Cognitive radio has two primary prioritisations: enabling highly reliable communications as well as efficiently utilising the radio spectrum, or the minimisation of *spectrum holes*. While the latter is important for frequency management, it is an objective of EmpathicWiFi, although it is a consequence of reducing interference.

1: Cognitive radio encompasses more than what is described here. However, while Mitola and Maguire [29] describe *cognitive radio* as something far more comprehensive, for all intents and purposes cognitive radio enables APs to perform radio resource management (RRM).

Haykin [14] states that the three main responsibilities of cognitive radio are: radio-scene analysis, channel state identification, and transmit-power control and dynamic spectrum management. These are performed individually and sub-optimally by consumer access points. In the majority of consumer devices, such actions are generally limited to detecting available channels and adapting their transmission and reception parameters accordingly [30].

The majority of work on RRM is not relevant for this thesis. The term and its sub-terms, however, are important for understanding the problem area and are therefore described in this section.

Centralised Radio Resource Management

In enterprise settings and in academia, solutions for central RRM already exist and have done so for a long time [31]. When available, it can provide the exact same optimisations as distributed RRM [2].

As part of cognitive radio, centralised RRM exhibits the same processing steps but with different architectural structures. RRM is discussed in different contexts; Haykin [14, sec. 7] differentiates between *centralised* and *decentralised* RRM, describing centralised as an “access point controlled transmit-power” and decentralised as “aided and controlled by primary transmitters themselves”. While this distinction is useful when discussing the details of cooperation among devices, this is not the focus of this thesis, which uses a slightly different distinction: the difference between centralised and decentralised lies in whether the APs *with only themselves and other APs* are able to perform RRM.[2]

2: An essential difference is whether users within each wireless network contribute to making the transmit-power decisions. These capabilities are already present, as modern APs gather and utilise the interference reports from connected clients to aid in selecting channels. These days, the clients’ reports are utilised regardless of who performs the optimisation.

In contrast to a dynamic and distributed channel selection, the centralised approach to radio resource management is fundamentally one or multiple controllers being solely responsible for parts of the cognitive radio steps [14]. In essence, a centralised

RRM will collect and analyse information across APs with a single point wherein it gathers all the data, processes it and subsequently updates the configuration of all devices.

The fundamental issue with centralised RRM for consumer usage is that it requires homogeneous devices or control structures. An ISP must supply both hardware and software to end-users, and only customers with both, as well as ISPs with a central controller, would be able to benefit from the RRM functionality. The fundamental issue with centralised RRM for consumer usage is that it requires homogeneous devices or control structures. An ISP must supply both hardware and software to end-users, in addition to providing a central controller service. Only those customers using both the provided hardware and software would be able to benefit from the RRM functionality. If generic versions of the hardware and software are available, there might be multiple competing solutions available on the market. Attempting to use these would require the ISP to have the aforementioned central controller, or in this case, multiple different controllers.[3]

3: Technological advances in interconnectivity and communication has not been thanks to a homogeneous architecture, but rather through competing solutions balancing offers of interoperability with heterogeneous solutions.

The fragmentation of the devices further detracts from the potential winnings of RRM.

Distributed Radio Resource Management

In Zehl, Zubow and Wolisz [8] ResFi is used to perform RRM, with channel and effect strength optimisations on multiple devices under homogeneous control. In a local setting within one housing unit, the devices communicate with each other and exchange channel information. When shared with other local devices, they are able to optimise frequency selection. Communication between nodes is based on each node within range of a probe request replying with its contact data.

Once safe backhaul communcition has been established across the APs, the load is reported by each individual access point, and as such, little to no switching is done between master and monitor mode. However, switching interfaces may result in a very brief period of downtime [9]. While initially the connected devices will be, and are described in Zehl, Zubow and Wolisz [8] as, neighbouring devices, ResFi's northbound interface can transmit messages to neighbours N-hops away by broadcasting it through the ResFi access point (AP).

For non-ResFi APs, each device reports a passive load as determined by ResFi AP. The load is used equally by the algorithm, regardless of whether it is directly or passively reported. It attempts to cooperate with adjoining rogue networks, although it is unable to adjust rogue APs' channel selection or effect [8].

In illustrating how ResFi can be used in a limited residential setting, Zehl, Zubow and Wolisz [8] refer to a simplified version of the weighted colouring channel assignment algorithm *Hsum*, although this has been adapted to deal with non-cooperative APs. With Hsum, the learning phase uses the load of all neighbouring APs. Communicating with the other devices is made possible by the initial probing and scanning.

By utilising a part of the dynamic frequency selection (DFS) functionality, ResFi is able to broadcast a Channel Switch Announcement (CSA-IE), which allows for near simultaneous channel changes by clients [8, sec. 11.9, 32]. With seamless switch-over APs can execute channel allocation algorithms on-demand, reacting to a significant increase of interference.

Cooperative and Empathic Radio Resource Management

When cooperating with neighbours and other nearby devices, devices may exhibit cooperative or empathic behaviour, rather than compete for available space on the radio spectrum. The latter of the terms is taken from the project description of Maseng [2]. When the cognitive radio process is used in an online, distributed and cooperative manner, we suggest that this is referred to as either cooperative or empathic RRM.

This definition of cooperative radio is arguably very similar to that of cognitive radio, in that it is an "intelligent wireless communication system that is aware of its surrounding environment" [14, p. 201]. However, contrary to a later discussion on multiuser cognitive radio in Haykin [14], cooperative and empathic RRM does not treat the process as a non-cooperative game, and does not optimise the solutions for rogue access points.

2.2. Network Information Dissemination

In this section we give an introduction to some of the elementary information dissemination definitions, mechanisms and strategies. As the field of information dissemination within computer science is broad we limit this to the areas relevant for our work or our discussions.

Dissemination Strategies

With a vast number of approaches to information dissemination here follows a brief summary of the different approaches or strategies that may be employed in dissemination mechanisms to varying extents.

While strategies such as *pull*, *push* or *synchronisation* are generally considered distinctly different, they are often combined to form more advanced variants. When combined they may overlap significantly in their functionality or in the mechanisms used. Distinguishing between them is done by evaluating what the node initiating the information communication does [33]. This includes situations where behaviour can be a function of time [34].[4]

4: An example includes starting with one strategy and over time changing to a different strategy that better suits information dissemination for the later stages of the application or use-case.

Push-based A push-based approach is one where the node initiating the transfer of information is the node also informing another participating node, and therefore the disseminating node in the exchange. Other nodes may not have requested the information in question, nor indeed have any need for the information. It may be old or irrelevant. While any approach seeks to minimise useless transmissions, to *push* information is particularly prone to redundant transfers.

Key to a push-based approach is that the node initiating the push must be sufficiently likely to transfer information wanted by the receiving nodes.

Fundamental examples of push-based approaches include broadcasting and flooding, definitions 2.2.1 and 2.2.4, but also gossiping in definition 2.2.7.

Pull-based The antithesis of a push-based approach is when the node initiating the transfer of information is requesting information from another node, rather than providing the information to the other node. The other node participating and responding to the request is the disseminating one. Thus the node *pulls* the information from others, provided that the node has the required information.

Key to a pull-based approach is that the node initiating the pull of information is in a position to know which nodes to contact. Additionally the initiating node is in a better position to choose when to pull the information, how to choose who to ask, and what information to request if applicable.

Fundamental examples of pull-based approaches include gossiping with pulling definition 2.2.8.

Synchronisation Subsequently, in a synchronisation-based approach the both the initiating and participating nodes are potentially disseminating and requesting. While there is still an initiating node, as well as one or multiple participating nodes, synchronisation is usually a type of set reconciliation problem. Participating nodes first exchange information to determine what they are missing, before transmitting the information between each other. This is a process that may occur in multiple rounds, such as seen in Skjegstad, Johnsen, Bloebaum *et al.* [35]. A simple approach is to first transfer the values of all missing keys, then transfer their respective values. Exchanging missing keys may be simple, but it may be a low-complexity approach to solving the problem with a trade-off for traffic.

Key to a synchronisation-approach is reducing the information exchanged to the bare minimum required to reconcile the differences.

A fundamental example of synchronisation in use is replication between databases in a primary scheme.[5] Pull- and push-based gossiping in definition 2.2.9 is another kind of synchronisation strategy.

5: A multi-primary scheme, previously referred to as a multi-master scheme, is one wherein multiple database nodes are available for writing, unlike a primary-replica where one node only works as a fallback node.

Dissemination Mechanisms

This section outlines a few common and elementary mechanisms or protocols for information disseminations.

Uncontrolled & Controlled Flooding

In the foundation of all flooding is broadcasting, commonly used in local networks, such as all Ethernet-based local networks on the link layer across the world.[6]

6: The Address Resolution Protocol is a link-layer protocol aiding with looking up the MAC addresses of devices on the same local network by using broadcasting.

Definition 2.2.1 (Broadcasting) *The act of sending a message to all one-hop neighbours in a network.*

At its core flooding is the application of repeated broadcasts beyond a single network.

Definition 2.2.2 (Uncontrolled flooding) *Indiscriminate forwarding of messages to all neighbours sans sender without any conditions.*

In most cases when flooding is discussed almost all types of flooding are *controlled* as in definition 2.2.3, even if they create broadcast storms. While controlled flooding exists in many shapes and forms, *uncontrolled flooding* is by its definition 2.2.2 very limited in behaviour. Controlled flood-based mechanisms have a cut-off point, one that can theoretically guarantee that the network will reach an end-state where the information is disseminated for everyone.

This thesis does not delve any deeper into uncontrolled flooding besides stating that it exists, and that any practical uses besides creating broadcast storms for the purposes of EmpathicWiFi are non-existent.

Definition 2.2.3 (Controlled flooding) *Forwarding of messages to all neighbours sans sender with checks such as time-to-live or pre-existing knowledge of the contents.*

Even blind flooding, which tracks the packets it forwards in order to prevent it from rebroadcasting multiple times, can contribute to a temporary broadcast storm in dense graphs with high amounts of data. Nonetheless it is an example of controlled flooding as it does not forward packets indiscriminantly.

Definition 2.2.4 (Blind flooding) *Retransmitting or forwarding a message via broadcasting to all neighbouring nodes that have not already received it from us [36, sec. 2], usually without verifying their need for the information beforehand.*

Definition 2.2.5 (Broadcast storm) *A network storm causing severe congestion and latency after a broadcast message results in multiple other nodes also broadcasting, thus contributing to an exponential growth in messages transferred.*

A particular weakness of blind flooding is that it may still create a temporary broadcast storm as it does not prevent overloading the network.

There are many optimised flooding mechanisms such as multipoint relay and neighbour aware adaptive power [36, s. 2]. These are just two out of a wide range of options that must be explored before implementing an information dissemination mechanism for EmpathicWiFi.

Optimised flooding mechanisms *prune* information by a combination of heuristics and algorithms in order to reduce transmission costs.

We bring up pruning again at a later point in chapter 5 and definition 5.3.1.

Gossip

An alternative to flooding, whether it is controlled and blind or utilising pruning, is to use a *gossip protocol* as defined in definition 2.2.6.

Definition 2.2.6 (Gossiping) *Contacting one or multiple randomly selected neighbouring nodes, in intervals, to receive or send information [37].*

However, gossip protocols in their many different variants are *a dime a dozen* [31], [34], [37]–[39].

The probabilistic choice of who to forward information from in gossip can be viewed as a probabilistic kind of pruning.

The primary forms of gossip protocols come in the dissemination strategies mentioned earlier. There is *push gossiping*, *pull gossiping*, the combination *push & and pull gossiping*, and any other variant that further specifies the behaviour of the gossip mechanism.

Definition 2.2.7 (Push gossiping) *Sending information or a message to a randomly selected neighbouring node [37].*

Definition 2.2.8 (Pull gossiping) *Requesting information or a message from a randomly selected neighbouring node [37].*

The gossip strategies as mentioned in definitions 2.2.7 to 2.2.9 may vary from protocol to protocol, such as making the push or pull mechanism eager or lazy, to name two [34].

Definition 2.2.9 (Push & pull gossiping) *Exchanging information or messages with a randomly selected neighbouring node if either node is informed [37].*

Gossip protocols also have different *end stages*. Messages may be forwarded on the basis of using a limited mode wherein a sequence field is used to limit the maximum number of transmissions a message receives. Similarly it may run in an unlimited mode without any limit to how many times a message may be retransmitted [34]. For the purposes of EmpathicWiFi a limited mode seems suitable as we have a fixed number of potential members from any given starting point: α, the maximum cluster size. However, it would not be negative for nodes to retransmit information to a previously unknown neighbour.

Perpetual gossiping is a type of gossiping where there is no end state to the gossiping besides ensuring that every node is informed. Its strength arguably lies in topologies with high degrees of mobility, where neighbours are both coming and leaving, such as Mobile Ad-Hoc Networks (MANETs). The original suggestion recommends using it such that the amount of information gossiped is kept under a constant threshold, while still ensuring the eventual dissemination of all the messages [39]. The lack of an end-state is to some extent problematic: once a node is certain of its cluster any additional information is unnecessary use of backhaul until the next interval for the channel allocation algorithm (CAA).

Staged Flooding

Similar to flooding, yet heavily inspired by gossiping, the *staged flooding* approach is a combination of two particular aspects.

The first part is a trade for latency with reduced commmunication by instituting a flood delay between receiving new *and useful* information and forwarding it to neighbours. This latency has been referred to in literature as the *latency-energy* trade-off when discussing flood delays in wireless transmissions, and the attempt at optimising for latency while simultaneously optimising for transmission costs are considered contradictory [40].

This is similar to how jitter-based techniques are used for wireless transmissions as is done for flooding within MANETs [41]. While jitter-based flooding is primarily used as a means to avoid collisions on a shared medium, our utilisation of jitter—or random flood-delay within an interval—attempts to reduce the simultaneous use of backhaul between neighbours, as well the total amount of data transferred as listen for longer before disseminating information to other neighbours. When a node utilising staged flooding receives a new message, it schedules its next forwarding and choses the jitter by scheduling its next dissemination update from within the shared interval by random choice.

This was originally inspired by is the notion of applying intervals and randomness as is done in gossiping [42]. However, it will include any new information inbetween now and then, both for forwarding and pruning. This builds on the assumption that more information allows us to make better decisions if receiving more information might change the outcome of what we forward to other nodes.

The second part is that while flooding usually acts on a message-to-message basis, the staged flooding works on the information contained within, of which we define and discuss later in section 5.1. We bundle up the information internally, and combine information received for dissemination. Two neighbouring APs that each send us the same copy of an information element allow us to take the latest version, or the first if they are identical, and only forward one of them.

As such we attempt to define staged flooding in definition 2.2.10. We wish to emphasise that we do not believe this combination to be original, but we must define it herein for us to use it later.

Definition 2.2.10 (Staged flooding) *A flood-based dissemination mechanism wherein nodes delay forwarding of new information by choosing randomly from within a given interval, similar to jitter-based flooding [41]. All new information received in the meanwhile, in the interval between first receipt and the scheduled forwarding, is included in the scheduled forwarding subject to pruning.*

Related Work | 3.

In this chapter we establish the set of related work that this thesis builds on and is related to.

Note that the set of specifically related work is small. This is in contrast to the amount of work within RRM and information dissemination. These fields have an abundance of literature, but these are not included here: rather, they are a part of the background theory in chapter 2. One exception to this is the work discussed in Distributed and Cooperative Sensing where we find that the descriptions, technologies and terms overlap with the goals of EmpathicWiFi.

We delve further into the work performed in one of the two master theses completed in 2017 and 2018, which are both related to this *EmpathicWiFi* project. The thesis by Nygårdshaug [43] suggests a design for information sharing over ResFi, and is more central to our work. Less related is Grønseth [44] as it looks at the channel assignment aspect, not information dissemination. We will not look closer at the latter, but there is ongoing work building on it by Illavalagan [45].

Once completed, we look closer at the *minmax* clustering algorithm for EmpathicWiFi in section 3.3. We look at our suggested mechanisms for minmax clustering later in chapter 5, after outlining our method approach in chapter 4.

3.1. Distributed and Cooperative Sensing

Yucek and Arslan [28] elaborate on several different areas within spectrum sensing. One of those areas is *cooperative sensing*, and a subset of it, namely *distributed sensing*. It is the intersection of sharing information, yet making their own decisions on which part of the spectrum they can use.

Empathic RRM falls under the definition umbrella of cooperative sensing, and subsequently so is EmpathicWiFi. However, to be more precise we can identify empathic RRM and EmpathicWiFi as part of *distributed sensing*, which is a specific subpart of cooperative sensing along with centralised sensing, as touched on in section 2.1. This is because cooperative sensing is the solution to problems that arise in spectrum due to noise uncertainty, fading, and shadowing, as cooperative sensing is proven to allow for much higher gains than only using local sensing [28, p. 124]. A device in a low-density area may utilise local sensing to improve connectivity, whereas several devices sharing their sensory information autonomously allow for optimal connectivity for entire clusters.

Challenges to cooperative sensing, especially distributed sensing, are primarily communication challenges. Information dissemination is crucial for cooperative sensing to work. Without efficient information propagation in distributed autonomous clusters, we cannot utilise cooperative sensing in a positive way. A certain balance between communication enough and as little as possible is important, as information dissemination directly contributes to the noise and frequency scarcity that cooperative sensing is utilised to minimise. When the number of devices involved increases it becomes more challenging to maintain distributed solutions in a way that keeps complexity low. This is our main area of interest in the context of the thesis goals.

A part of information dissemination in cooperative sensing is the control channel.[1] Most control channel discussions are less useful to EmpathicWiFi due to the assumption and invariant that communication between APs is facilitated by internet or WAN access. An exception could include discussions on how scanning data is sent to the control leader in the time-division multiple access (TDMA)-based protocol in [46], but the concerns are primarily avoiding collisions on the control plane. That being said, spacing communication sufficiently apart for us to not overload WAN access bandwidth is important, and an aspect we take into consideration in our information dissemination.

1: Also referred to as the pilot channel.

Unlike *centralised sensing*, wherein spectrum usage information is still gathered from the nodes within the network but analysed and made actionable by a central unit, distributed sensing involves sharing the information amongst the empathic nodes while making independent choices.[2] This fits well with the defined objectives of EmpathicWiFi.

2: Independence in the sense that the decision is made locally without aid or direction from other nodes. Each node is *dependant* on the information given by other nodes but makes its *own decision* based upon the information it receives. A *co-dependant* relationship exists between the nodes as cooperation and information is required to make better decisions for themselves and the cluster as a whole.

Gossiping Updates for Efficient Spectrum Sensing

Yucek and Arslan [28, sec. 5B] describes and discusses both wireless sensing and how to share sensory information without using a lot of bandwidth. One approach introduced in the survey focuses on efficient coordination between cognitive radios using *GUESS* [47].[3]

3: ‘Gossiping Updates for Efficient Spectrum Sensing’.

While we wish to accomplish the same, we do not wish to utilise the radio spectrum — nor implement gossiping in this work — for disseminating the spectrum sensing information. Performing all communication over the same medium which we are trying to free up resources for diminishes the already scare radio airtime. In contrast, we wish to use and setup persistent wired connections, a distinct sub-set of radio spectrum users (WAN for MDU/high density unit (HDU)), where an alternative data medium is available. Besides our initial setup we will be able to exchange information without incurring radio penalties.

The choice of radio communication in [47] is due to all devices being primarily *wireless devices*. All considerations that apply to primarily wireless devices do not apply necessarily apply to wired devices with radio capabilities. In particular, the lack of alternative data mediums as mentioned above and movement mobility, to mention two.

3.2. Former EmpathicWifi Work

Each part of the thesis builds on the earlier work of presented in the thesis from Nygårdshaug [43]. Furthermore, the foundations for the clustering algorithm are based on the ongoing dissertation by Rønning [10].

The stark difference between this and earlier work such as Nygårdshaug [43] is a focus on implementing a fully autonomous system. In addition to the lack of any central controllers there will not be any leaders or otherwise nodes that have any further or extended roles in the network. These changes impact our decisions, and it impacts what we may build on from earlier work. The intention of creating a distributed system is twofold: allowing the system to scale up with multiple users, and to eliminate any requirement for permanent or third-party managed infrastructure.

Rønning [10] makes it clear that nodes have different jobs in different intervals. Ignoring when the intervals are, we can divide the jobs of each individual node into multiple parts:

- Gathering radio measurements for interference and weight calculations
- Fetching weights to decide its cluster membership
- Sharing weights to other nodes to let them decide cluster membership

These jobs impact the requirements in section 4.2.

ResFi-based Archictecture

Nygårdshaug [43] takes a different approach to clustering and communication both in the choice of abstraction layer and its assumptions or requirements.

As part of the process Nygårdshaug [43, p. 62] identifies three problems that Raft, as well as the rest of the suggested abstract architecture, can handle. The following three problems are in part requirements stemming from

1. Direct contact between neighbouring access points
2. Underlying group communication protocol
3. The group state is synchronised across all members

The first two of these we discuss in ResFi Overlay Network Application, whereas the third point is in Group Synchronisation and Raft, which mentions more of how Raft works.

Group Synchronisation and Raft

As EmpathicWiFi is a distributed system in itself, it is important to distinguish between distributed consensus as a term used to describe what we wish to accomplish versus a means to accomplishing it. This author considers *distributed consensus* to generally mean a distributed system involving nodes where *leaders* exist due to being chosen or required for the distributed system to successfully *achieve* practical consensus in the system.

Nygårdshaug [43] provides group state synchronisation by using the distributed consensus protocol Raft [48]. Considering its origins, as well as its then current status in educational settings, Raft was and is considered to be an easy to understand protocol. The problem of consensus is central to EmpathicWiFi, but may have varying degrees of importance. Distributed consensus is here the goal of ensuring the same state is agreed upon by all the members of a cluster or group.

Summarised, nodes in Raft have different relationships that dictate its and other members behaviour. With all nodes starting out in the same state, a follower starts an election for a leader if no communication has been received from any a leader. The details for how elections work, and the remaining details of Raft, may be read in Ongaro and Ousterhout [48]. Surmise to say that a group of Raft nodes will elect a leader of the group, which will remain the leader until it times out, steps down or otherwise becomes unavailable.

ResFi Overlay Network Application

However, one consequence of using Raft is the underlying assumption of an overlay network. The distributed consensus given by Raft requires the nodes to directly communicate with each other, across neighbours if necessary. For this to be functional the hypotethised overlay network must be built on top of the ResFi communication, which would be a new implementation. Nygårdshaug [43] suggests this overlay network, but does not specify any more how routing is dealt with. If the clustering algorithm does not require or is not impacted by the lack of the safety and fault guarantees in distributed consensus then a different approach to information dissemination may be used. Removing the need for an overlay network allows for a more relaxed approach to distributed consensus, one where the properties of termination, integrity and agreement are more relaxed. The use of an overlay network must be decided based on its positives and its negatives. Information dissemination can be done without an overlay network.

This author interprets the necessity of a overlay network in this decision as a consequence of using Raft to synchronise state between nodes rather than a consequence of choosing *minimum cut* as the optimal clustering algorithm [43, sec. 5.7.2]. We cannot find that all nodes must be able to contact each other directly in order for the mimimum cut algorithm to work, but the nodes must indeed have access to all of the link weights in the network [43, sec. 5.7.4]. This is unlike K-means clustering, which without going into the details of how it works, is identified by Nygårdshaug as ill-suited for a distributed environment due to its extensive information requirements [43, sec. 5.6.4].

The minmax-clustering algorithm by Rønning [10] allows for a fully distributed and autonumous process, wherein both the clustering and group selection is done without any leader election.

The change in clustering to an autonomous approach further adjusts these assumptions for EmpathicWiFi, which will be explained in section 4.2.

While the minmax-clustering provides the all the properties of distributed consensus in an environment without hostile agents, it may currently only satisfy parts of distributed consensus. Integrity—an area for future work and proofing—may be negatively impacted by hostile agents, and may cascade to impact the resulting agreement or lack thereof between nodes. Fortunately a relaxed distributed consensus is acceptable; however, we do not confirm or reject the possibility of negative consequences of hostile agents beyond a reduced optimisation effect.

With that, we suggest that a simpler information dissemination approach is a better initial step for an EmpathicWiFi implementation. That as a solution may allow for iterative progress without requiring the design or evaluation of a suitable overlay network.

Alternative Clustering Algorithm Requirements

The discussions and suggestions by Nygårdshaug [43] are in large part based on the initial assumptions made for both clustering and node communication and group state synchronisation.

As the clustering is explored in detail both in the thesis and in Rønning [10], we wish to continue this research by evaluating information dissemination in the context of the new CAA and its differing assumptions or lack thereof.

Both Rønning [10] and Nygårdshaug [43] target diverse and varied device landscapes. Nygårdshaug [43] refers to particularly diverse and unpredictable landscapes of network devices, or specifically APs, as a "chaotically deployed landscape" of network devices. In this regard the goals are similar, and both approaches wish. Another similarity is in the open question of the maximum group sizes. Neither of the clustering algorithms nor the suggested architecture around them define a maximum group size for the clusters.

System Architecture or Lack Thereof

The abstract software architecture and protocol component overview suggests the interplay between the suggested overlay network, the distributed group creation protocol encapsulating Raft, Resfi, and the routing component of the AP itself.

Communication within the system is limited to an abstract description in its design, with the most concrete component being ResFi as providing the north-bound API for communicating with neighbouring APs.

3.3. Empathic *minmax* Clustering

This section goes through the high-level overview of how the clustering algorithm and its implementation works, as well as the different structures that may be used for it. All work discussed introduced here bases itself on the work of Rønning [10] unless otherwise mentioned.

There are two different implementations of the clustering algorithm. The first implementation is the reference implementation used by Rønning [10]. It is implemented in Python and works on an *undirected graph*. The second implementation is later discussed in chapter 6, and while it works in a similar fashion there are some minor differences. Unless otherwise specified we refer to the initial reference implementation when we discuss how it works in this Empathic Minmax Clustering.

While the following sections provide additional detail to its mechanism, the key take-away is that the clustering has a deterministic manner that follows the same path every time it performs the clustering for a node $V_s \in G$. Furthermore, provided that the graph G and the parameter α is the same for all nodes, every $V_i \in P_i$ will derive the same partition P_i [10].

Graph and Input Preparations

The minmax clustering as implemented works on a *undirected graph* with multiple empathic nodes $V \in G$. While it may be represented as a matrix or in a different more data-dense structure, it is implemented as a key-value structure consisting of all vertices and their edges. Each edge has its own local structure identifying its edges to other nodes, allowing for a bidirectional graph. Doing so in the reference implementation viewed will however result in incorrect clustering, as edges are assumed to have the same weights for the purposes of the simulation or to already have been pre-processed beforehand.

The information each node gathers and wants other nodes to have access to is a graph of all the nodes nearby and a weighted measure of each node's interference to their neighbouring nodes. Assuming the function generating the weights yield the same for both itself and others observing the interference, this can be interpreted as a bidirectional graph composed by the weighted edges representing the interference between nodes.

Each of the nodes in the graph are unique: there is only one node in the graph per node. We refer to the set of all unique nodes in the graph, including ourselves, as the *discovered nodes* in definition 3.3.1.

Definition 3.3.1 (Discovered Nodes) *The uniquely discovered nodes are a set consisting of the nodes that a given node discovers through gathering information/reports (see definition 5.1.2)* *from its neighbours, regardless of the mechanism of information dissemination employed.*

The number of discovered nodes should be as close to $\alpha + 1$ as possible in order to successfully confirm a partitioning.

Nodes are unlikely to have edges representing the same strength of interference. This is because different APs are likely to have varying signal strengths. Until a given node performing clustering is able to calculate the agreed-upon weight between two nodes, $W(A, B)$, it will have to settle with the weight seen from node A, $w(A, B)$. The weight represented by $W(A, B)$ could be implemented as in eq. (3.1), but what is an ideal function for smoothing out the values is not looked any further into in this thesis.

$$W(A, B) = W(B, A) = \max(w(A, B), w(B, A)) \tag{3.1}$$

While $W(A, B)$ is unknown, the clustering will utilise $w(A, B)$ to evaluate which nodes are members of its cluster.

Performing the Clustering

The clustering process as simplified in fig. 3.1 takes the graph G as its input, along with an identifier of the *seed node* V_s, which is the node we wish to perform the clustering from, working as the start point in the graph. Once any internal initialisation is complete there are in this author's view particularly important parts of the process useful to know.

It starts by seeing if there is an outgoing edge with a weight higher than the current *minimum weight threshold*; the initial minimum weight is unset or 0. It selects the heighest edge and follows it. The process is continued until either it discoveres an edge with a lower weight than the minimum or the number of cluster candidates is higher than the maximum cluster size α. In the event it found no higher edge candidates on its same journey, the nodes are contracted one-by-one into the seed node. The minmax-clustering process has now found a first, and perhaps, temporary clustering.

If the process could not find further edge candidates and there is still room for additional cluster members it repeats the process, starting again from the seed node—now expanded—with all of the edges of its new clustering members. This process is run until the termination case where we either do not have any candidates over the minimum weight or any further edge candidates will put $|V|$ over the cluster size α threshold.

```
Result: AP is part of a cluster.
Input : Initial seed vertex seed, and initial input graph
partition = ⟨⟩;
set initial minimum weight = 0;
repeat
    update minimum weight to the maximum(edges);
    if maximum(edges) > minimum weight then
        repeat
            breadth first search visiting nodes where minimum
              weight is higher;
        until visited nodes ≤ space left;
        return graph;
    if |visited nodes| + |partition| ≤ α then
        forall node ∈ visited nodes do
            contract(node, partition);
until size(cluster members) ≥ α or no more edges;
if seed node has remaining edges then
    update node weight threshold;
```

Figure 3.1.: Simplified description of clustering algorithm by Rønning [10]

By prioritising the highest weighted edges of its current node it prioritises clustering nodes together that have the highest amount of interference between them.

This process requires the interference weights from the empathic nodes for us to evaluate or calculate the cluster partitioning. It does not, however, need a full or infinite graph of all connected devices. Weights can be acquired from *one node at the time* as the clustering continues, allowing for us to exploit a way to reduce the memory requirements that the input graph has.

As a result of the process the reference implementation returns a new graph where all other members of the cluster are contracted or merged into the seed node. Exactly which additional functions the clustering implementation provides, whether it is ease-of-access to the members are in a standard container, or accessing a generated list of other candidate cluster nodes, depend on the use-case.

Part II.

Methodology

Strategy 4.

The thesis looks at how we may distribute our *sensed information* among our soon-to-be cluster members, while minimising the cost to nodes that are not empathic, as well as the cost to nodes in our and other clusters. In order to answer which information dissemination design is best for the purposes of EmpathicWiFi we need to approach it from different sides.

In section 1.2 we have established multiple problems, as well as a set of research questions we attempt to answer. By approaching these with a primarily positivistic attitude, we attempt to discern their relative qualities using the observable facts as seen in our simulations.

In section 4.1 we outline how our method primarily relies on *controlled experiments*. By developing a custom set of relatively simple simulation models, we will then evaluate and discuss in what way they are suitable for the purposes of information dissemination for the project's goals. These form the basis of our results, which we evaluate by contrasting the resulting metrics between the models.

However, we identify the rough requirements of the empathic clustering algorithm in section 4.2. These represent our interpretations of which situations EmpathicWiFi are tailored to, as well as which aspects are most important.

Subsequently we introduce an information dissemination approach to implement in our controlled experiments in section 4.3. By simulating the approach and its derivatives we will be able to compare and evaluate the effect of empathic clustering on information dissemination. Our hypotheses are either rejected or confirmed through the light statistical comparisons of the metrics as provided.

4.1. Controlled Simulations

For our evaluations we need to write models using a form of event simulator. One of many discrete event simulators used for simulations of peer-to-peer applications is OMNeT++.[1] It is a versatile framework for building models and performing deterministic simulations with them, ranging from evaluating

1: The discrete event simulator *OMNeT++* is public-source, cross-platform and available online both as a tarball or as Docker containers. It may be downloaded from https://www.omnetpp.org.

software defined radio to vehicular road networks.[2] A graphical interface allows for stepwise confirmation and visualisation of the models behaviour. A multitude of different event simulators are used in academia [49]–[51], some of which are based on and extend OMNeT++ in fundamental ways. INET is one that enables simulations using TCP/IP, accurately expressing the overhead, latency and behaviour aspects of its stack [51]. Castalia is another one that adds the capability for simulating wireless sensor networks as well, including radio interference [52]. An alternative to these were ns-3, which is specifically a discrete network event simulator with built-in support for the features that OMNeT++ needs extra packages to provide. In [51] a third alternative is OverSim which simulates scenarios and allows for performance evaluation of structured overlay networks in distributed hash tables.

2: See Veins https://veins.car2x.org/

The extendability combined with its decent performance for our data sizes make OMNeT++ our choice. Furthermore, its continued development and support contributes to it as the choice as opposed to more portable and application-based alternatives such as OverSim [51].

Our work will attempt to simulate the events in the network in a fully autonumous manner, wherein each node and its behaviour is simulated individually in the system. This is different from simulating a set of nodes with a shared data structure as in Rønning [10]. Each node must make their own individual assessment with the information gained and disseminated.

Furthermore, by utilising software tests together with the framework it allows us to verify that our implementation works according to our unit or integration tests.

Technological details and metrics

In this thesis we will create a model using only OMNeT++ without any additional frameworks. It will be used for simulating the core characteristics of the different designs for spreading the information. While frameworks such as INET can add substantial value by allowing us to identify and simulate the networking stack, it is not required as the limitation on accuracy is made to focus on the traits, rather than complete behaviour with TCP/IP or other networking stacks. This is important in this use-case where all communication happens over a stable and wired connection.

At no point will any wireless simulations be performed or other wireless components taken into consideration.

Similarly to a real-life scenario the member nodes will perform information dissemination and clustering individually, but the implementation details are discussed in further detail in chapter 6.

In evaluating and discussing the approaches for dissemination across potentially infinite networks, from small ones with few neighbours to densely packed urban networks that are almost complete graphs, we must consider certain metrics to assess them. Let us look at the primary and secondary metrics and how they help us compare the designs.

The number of discovered nodes

The amount of information each empathic node has collected of other empathic nodes throughout the simulation will be used as part of the results. This allows us to highlight whether one mechanism or another contributes to less disturbance on the backhaul network, in addition to requiring less bandwidth from users, as long as the inherit challenges of the dissemination approach are considered.

The number of discovered nodes allows us to identify how much optimising the dissemination approach must be performed, and lets us answer if we are able to prune information in the process that is not important a given node or its neighbours.

The amount of data transferred

While measuring the exact size of packages across the entire network is not performed, we will use the number and size of information elements to extrapolate payload sizes. By combining this with the knowledge of TCP or UDP headers, we will establish how much information can be transferred within single transmissions.

The number of information elements transferred, combined with the amount of useful information discovered, allow us to evaluate how effective such the dissemination mechanisms are.

The metric directly establishes how our mechanisms affect backhaul communication. This is particularly important in order to see if users with residential data caps could be negatively impacted by the use of EmpathicWiFi. A substantial amount of data transferred beyond reasonable use could be cause for further work and optimisation in that area.

The time to converge

The primary motivator is the time it takes nodes to *initially converge* and *adapt to membership changes*. Small graphs in particular will be used to showcase the rudimentary behaviour of the minmax graph clustering algorithm, similar to how they are in the clustering and channel allocation algorithm (CCAA) [10]. The proposed CCAA tentatively suggests primary intervals of 5 minutes, with a possibility of longer or shorter intervals after initial clustering.

The time convergence will take is linked to how the system disseminates information. An information dissemination approach which significantly reduces the convergence time allows for different design decisions and priorities, but will typically with higher simultaneous data usage.

We expect our modifications to the dissemination approach in chapter 5 to affect convergence times, but hopefully without going beyond intervals of five minutes.

Limitations

As we rely on a discrete event-based simulator without simulating the network results our results cannot be guaranteed to be realistic if actual protocols such as TCP or UDP were used. These limitations do not prevent us from assessing the dissemination behaviour, but they do prevent us from making statements of network protocol compatibility in cases such as extreme blind flooding.

Furthermore, we cannot test infinite networks. The constraints on memory for this thesis prevent us from larger simulations, but we believe simulations with 2000 nodes to be sufficient for us to evaluate the data.

In addition, the simulation will not be able to take into account of processing time the clustering process will require on embedded devices, nor the exact it takes to transfer the information. The latter is possible to implement, but we have not implemented this, and rather set a very conservative delay that always takes 50 ms to transfer. This is extremely cautious for small packets in a local network, and grows to be far too lenient as larger information is disseminated. However, we will be able to gather the time in terms of how the dissemination schedules or delays when forwarding information with regards to hypotheses 2–4.

As we individually simulate each node in the network there is an exponential growth of memory requirements as each node must maintain a copy of the graph. A network consisting of 2000 nodes, gathering 200 or more vertices in each of their graphs, is more memory intensive than simulating values of $\alpha = 100$. This is possible to either work around or to solve, although each come with its own costs such as increased development or a removal of data separation. The priorities have been to ensure that the clustering implementation is accurate and behaving according to the specification. Trading certainty and easily identifying issues with the clustering and simulation for gains in simulation sizes is a trade-off not worth pursuing.

Mitigation strategies include reducing the cluster size, eliminating certain statistics from collection, or otherwise making implementation optimisations. In practice we perform multiple executions with the same random seeds and change the statistics that we record.

4.2. Requirements for EmpathicWiFi

The following factors and requirements relate to the workings of the CCAA. In this section we first provide a brief introduction to the information requested by EmpathicWiFi within a given period. Secondly, we discuss the requirements and prioritisations that an information sharing system for an autonomous distributed RRM presents.

State or Lack Thereof

The state required for the CAA is relatively minimal. Grouping devices together within a cluster is done in intervals[3] by each member itself, in parallel with all other devices. In a weighted graph, each AP and its neighbours are represented as nodes in the graph, with weighted vertices connecting them. In a step-wise gathering of edge values, each node will follow a *min-max* vertex reduction process, wherein knowing the weight of all vertices from a given node is critical. Each individual node is responsible for calculating this value, and any node may require it.

3: Currently, there are two intervals: a primary and secondary interval. The primary interval consists of performing the entire process, including measurement, stabilisation, grouping and channel/power allocation. The secondary interval does not perform grouping but utilises measurement and channel/power allocation within the already existing group to make required radio changes.

As such, there is no need for persistent state present *within* the cluster itself. This applies even if there are multiple different intervals—with different degrees of responsiveness to updates—used by the CCAA. Each node will keep track of the interval and its membership internally, but this is not stateful: any node can repeat the process from scratch and yield the same result provided the same weights are present.

Integrity

The integrity of interference information is essential to ensuring that the performance of the CAA is constant, and not negatively impacted as a result of tampering. For a cluster and a grouping to be successful, it is dependent on replies based on actual interference readings from nodes. The reliance on integrity depends on the level of trust we require from our neighbours. As we increase our dependency on cooperation between nodes the level of trust required also increases.

We must either be able to trust information that nodes *distribute* on behalf of others, or we must be able to communicate directly with the nodes to gather their information in which case we may compare the information with that of supposed neighbours.

However, while there are many different approaches to solving integrity as seen in distributed consensus, it is not a requirement that the dissemination methods in this thesis will attempt to meet in their models. This in spite of possible consequences of rogue information being an inability to form consistent clusters, and subsequently an inability to perform CAA. The only factor in integrity is the trustworthiness of our neighbouring APs coupled with the integrity provided by the public key-exchange ResFi performs.

Dynamic Membership and Scope

Initial membership and bootstrapping The initial setup of an empathic cluster must be a process which any empathic node can perform. This initial setup is often referred to as bootstrapping. Specifically, it is bootstrapping when no pre-existing support structures are a requirement for the process to complete. There are no requirements for the bootstrapping process, at least not in terms of time or performance. Whether bootstrapping is a once-performed operation or a frequent action due to a stateless design, is not given.

If the clustering process once-performed is otherwise stateless and straightforward, then there is an argument to be made for a stateless approach to bootstrapping too. It should not be a requirement for the system to keep state across clusterings if it does not require state within the intervals of the clustering process.

The hypothesised information dissemination used in the doctorate thesis suggests intervals, which creates specific requirements for sufficiently quick bootstrapping or the ability to join a cluster on-demand dynamically. As part of the clustering process, there is a prerequisite for information dissemination within the network.

If an overlay network is employed, it may allow for dynamic memberships where nodes join on-demand. Alternatively, it may frequently perform bootstrapping if the cost is not too high compared to the alternative of employing flooding or gossiping.

However, both the information dissemination process and any potential bootstrapping must be possible to perform in a distributed manner. Preferably as autonomous as possible in order to reduce the dependency on other nodes.

Regardless of the implementation, neighbouring nodes interconnect, share topology information and perform individual partitioning with the help of backhaul communication when there is an empathic cluster. We define an empathic cluster as *existing* when two empathically compatible devices are within transmission range of each other. An empathic clustering is the result of a performed partitioning, performed by all nodes individually. Whether this means that a member must join a pre-existing cluster and propagate its state, or whether a member will only join a cluster by taking part in its initial partitioning, is a decision to be made later.

What to weigh when we decide an approach for bootstrapping is something we discuss in section 4.3.

End-user involvement Any empathic clients cannot require end-user involvement to be involved due to technical limitations. Manufacturers may wish to allow end-user configuration of an empathic client, but this can not be a requirement for the functionality to work. There cannot be a requirement for any manual configuration.

Mobility of wireless hardware End-user radio hardware rarely changes. We believe this is trivial for consumer networks, and to an extent also enterprise settings, although these do not have the heterogenous challenges that residential WLANs do. Exceptions, such as devices being provided and replaced by ISPs do occur and increase mobility, but they are by and large insignificant compared to the lack of physical mobility in APs. Another exception is the addition of 802.11s mesh network APs provided to improve signal coverage. The location of a new or otherwise replaced AP is by and large the same unless significant infrastructure changes occur, causing wired connectivity points to be physically moved.

In the context of dynamic membership, this means that any new node neighbouring to an existing empathic cluster must be able to gather the required information to join the network of its own volition. Being within the mutual transmission range is the only requirement for two empathic APs to cooperate.

However, the scope of an empathic cluster is not boundless. Devices would in theory gain from cooperating with a more significant number of nodes, but this has associated costs and constraints. The primary constraint is the amount of memory each AP can dedicate to a potential empathic client. Thus, there must be a limit of the number of nodes any empathic client keeps in memory.

The suggested specifications of EmpathicWiFi operate with a fixed and finite cluster size α. Any given member of an empathic clustering is only a member of one of a potentially infinite number of subgraphs. Thus every node must have an invariant cut-off point where information dissemination stops. Once topology information goes beyond the threshold of a cluster, it loses all value concerning the clustering algorithm.

Performance

Processing While of lower is importance, the implementation must keep in mind that the required processing power should remain low in order to support all types of embedded devices. Consumer network devices are used in greatly varying climate conditions and employ passive cooling. Depending on the internal architecture, such as APs providing Quality-of-Service (QoS), an application may be further constrained to time slices or budgets. It is vital to consider performance as a general priority when evaluating possible solutions against each other.

Memory An implementation must thus far deal with cluster sizes in the low hundreds.[4] Cluster sizes may range from in the lower hundreds to higher numbers, but it is not known for which values of α the benefits CCAA are most helpful. A reasonable maximum is the maximum number of APs within an MDU or other geographical boundary, adjusted for the possible growth over time of devices. In terms of peer-to-peer networks or distributed systems, this is a relatively low number of member nodes. Nonetheless, it is still essential to keep the memory footprint low, especially as different embedded devices may have tighter memory budgets for processes.

4: The numbers used in Rønning [10] have values of α where $100 \leq \alpha \leq 200$. Higher values of α are possible, but it is unknown which cluster sizes yield the best balance.

For a node to perform more efficient information dissemination, more information about the given topology is a requirement for the nodes. Such topology information can be memory heavy for larger sizes of α.

For an implementation to reduce the nodes' number of duplicate transmissions, more memory is required to track which transmissions are already known for them to check before forwarding a transmission. Keeping track of received messages is especially memory heavy for connected graphs, in addition to increasing the complexity of the information dissemination protocol and implementation.

Working from the presumption that cluster sizes are—at the very least initially—small, we may perform simulations to accurately gauge whether spending time on reducing traffic to a minimum or optimising for the efficiency of information sharing is ideal.

Disturbance & Efficiency

As a close relative to performance is the efficiency at which we disseminate the information between nodes in the cluster, which we refer to as communication efficiency. While keeping network traffic at a minimum is relevant to our goals, it is unlikely to be the bottleneck that users in a domestic network environment will experience. Internet access in developed countries is, for the most part, not rate-limited and allows us to communicate across networks without incurring any further costs. As such, we should utilise the backhaul connectivity to its maximum for us to minimise the radio interference end-user devices experience due to weak channel and spectrum distribution/usage.

While bandwidth over backhaul is of low concern, it is important to avoid a situation wherein a substantial number of nodes communicate simultaneously over backhaul. Simultaneous and synchronised communication may adversely affect delay-sensitive services not only for individual tenants in an apartment block, but also networks in an MDU where Fibre to the Curb (FTTC) provides a common bottleneck for cross-unit communication.

Fitting Within the Intervals

Time spent sharing information, and subsequently time spent clustering, must be relatively predictable for values of α. Periodic clustering with multiple intervals where different tasks are performed is a part of the suggested workflow in Rønning [10].

Devices must have a synchronised clock, as well as be able to perform the tasks within each interval.

Consequences to overrunning in terms of time may lead to not completing the CCAA and possibly having to wait until the next period. If the issue persists a given node may be unable to join any correct cluster.

Connectivity

The constraints of the project require that a working internet connection is present. This is because communication is sent over the internet after having established encrypted contact with ResFi [9]. In turn, this means that we wish to utilise the cabled internet connection as much as possible, as it is less crowded, is more stable, and, most importantly, its use does not contribute to the radio spectrum issue itself.

Communication need not go two ways, but there are two options: nodes can **either pull or push information**. Pulling information leans towards a solution based of DHT. Pushing information may utilise **overlay networks for routing, but** will be more likely to use **gossiping or flooding**.

While mobile broadband users are equally welcome, they are not the primary target demographic for the solution. We suggest that mobile broadband users are primarily 1. living in rural areas where other options are unavailable, 2. visiting areas for recreational activities, with lower data caps. The exception is mobile broadband users in high-density areas where the data usage is sufficiently high, thus noticing the negative effects of lack of radio spectrum. Because of this they are not a weighted requirement, but will benefit from any optimisations to the amount of traffic sent and received.

4.3. Push-based Approach

In order to evaluate a dissemination approach for EmpathicWiFi we must choose one. In this section we outline why we have chosen a baseline form of flooding as a means to compare to in Controlled Flooding, what the baseline approach when implemented is in Implementing Blind Flooding, and why we subsequently altered it with inspiration from gossiping in Implementing Staged & Empathic Flooding.

The approaches are qualitatively compared for the requirements we outlined in section 4.2 where they specifically differ.

Controlled Flooding

As a push-based approach to information dissemination, controlled flooding as introduced in definition 2.2.3 provides us with a simple mechanism for evaluation. Central to our approach will be to apply mechanisms that attempt to the reduce the amount of information disseminated. By later modifying and specifying a different form of controlled flooding, inspired by gossiping, we enable a comparison between two information dissemination approaches in the context of the properties that exist in the minmax clustering algorithm. The two approaches allow us to compare the results of information dissemination implemented without empathic controls against implementations *with* empathic controls. This establishes a foundation that can be used to examine other information dissemination approaches and their interplay with the algorithm's properties.

See pages pages 16 to 18 for the introduction to the dissemination mechanisms in section 2.2.

Central to flooding is the notion that empathic APs learn from the information they have received, potentially apply mechanisms to reduce the information, and subsequently informs other empathic APs in the network. Rather than nodes asking nodes for information, the nodes themselves share their or others' observations by pushing it outwards. Temporarily missing information may surface at a later point, in which case it could be cached until the node receives a new update superseding the previous information.

Ways to do this include controlled flooding, blind flooding, or staged flooding, wherein an initial node broadcasts its information across all connected neighbours. The receiving nodes then repeat that process until the cut-off or termination point (see section 6.2) is reached.

Controlled, Yet Chaotic

Another motivation for using flooding to push information across backhaul is that the work of Rønning [10] simulates a simple flood-based mechanism. Our ability to compare results is restricted, but it allows for a comparison of results post-publishing. Furthermore, we can expand on the flood-based approach with modifications as mentioned in Implementing Staged & Empathic Flooding and chapter 5.

Cut-off & termination

For a simple flood-based information dissemination approach to work there needs to be a base case where messages are no longer forwarded. Two base cases are present for a flooding to be implementable without incurring loops.

1. Messages received are not broadcast back to its sender.
2. Broadcasting ceases once a cluster membership is certain.

The first case applies to all broadcasting implementations as a means to avoid infinite recursion of duplicate information. It is clear that two nodes V_i and V_j continually retransmitting the same information is undesired, and this is an easy elimination.

The second case is specific to our minmax clustering. The simulation proceeds until all nodes have validated that they are certain about their cluster membership, at which point nodes cease forwarding information. Provided that the information received by a node is then broadcast to its neighbouring nodes first, all the nodes will receive as much or more information than required to validate its cluster membership.

There are further improvements within our grasp, but this is the baseline cut-off & termination criteria which we build on.

Less is More

As mentioned in the previous section, the disseminating node in a *push-based approach* is the node that initiates the transfer of information. Central to a push-based approach is the notion that the disseminating node is in the position of having information. From there it follows that it is *ideally* the case that a node spreading information also knows something about whether the recipients are interested in the information.

If it is possible for an initiating node to be in a situation where it is able to tell—to a degree—whether or not information may be useful, then the amount of information transmitted can be reduced. If only the nodes which receive information are in the position where they can assess whether or not they need the it, then that makes a *push-based approach* less ideal, and its benefits such as speed may not be worth it. Our assumption is that minmax clustering provides us with the former property.

Other Approaches Considered

In this section we make brief comparisons of the systematic properties of the other dissemination approaches found in Paganelli and Parlanti [53] and Zhang, Wen, Xie *et al.* [54], as well as brief notes from Monnerat and Amorim [55], Kaashoek and Karger [56] and Stoica, Morris, Liben-Nowell *et al.* [57].

We do not make thorough considerations of the different approaches here, as our primary reason for choosing controlled flooding is the combination of: low complexity, being featured in related work, and easy to combine with our suggested mechanisms in chapter 5.

Why Not Gossiping?

Gossip protocols are traditionally quick to forward information, typically forwarding information to a number of neighbours immediately. This can contribute to a state of synchronised management traffic that could negatively affect delay-sensitive services over WAN.

We may alleviate concerns of high-load synchronised management traffic over backhaul by changing the mechanism. Utilising a random duration of waiting from within a set interval, static or dynamic, is a possible approach to spread out and reduce the amount of messages sent. Furthermore, the usually bounded nature of gossip protocols avoid exacerbating traffic usage in dense networks. Modifying such an aspect of a gossip protocol does not disqualify it as a gossip protocol, due to the relatively relaxed requirements [42].

However, its strengths are not important to our application. Mobility is not a strong requirement for EmpathicWiFi as target users are primarily static, even if there are exceptions and scenarios such as mobile broadband where EmpathicWiFi can still be applied.

Why Not DHT or Overlay Networks?

While the author originally gravitated towards distributed hash tables (DHTs), the implementation complexity placed DHTs outside of the scope of possibilities in this thesis. Our research questions point at other dissemination approaches as areas we want to explore, but there is enough to evaluate with the chosen flood-based approach.

A further detriment to DHTs as an alternative is that the many different approaches have different goals and strengths. While this is a positive trait for future work on the subject, it makes it extra challenging to find or devise a DHT implementation that suits our needs. Furthermore, known or large-scale DHT implementations are focused around solutions such as file sharing or high performance computing [58]–[60]. The size of the data transferred in EmpathicWiFi does not exhibit similar properties to that in large-scale peer-to-peer file sharing or file storage network; minmax clustering requires a significantly smaller amount of size to perform its task. This indicates that while a novel approach, it is unlikely to give a return on the investment in complexity without considerable upscaling of the problem.

Despite this, an initial hypothesis was that sharing neighbouring contact details with new neighbours could be done with a $\langle \text{key}, \text{value} \rangle$ data structure in the form of a DHT, where `key` would be a unique identifier usable from the cluster graph, and `value` would be an IP address and a port. While simplified, this allows for a pull-based approach to dissemination and is a possible path to explore with DHTs. The information disseminated allows us to look other nodes up and can function as a type of overlay network.

An overlay network in itself is not an approach to information dissemination. Similarly, implementations of DHTs utilise overlay networks, but the overlay networks are built on the information contained within the DHT□itself. Using an overlay network is possible together with flooding, as the overlay network only provides a further network abstraction on top of the pre-existing physical or logical network. Establishing the backhaul communication with a neighbour is the first step of establishing an overlay network if such an approach is wanted.

Overlay networks are often used for areas such as multicast over unicast and quality-of-service [61], and establishing a dynamic overlay network containing the entire cluster could allow for routing information between empathic APs.

An example of a data structure that help implement overlay networks, and that in one case is used to implement a DHT, is the concept of *finger tables* in Chord by Stoica, Morris, Liben-Nowell *et al.* [57, sec. 3.D].[5]

5: Per Stoica, Morris, Liben-Nowell *et al.* [57], finger tables are lookup tables of fixed size that each node has, maintaining information about only $(\log n)$ nodes. Simply put, the lookup entries make it easy to find which successive nodes to contact when looking up or trying to contact a given node. The entries of the table map to a ring of nodes wherein each new row increases the distance skipped in the ring by an exponent of two. Thus when performing lookups only $O(\log n)$ messages are required.

Why Not A Pull-based Approach?

While we believe that a pull-based approach will function best with an overlay network or DHTs, as supported by the peer-to-peer use-cases in various DHTs [54], we have not chosen it because we first wish to evaluate a push-based approach.

In addition to the motivations for a push-based approach, we believe that a pull-based approach cannot be implemented in a simple way that does not yield excessive multi-hop queries for information. However, until future work deals with the question we do not know if the properties of minmax clustering might support a pull-based approach more than this author believes.

We believe the initial process of a pull-based approach would involve every empathic node querying neighbours for their neighbour reports. First when a given node has fetched information from all its empathic neighbours is it able to proceed with querying nodes two hops away. It then needs to repeat the same process for all nodes two hops away, or request that set of information from the node that is one hop away. While we think nodes further down the chain will require fewer jumps due to minmax clusetring being deterministic in the way it traverses a graph, we suspect the overhead of successive pull-messages will contribute to an overhead unless mechanisms are devised to avoid it.

This is in addition to issues that must be dealt with when establishing direct connections to other devices over WAN.[6]

6: While statistics for Carrier-Grade NAT are hard to come by, approximately 17% of all autonomous systems employ it [62]. Either a solution including NAT traversal support must be present, or IPv6 is a requirement, which too will limit usability with its still relatively slow uptake.

Why Not Distributed Consensus?

Distributed consensus as a goal is laudable and a positive goal for any EmpathicWiFi implementation. However, using a system wherein elections must take place increases the complexity while not offering high benefits for an issue where integrity, concurrency issues and fault-tolerance have a low priority [63], [64].

For the initial introduction to distributed consensus in the context of our related work, see section 3.2 for a brief section on Raft as envisioned used by Nygårdshaug [43].

Elected leaders are selected members seen in clusters with an attached and centralised state, which other nodes will replace should they be unresponsive or otherwise leave the group or cluster. These elected leaders tend to perform additional tasks due to their centralised state, which other nodes take advantage of [48], [65]. Such is not the case for EmpathicWiFi; we priorities autonomous nodes that distribute the workload without singeling out specific nodes for additional work. If a node disappears and fragments the network there is no damage done to EmpathicWiFi: a disjointed empathic network does not interfere with each other, as they cannot see each other, but rather reduces the problem space and possibly also cluster sizes.

In comparison to Raft, the distributed consensus system PARSEC is able to implement secure, distributed consensus without any leaders [65]. By utilising *Boneh-Lynn-Shacham private keys* or private key shares [65, p. 13],[7] it is also able to deal with a loss of 1/3 of all nodes.

7: Where any $\frac{N}{3}$ signature shares will result in the same, bit-by-bit identical signature.

Operating without leaders is an interesting property, as well as its consistency within its gossip graph, which does contribute to it being interesting as a basis for EmpathicWiFi. While ResFi allows secure communication between nodes, PARSEC also enables initial trust to develop into trust for the entire system. PARSEC, along with Hashgraph and others, are inspired by Blockchain technology and have been referred to as potential replacements for them [65].

However, the initial membership structure in PARSEC creates challenges and limitations for us. The key exchange as mentioned above requires the initial set of machines to be known *a priori* [65, Appendix A]. Empathic nodes are never known ahead of time, which excludes PARSEC as a candidate.

As such, both Raft as an established distributed consensus implementation, and PARSEC as arguably more niche in its space, are two solutions that do not fit well with the requirements we have outlined for EmpathicWiFi — primarily with regards to distributed or autonomous work, and membership bootstrapping without user involvement or a priori knowledge.

Implementing Blind Flooding

Blind flooding is implemented based on our definition 2.2.4.

Firstly, blind flooding, while simple, can utilise information from sources such as minmax clustering or knowledge of what information it has and has not seen before. One difference between flooding and primitive broadcasts is that by flooding the network with information, each node only forwards that information which was new to itself, rather than forwarding already forwarded information.

Any old information is not propagated to its direct neighbours, and thus not forwarded to nodes downstream. This creates an effect similar to pruning in multicast, except that it is performed by a received node rather than evaluating or expecting that downstream nodes will inform us if they are interested in the information.[8]

8: As the information is old we have presumably already forwarded the information to our neighbouring nodes.

Then, in turn, the neighbouring nodes broadcast the same message to their neighbours, except for the neighbour they received the information from. While the number of messages and reports sent quickly becomes exponential as the number of nodes and edges increases, the implementation is rather simple as we will later detail in section 6.3.

State or Lack Thereof

There is no state required in order to *perform the dissemination* with blind flooding. The information is gathered until the process completes, at which point it can be restarted without retaining any of the previous information.

Dynamic Membership and Scope

There is no initial bootstrapping process required for blind flooding to work.

Performance

Blind flooding allows us to forward any and all requests received to all of our neighbours at a speed only limited by the network and the devices processing power. The benefits to the speed are strong when you consider its strength in networks with low mobility, and that the use-case for EmpathicWiFi is tailored specifically to networks with a strong enough concentration of APs for there to be fixed broadband provided to the units [2].

Empathic nodes are highly likely to be available via backhaul. We suggest that should nodes be unavailable in an an urban setting there is a decent chance that the issue is affecting multiple people in the area.

Depending on the location of the nearest ISP router and its setup, it may be possible to communicate with other empathic nodes within the same dwelling unit in a situation where the on-premise equipment is autonomous and functional.

The memory required to perform blind flooding is none if we exclude the state required to maintain and perform minmax clustering. As nodes must remember the reports they have received, in order to update their own graph of discovered nodes, this state is useful and present regardless of the dissemination approach.

Disturbance & Efficiency

A minimalist contender for simulating information dissemination is what is considered the worst-case scenario in terms of traffic: broadcasting a node's observations to all neighbouring listeners, upon which they all broadcast it too. It is a form of pushing the information across the cluster as seen in section 4.3.

As seen in definition 2.2.4, the act of forwarding messages is usually performed immediately, creating a cascading effect as the information ripples out through the network. The amount of information does not have to be much for this to contribute to a continuous and synchronised flooding of the cluster's nodes, as well as any other node in α hops range, that negatively affect and disturb other delay-sensitive services that utilise backhaul.

As the cluster sizes increase and connectivity $p > 2$, the level of traffic required would be exponential unless a new or existing optimised flood-based approach is implemented. This is a particular problem for flooding, unlike gossiping, as traffic is forwarded to *all* our neighbours rather than a pre-defined amount per round. This means that the duplicate amount of information disseminated increases drastically as a product of the graphs connectedness and our cluster size. Our choice of the cluster size threshold α directly impacts how much information the empathic nodes need in order to validate their clustering, and thus impacts how long flooding will continue and how many hops the information will traverse.

We believe it is worthwhile to find an alternative, as we hypothesise that the amount of traffic for desired values of α make blind flooding untenable as a solution on its own, unless it is limited to small values of α, e.g. 10–20.

Implementing Staged & Empathic Flooding

An alternative approach drawing *considerable* inspiration from gossiping is a periodic flood-delay based mechanism with queues. In this thesis we describe this as *staged flooding*, and it is defined in definition 2.2.10.

Our motivation for choosing staged flooding as our modification to blind flooding is because we wish to allow nodes the opportunity to listen for longer periods before disseminating information downstream.

State or Lack Thereof

The state required in order to *perform the dissemination* is not different from that of blind flooding.

Dynamic Membership and Scope

Staged flooding does not affect the properties of dynamic membership or scope compared to blind flooding.

However, adding empathic pruning mechanisms to staged flooding should affect scope. The use of empathic pruning mechanisms should result in less information being forwarded overall, when compared with blind flooding that does not implement such mechanisms.

Performance

Processing changes should not be impacted for staged flooding, unlike the memory requirements which are slightly increased compared to that of blind flooding.

In addition to the clustering information, we must also maintain a queue of reports scheduled for our neighbours. This queue is limited to the degree of the node in the network graph, or by its number of neighbours, combined with the factor of reports currently queued. The information itself is not duplicated, but we must maintain queues of pointers or identifiers for the information we wish to disseminate.[9]

9: This information is later introduced as *neighbour reports* in chapter 5.

Disturbance & Efficiency

By making a random choice for the delay within a set shared interval, also known as jittering or delay-jitter [41], every time we plan to disseminate information we can alleviate concerns of disturbing delay-sensitive services over backhaul by negatively contributing to a synchronised load on the network.

Furthermore, this allows us to listen longer for information, and send fewer packets at the same time, but will contribute to larger amounts of information being disseminated at the same time.

This should result in an increase in time it takes to converge to a clustering. Increasing the time it takes is not necessarily negative, but it is important that it does not increase by too much, and that we can further adjust the interval in the event it causes dissemination to take too much time.

Suitability with ResFi

As Nygårdshaug [43] discusses in his work, the ResFi API provides us with multiple functions which we can use to gather neighbouring ResFi-enabled nodes, send messages to individual nodes, send messages to all nodes, and a substantial number of other RRM related functions.

We do not suggest using the built-in functionality for multi-hop broadcasting in ResFi, as provided by the ResFi **API!** (**API!**). As connectedness $p > 2$ increases beyond 2 neighbours per node, naïve blind flooding will contribute to an exponential increase in traffic. More importantly, using the built-in multi-hop broadcast functionality in ResFi prevents us from queuing and scheduling updates, pruning information before forwarding it, or otherwise changing the messages send over ResFi's established backhaul communication channel [9].

ResFi provides us with our empathic neighbours, and identifies all neighbouring nodes in a unique manner by utilising IPv4. This can lead to multiple issues, of which we mention a few in section 5.1, but we are not evaluating or looking further into the consequences or challenges with using IPv4 instead of IPv6, nor vice versa.

The one-hop broadcast functionality is, however, useful for transmitting our initial information elements to neighbouring nodes. As described by Nygårdshaug [43], ResFi will establish connections to our empathic neighbours within range. Subsequently an implementation of blind or staged flooding will send its first message containing its local topology and interference, and contribute to the continued dissemination of information received from its empathic neighbours.

We are — with one-hop transmissions to specific neighbours covering the required functionality for us to disseminate information — unable to pinpoint aspects of ResFi that are incompatible with blind og staged flooding as described in the previous sections.

Suggested Mechanisms for Empathic Dissemination | 5.

In this chapter we set the frame of how we utilise *minmax* clustering based on the work introduced in sections 3.3, 4.2 and 4.3. We introduce the concepts of information elements we will use, along with their sizes and structure without considering how the specific information elements will be represented in a simulation implementation. Our suggested mechanisms for pruning, along with what we consider useful in the dissemination process, are introduced along with their definitions and theory.

5.1. Terminology & Information Elements

The terms in this section are defined by us on the basis of much verbal and written cooperation as part of the project, but not taken from the written works of Rønning [10]. They are defined here in order to support the descriptions of our experiments and models.

Neighbour Observations & Reports

Definition 5.1.1 (Neighbour Observation) *The neighbour observation is a tuple NO of information made by a node V_i observing a neighbouring node $V_{i,j}$.*

$$NO = \langle identifier, interference\ weight\rangle$$

It consists of a unique identifier and a weight that representes the degree of interference V_i experiences relative to its other neighbouring APs.

For minmax clustering the neighbour observation represents one directed edge in an input graph from V_i to $V_{i,j}$.

The size of a *neighbour observation* as seen in definition 5.1.1 depends on the identifier employed and the precision of the interference weight. In order for us to evaluate the data traffic when used with ResFi, which is our goal, we take the identifier from ResFi and use the same unique identifier that ResFi uses. The unique identifier used both in results and in the API for identifying a node is a node's IPv4 address [66, framework/agent.py, l. 637].[1]

1: A future modification and improvement to ResFi might be to swap the identifier for an IPv6 address, but this quadruples the space consumed by the identifier if employed naively.

Remark 5.1.1 (IPv4 as Identifier) There are issues with using an IPv4 address as an identifier, as briefly touched on in part I, including the global shortage of available IPv4 addresses. The most important assumption and requirement is that each node has a unique IPv4 address within the network tying neighbouring devices together.

Any use of mobile broadband could invalidate this assumption due to its heavy use of network address translation. Using IPv6 could be a requirement for any implementation that wishes to be flexible in terms of users, and especially if direct connectivity between nodes is required by the dissemination approach.

An IPv4 address consists of four parts, each representing an unsigned numeral (or `char`) at 1 byte each for a total of 4 bytes or 32 bits. The interference weight is 4 bytes for `float` numerals, or 8 bytes for `double` numerals. This depends entirely on how the interference is calculated, and as such floats provide sufficient precision for our simulation purposes. As weights from both nodes need to be taken into consideration we presume that the measurement is an absolute numeral and not relative between other nodes. Thus the size of one information element is 8 bytes, as seen in eq. (5.1).

$$|\langle \mathrm{id}, \mathrm{weight} \rangle|_B = 4\,\mathrm{B} + 4\,\mathrm{B} = 8\,\mathrm{B} \tag{5.1}$$

In contrast to an observation, the size of a *neighbour report* in definition 5.1.2 varies greatly and is in a network solely affected by the number of neighbours a node has.[2] This is specifically relevant for urban settings. Areas with the highest levels of overlapping and neighbouring APs are the most in need of empathic clustering, and simultaneously the areas where the backhaul communication costs will be the highest. Whether it is implemented as a set, a map, a linked list or otherwise a different data structure is not important for the results in the simulation.[3]

2: The number of neighbours for node V_i is the number of outbound edges V_i has, and represents in graph theory the degree of the vertex.

3: It is important in order to reduce the memory overhead of simulating 2000 empathic nodes.

Definition 5.1.2 (Neighbour Report) *A neighbour report is a tuple consisting of an identifier and a set NR of neighbour observations made by a node V_i for all of its j visible—non-hidden—neighbours. It is constrained by the identifiers; there cannot be more than one observation per identifier.*

$$NR = \langle identifier, observations \rangle$$

For minmax clustering the neighbour report represents the set of all directed edges from V_i to $V_{i,1}, V_{i,2}, \ldots, V_{i,j}$.

The size of the neighbour report as seen in eqs. (5.2) and (5.3) is the result of adding the constant identifier size with the factor of neighbours or edges it has.

$$\mathrm{N}(V_i) = \{ \text{ neighbour} \mid \text{neighbour of } V_i\} \tag{5.2}$$

$$|NR|_B = 4\,\mathrm{B} + |NO|_B|\mathrm{N}(V_i)| = 4\,\mathrm{B} + 8\,\mathrm{B}|\mathrm{N}(V_i)| \tag{5.3}$$

A set or *collection of neighbour reports* is a map of neighbour identifiers to neighbour reports. When disseminating reports they may be sent individually, or they may batched up. The choice largely depends on the information dissemination approach in use.

There is no overhead to a set or collection of neighbour reports. However, any transport layer will incur an overhead from fields such as a sender identifier, time-to-live fields, or others.

Transfer Layer Overhead

The overhead of using TCP or UDP over IPv4 can be substantial. Using Ethernet and communicating over IPv4 there is a total combined header size of 58 bytes. Then there's the difference between UDP and TCP, adding respectively 8 or 20 bytes to the header size. To keep transmission sizes within the maximum transmission unit of 1500 for Ethernet, this allows for a theoretical maximum payload size of 1472 or 1460 for UDP and TCP respectively.

The size of a collection of reports sent over UDP is seen in eq. (5.4) represented as bytes, where *i* is the number of reports and *j* is the average number of neighbours per node. Likewise we see the size for a collection of reports sent over TCP in eq. (5.5).

$$\left|\{NR_1, \dots, NR_i\}\right|_{\mathrm{UDP}} = 58 + \underbrace{8}_{\text{UDP header size}} + i(4j + 4) \tag{5.4}$$

$$\left|\{NR_1, \dots, NR_i\}\right|_{\mathrm{TCP}} = 58 + \underbrace{20}_{\text{TCP header size}} + i(4j + 4) \tag{5.5}$$

With collections the way described above we can fit 184 or 182 information elements in one UDP or TCP transmission respectively. Any arguments for choosing TCP or UDP should not rest on the overhead of the header, but rather the overhead of initialising and maintaining connections, as well as dealing with connections with such a relatively insignificant difference in overhead.

Given an average neighbour density of 15 or 16 neighbours, it is theoretically possible to transfer 11 reports in one packet whether it uses TCP or UDP.

$$\left\lfloor \frac{1472}{8\lfloor 11 \rfloor + 4} \right\rfloor = 21 = \left\lfloor \frac{1460}{8\lfloor 11 \rfloor + 4} \right\rfloor$$

If the average neighbour density is 8 it is possible to transfer 21 reports. According to Zehl, Zubow, Döring *et al.* [9] messages are signed, but no mention of any additional ResFi header overhead is specified.

Space and Complexity Consequences

The amount of information a node must retain at a minimum is the minimum for a directed graph representation of all neighbour reports and their associated observations. If the partition found is believed to be a certain result, challenges of integrity notwithstanding, all other nodes not part of the partition can be eliminated. In practice, depending on the information dissemination approach, the amount of memory can be substantial.

Remark 5.1.2 The amount of information the bidirectional graph requires at a minimum is

$$V + 2E$$

or in terms of neighbour reports and neighbour observations

$$4\,\mathrm{B} \cdot |NR| + 2 \cdot 4\,\mathrm{B} \cdot |NO|$$

While the space complexity is linear, its actual memory usage will depend on the exact implementation details. A node retaining 100 empathic neighbour reports with an average of 5 neighbours per node occupies 1.4 kB. A three times as dense network increases the space required to 3.4 kB. It follows that a node that learns of 500 neighbours with an average density of 15 must retain at a minimum 17 kB of graph information.

Memory Requirements for Transmission Reduction

We have implemented three distinct information dissemination approaches, all rather similar, yet different. One of their differentiators is how, or whether they do at all, reduce the number of duplicate messages.

EmpathicWiFi is unusual in that the topology of the network of its nodes is the exact information it benefits from receiving. However, once information of that topology has been received and sent once, there is no point to retransmitting it to other nodes. In order to avoid retransmitting the information the device can maintain a record of what information has been sent to who, or, assuming a static topology, all information received.

There are different constraints depending on the method used to disseminate information.

A flood-based approach will not—normally—wait with retransmitting the information until a later period or interval. Put in a different way: it does not need to track who sent what, it merely needs to check whether it knows of the information itself.

A staged flooding approach will need to either store the messages for later sending, or store a record of what information is queued for which nodes. The latter does not *require* much of additional space beyond $O(|V|)$ in a good implementation as it may store only as many pointers as there are vertices with associated weights we need to send out.

Analysing the space and complexity of the message passing in the process is arguably the hardest, yet one of the most important parts. These were two primitive approaches to reducing the amount of memory it takes to keep track of what has been sent and what will be sent. By checking if it already knows of the node it may decide whether to forward the message or not.

5.2. Useful vs. Useless Information

Primary to utilising the clustering algorithm to reduce the amount of information transmitted is attempting to reduce what we need to forward. The question that begs answering is 'What does a node need?'

In order to do this we need to establish what is and is not useful.

Only Information Useful to Us

For a given node V_i to perform a clustering, it must have the weights of the nodes in its partition P_i. For the node to be *certain* of the partitioning, it requires knowledge of $\alpha + 1$ nodes. However, remember that α is the maximum number of members in a cluster. A neighbouring node V_j in partition P_v which has only one member, will need the reports of nodes from α jumps. Assume that the heighest edge $\max(e) = u, v \in E | u = V_j \wedge v = V_i$ is between the two vertices. Assuming that V_j has all of the nodes required, these jumps must be at least partially within the same partition of P_i. Thus it follows that the set of nodes $n \in \mathrm{P}(V_i)$ *may* be required by V_j. Assume that the heighest weighted edge is *not* between the two vertices. If V_j has fewer than α nodes in $\mathrm{N}(V_j)$ it may still become the heighest weighted edge in the process of clustering. Thus it follows that the set of nodes may still be required by V_j. From there we cannot remove any elements from our own partition without effecting the partitioning of V_j.

However, can we remove other nodes that we have received? Which nodes are useful? Assume that the information is disseminated in a series of rounds with arbitrary periods of waiting. The node V_i receives new information V_x for its graph G and inserts updates the graph. V_i checks if the set of useful information for it $U_i = \emptyset$ is empty. If it is empty, it performs a new clustering and inserts all references to other nodes it does not have a report from yet, as well as all nodes it believes to be current members of its partition, into U_i. Subsequently it checks if $V_x \in U_i$. If $V_x \in U_i$ we perform clustering again and insert any new missing references into U_i. Regardless of the order of our information, the final set of U_i must contain all possibly required nodes to have reports from in order to reach the same conclusion as us. Any nodes not in U_i can be eliminated when transmitted to V_j.

$$\forall v \in U_i \nexists v \notin \overline{U_j}$$

While the node V_j may border to $|\mathrm{N}(V_j)|$ other clusters, we cannot know which information the other partitions would require without performing the clustering on their behalf.

Only Transmitting Information Useful to Them

A simple extension of this approach is to skip making guesses of what we know a node may need, and rather evaluate what it may need on its behalf. For each $V_{ij} \in \mathrm{N}(V_i)$ we perform clustering and track which nodes are useful to their clustering process. We then prune all nodes not useful to their clustering process from our point of view.

The trade-off to this approach is that this requires us to perform clustering for all $|\mathrm{N}(V_i)|$ neighbouring nodes, and quite a few times too. If all neighbouring are members of the same partition we may have to update their list of useful nodes every time we receive new information that is also useful to us. This can cause us to perform clustering more by a factor of $|\mathrm{N}(V_i)|)$ in the worst-case.

The extra processing time may yield less redundant information, but at what cost compared to the cheaper alternative in the previous section?

The ”Usefuls“

From the two preceeding sections we propose definition 5.2.1 as a definition or *criteria* as what constitutes useful nodes in the eyes of a node where clustering is being performed.

Definition 5.2.1 (Usefuls) *The set of vertices for a given node V_i that are 1. members of the partition P_i for V_i, 2. members of the set of verification nodes and 3. members of nodes missing that impacted the clustering. Useful nodes may be calculated for any node to determine its own usefuls, or the useful nodes for any and all other $V \in G$.*

Point three is useful in the sense that it is information that may directly impact the resulting clustering partition. However, it is not essential to pushing information in chapter 6.

The size of the set of usefuls depends on the maximum cluster size α with a maximum size of

$$\alpha + 1$$

5.3. Pruning & Reducing Outbound Traffic

Building on the discussion of useful information we establish and define two approaches to *pruning*: one based on the work-in-progress by Rønning [10], and the other an extension to that approach developed here.

Definition 5.3.1 (Pruning) *The act of removing, stopping or otherwise causing a whole or part of a stream of information to cease moving towards recipients.*

There are two fundamentially different kinds of pruning here. The first two sections introduces different types of *empathic pruning* per definition 5.3.2. Both of these build on the lessons learned in section 5.2 regarding *usefuls*. The primary characteristic of these types of pruning is that they make evaluations based on the result of performing minmax clustering, and whether or not the candidate for recipient *needs the neighbour reports* for its clustering.

Definition 5.3.2 (Empathic Pruning) *Pruning reports that are not considered useful by being a member of a set of usefuls (definition 5.2.1) from being forwarded to one, multiple or all neighbours.*

The second type of pruning attempts to reduce duplicate retransmissions. Unlike empathic pruning it does not eliminate based on usability; pruning of the other kind is based on situations where we can assume that a given neighbour *should already have received the information*. This is a critical difference which will be brought up in Results and Discussion.

Regardless of how information is pushed, any attempt at reducing the information transferred must not eliminate information that makes the confirmation of a node's partition impossible. If such a reduction occurs nodes may not confirm their cluster membership, and be in a failure situation that may not be recoverable if by induction other nodes in an entire subgraph miss their required information. We argue that the initial focus needs to be an implementation that works, followed by then reducing the amount of information transferred. Once we reduce the amount of information sufficiently we may proceed to balance our peak bandwidth usage with the time constraints.

Introducing: Projective Pruning

As we see in definition 5.3.3 *projective pruning* uses its own set of usefuls in order to determine what it forwards to neighbouring nodes. Any reports received are pruned without taking the destination node into consideration.

The definition of projective suits with some degree of overlap over geometry and psychology. The reasoning behind it stems from the key aspect of its behaviour. By using what it believes is useful to itself it projects its assumptions of what is useful onto its recipients.

Definition 5.3.3 (Projective Pruning) *Empathicly pruning (definition 5.3.2) where the set of usefuls is the current forwarding node's set of useful nodes.*

This makes an implementation of projective pruning easier, as it can prune reports on a message-by-message basis, while simultaneously suitable for a dissemination mechanism that is staged or queues and schedules the act of forwarding information.

See fig. 5.1 for pseudo-code for how projective pruning can work.

Introducing: Sympathetic Pruning

By increasing the amount of work *sympathetic pruning* per definition 5.3.4 hopes to further reduce the set of neighbour reports that are forwarded to our neighbours. It attempts to do this by calculating the set of usefuls for each neighbour, thus performing minmax clustering for every one of them.

```
Data: Queue of neighbour reports to send
Result: Queue of reports where reports that should not be
        forwarded have been removed
1 while reports are queued do
2     pop the next queued neighbour report;
3     if it is a member of our node's set of usefuls then
4         keep the report in queue;
5     else
6         remove the report;
```

Figure 5.1.: Pseudo-code for how projective pruning works

Definition 5.3.4 (Sympathetic Pruning) *An empathic pruning mechanism (definition 5.3.2) in which where each neighbour has its own set of usefuls as seen by the current forwarding node. Each neighbouring node's set of usefuls as calculated by the sender is used for the empathic pruning.*

As the minmax clustering takes a network graph and a seed node as input to the algorithm, we therefore replace our own node V_i as the vertex node with each of our neighbours $V_{i,1}, V_{i,2}, \dots, V_{i,j}$. While fig. 5.2 shows the rough mechanism of how the pruning of the reports works, it does not take into account any complexity following from the combination of queuing reports or retaining reports for nodes that have yet to announce that they are empathic.

Unfortunately no node can *know* which information a neighbour has beyond the information it has forwarded to the node. We will not know, and we will probably still get it more wrong than right. However, we can attempt to be sympathetic to what the node wants.

Thus this pruning mechanism is sympathetic, as it cannot truly know what the node deems useful, but it can make an attempt to put itself in its shoes.

The strongest downside to sympathetic pruning is the increased complexity. It is clear that we must perform clustering more than for projective pruning, as a node must perform clustering for itself, and subsequently also perform clustering for nodes that are not members of its own set. There is an increase in the space complexity because it must store multiple sets of useful nodes, one per neighbour in a different cluster. [4]

4: Optimising the clustering process by having nodes share sets of useful nodes if they are members of the same partition is a way to lower the requirements of this pruning mechanism.

Self-pruning: Pruning Duplicates

In Lim and Kim [67] two approaches to reducing redundant forwarding or broadcasting of information are suggested. One of them is *self-pruning*, an easy-to-implement mechanism that allows us to eliminate our neighbours that are also neighbours with those neighbours sending us new information. By utilising self-pruning we can reduce the dissemination cost. Unlike blind flooding, where the number of recipients $|T(V_j)|$ would be $|N(V_j)| - 1$, we can for certain degrees of connectedness reduce it by 20%–50% [67, fig. 6–7].

```
Data: Map of queues of reports to neighbours
Result: Queues for neighbours where reports that should not
        be forwarded have been removed
1 foreach neighbour's queue in queues per neighbour do
2     if neighbour's set of usefuls is out of date then
3         update the set of usefuls by clustering on neighbour's
            behalf;
4     foreach report in queue do
5         pop the next queued neighbour report;
6         if it is a member of our neighbour's set of usefuls then
7             continue;
8         else
9             remove the report;
```

Figure 5.2.: Pseudo-code for how sympathetic pruning works

Definition 5.3.5 (Self-pruning) *The act of pruning reports from forwarding if our forwarding targets are neighbours of both the current forwarding node and the node which originally sent the reports to the current forwarding node [67], [68].*

Originally we wanted to implement multiple types of pruning mechanisms that are unrelated to the min-max clustering mechanism itself. The combination of pruning mechanisms such as dominant pruning and *improved self-pruning* should be explored further.

There are other mechanisms that reduce the amount of traffic sent by reducing the number of duplicate transmissions that occur with high connectedness and to multiple paths to other nodes "downstream". *Dominant pruning* is one of them [67], wherein the strategy change from reactively pruning recipients and instead proactively designate which nodes will forward the information [69]. There are also improvements made to the self-pruning mechanism that show improvements without changing the strategy from reactive to proactive [69].

5.4. Scale of Communication

With definitions 5.1.1 and 5.1.2 as background information we can take a look at the potential scale of report sizes in terms of size and number of reports.

Amount of Bytes Transferred

The first initial report for a node V_i with $|\mathrm{N}(V_i)| = 15$ neighbours has a size

$$|NR| + 15|NO| = 4\,\mathrm{B} + 15 \cdot 8\,\mathrm{B} = 124\,\mathrm{B}$$

How much could be transferred from a node in a given cluster? Given a vertex with 7 queued reports, each with an average number of observations, the size of the reports would be

$$\sum_{j=0}^{7} \mathrm{f}_R(V_j) = 15 \cdot \big(4\,\mathrm{B} + \mathrm{f}_O(15)\big) = 15 \cdot 124\,\mathrm{B} = 0.85\,\mathrm{kB}$$

How about *one* forwarding of almost all the reports within the cluster size α? If one node were to forward i.e. 100 reports, the sum is a bit larger.

$$\sum_{j=0}^{100} \mathrm{f}_R(V_j) = 100 \cdot \big(4\,\mathrm{B} + \mathrm{f}_O(100)\big) = 100 \cdot 124\,\mathrm{B} = 12.1\,\mathrm{kB}$$

In the worst-case scenario a process with push-based dissemination it may be theoretically possible that a node will use α^2 nodes. In the case of staged flooding, this requires an unlikely permutation of worst-case jittering that comes into existance without any cooperation. Such an event requires that all edge weights match with the unfortunate order of the messages received such that the clustering process evaluates α^2 neighbour reports. We think this is unlikely as the average number of neighbous, the degree in the graph, may exhibit a great deal of variance.

Number of Reports

A caveat to keep in mind is that a graphs consisting of small-world behaviour[5], or indeed any case of two hops or more being connected together and creating cycles, will exhibit duplicate information transfer as a result of information taking multiple different paths to the same destinations, unless it is optimised for. A graph consisting of cycles separated by two-hops or more are not detected, or in any way mitigated, by self-pruning.

5: Links to other nodes traversing larger distances of the network.

There is no mechanism implemented to reduce duplicates for nodes that are two hops away, although such a mechanism *can indeed* be implemented in a real system. One mechanism for this is to calculate the minimum-spanning tree, and to use this in order to reduce the number of retransmissions, but this is *not implemented* in this thesis and our simulations.

For an example relating to the number of reports, imagine a vertex V_i with 15 edges. 14 of these edges go to vertex $V_{i,1}, V_{i,2}, \ldots, V_{i,14}$. None of the neighbours of those vertices are neighbours of V_i.

$$\mathrm{N}(V_i) \cap \{V_{j,k} | \mathrm{N}(V_j), 1 \leq j \leq 14\} = \emptyset$$

In such an event V_i may forward upwards of α reports to each of the 14 nodes, and each of those nodes may forward the same amount provided they are in the same partition P_i.

The sum of sent reports from V_i would in which case be

$$14 \cdot 100 = 1400$$

and the sum of sent *and* received is

$$2 \cdot 1400 = 2800$$

In other words, the number of reports sent, and subsequently the amount of traffic, can increase very quickly when the density or range increases.

This is not necessarily a realistic example in itself for $\alpha = 100$, but can be altered to apply to any number around α, or even higher. The total number of reports forwarded is likely to be even higher over the duration of the simulation, as our sets of usefuls are likely to contain nodes that in the end were not members of our or our neighbours partition P_i. This will happen as we cannot guarantee we will receive the correct information in order, thus calculating temporary clusters which are to some extent incorrect.

We keep in mind that the order of the reports we receive combined with the time of forwarding reports impacts the history of the set of members in P_i as calculated throughout the simulation *before* the reports actually required for a node V_i to reach P_i with “certainty”.

Simulation Models and Development 6.

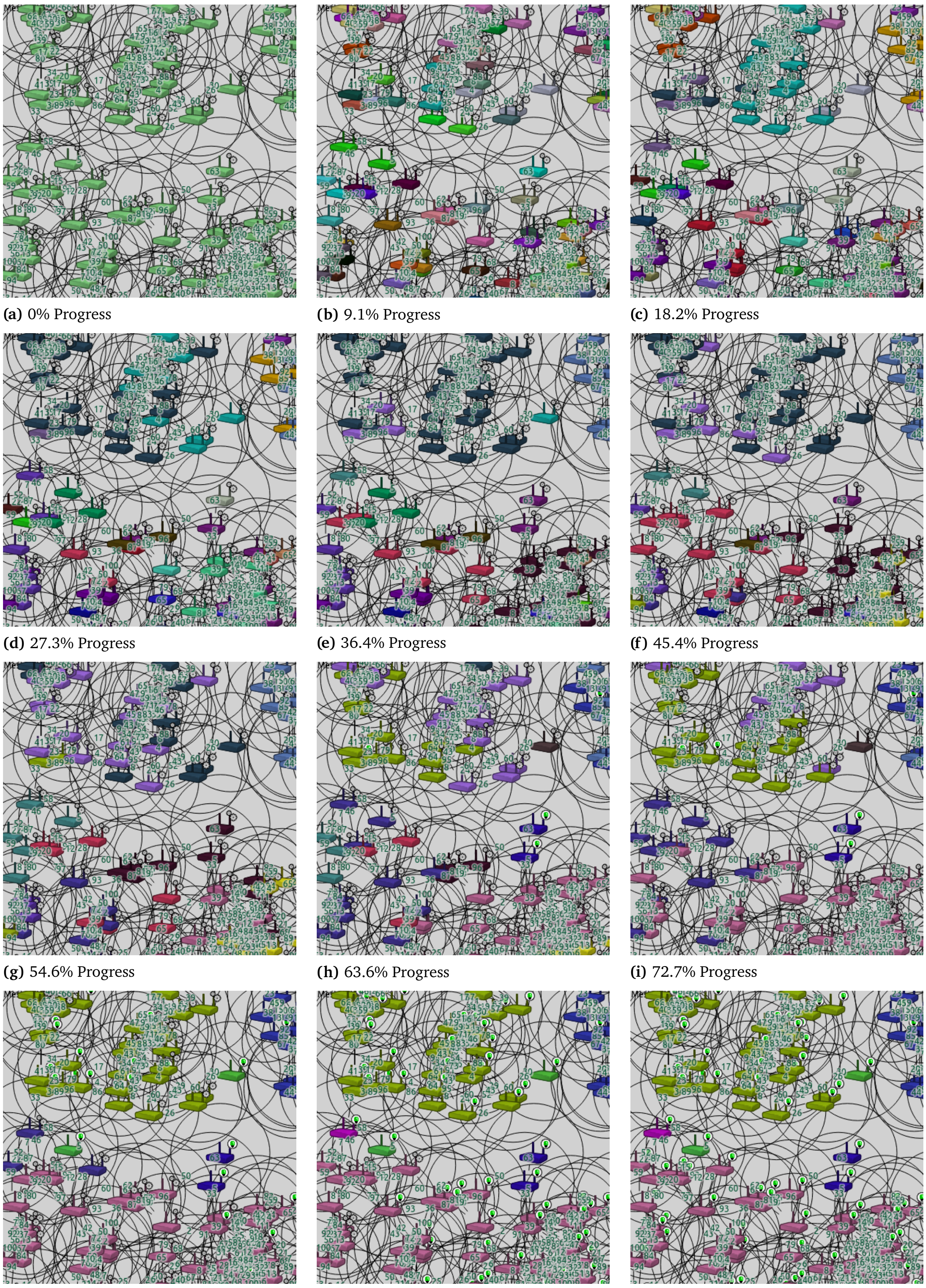
In this chapter we show how the simulation models are implemented, as well as which features they have. The limitations to our simulation approach are outlined in section 4.1, after describing our choice of simulation framework.

One way of visualising the experiments is shown in fig. 6.1. The grid of images shows the same cropped section of the simulation visualisation, taken throughout the simulation.

The goal of the C++ simulation implementation in OMNeT++ has been to focus on simplicity, reusability, and general applicability to minmax clustering. As such, the codebase has gradually grown, with the early addition of unit tests using for the clustering algorithm with *Catch2*, as well as regression tests for the simulations. The latter tests are implemented as a combination of customised OMNeT++ fingerprint tests, and adverserial tests that pinpoint any unintended changes to clustering results. A Continuous Integration workflow for the source code repository runs the fingerprint tests each time a new change is made, and notifies of any errors.

Once an approach was chosen with a focus on iterative development and incremental changes, it was subsequently challenging to correctly prioritise the changes to make. The two approaches have multiple aspects that can be altered. Therefore, we chose a dynamic approach to adding and changing central behaviour for the staged flooding model. The pruning mechanisms as implemented in this chapter are all implemented as classes implementing the interface `Pruner`, which provides a set of hooks pruners must implement the behaviour in.

At the time of publishing, the codebase contains 2650 lines of C++ source and 1032 lines of C++ headers. The source code repository for the simulations is not (yet) published.



(a) 0% Progress **(b)** 9.1% Progress **(c)** 18.2% Progress

(d) 27.3% Progress **(e)** 36.4% Progress **(f)** 45.4% Progress

(g) 54.6% Progress **(h)** 63.6% Progress **(i)** 72.7% Progress

(j) 81.8% Progress **(k)** 90.9% Progress **(l)** 100% Progress

Figure 6.1.: A sequence of cropped screenshots showing an a view of the clustering progress in the staged flooding model from section 6.3.

Visible are the range indicators, the weights between devices, as well each AP's execution status. From the first to the last frame we illustrate the gradual convergence to correct clusters, represented both by the colour of the AP, and the green bulb upon completion.

6.1. Overview of Models

We can see an overview of the pruning approaches for the models in table 6.1. Each row is one of the simulation model with their *primary* configuration in use. While the flooding only comes with one, both the standard gossip and empathic gossip model come with a product of features. They may be changed on a per-experiment basis, but doing so would not match the configuration in which the experiments have been executed with and which is discussed in chapter 7.

As part of the project work a faithful gossip model was partially developed. Work remained in order to bring it up to the requirements for using projective or sympathetic pruning when combining it with gossiping.

The simulations are divided into *experiments*, *networks*, *models* with *features* and *statistics*. Each experiment is a combination of different simulation variables, such as our cluster size α, or the state of our pruning mechanisms. The different *models* have different *feature toggles* and *feature parameters*, allowing the simulation to represent several different model variants or permutations. From these we get a matrix of executions, each unique in its permutation of the available *variables*. A typical experiment toggles more than one *feature*, thus changing the behaviour of the model or the network. Each run generates a set of outputs, ranging from already processed statistics, scalars, timestamped vectors to simple counting. These executions are performed with a certain number of replications, each of which alters the seed to the deterministic random number generator.

An underlying assumption for all of the simulations is that there is no divergent behaviour within the same experiment, such as nodes with contradictory modus operandi. This is because any empathic node plays by definition by the same cooperative and empathic rules.

Model	Type	Queue Reports	Prune sent	Self-pruning	Projective Pruning	Sympathetic Pruning
Blind Flooding	Naive	—	—	—	—	—
Staged Flooding	Simple	Yes	Yes	Yes	—	—
Projective Flooding	Enhanced	Yes	Yes	Yes	Yes	—
Sympathetic Flooding	Enhanced	Yes	Yes	Yes	—	Yes

Table 6.1.: Table overview of simulation features and behaviours.

6.2. Common Implementation

The different simulation variants are described in their respective chapters, whereas the shared features, characteristics and implementation specifics are detailed in this section. At a base level all models in the simulation send a representation of its known neighbours and their weights to other neighbouring nodes.

The models have been created with a focus on maintaining simplicity and understanding of how they work. In order to do this, no other layers except for the basic `cSimpleModule` is used to represent devices. The models used both the simple model `cSimpleModule` for the purposes of modelling the scale and complexity of solutions.

The `cSimpleModule` model is appropriate for focusing on space, complexity and the logics of the solution. Compared to the full networking stack available by utilising INET[1] the simple module is insufficient for us to look at issues such as network saturation, but this is not necessary for our specific area of focus for simulating the scale and complexity.

1: As mentioned earlier, a framework for OMNeT++ that allows for full simulations of the TCP/IP stack and allows us to simulate our models with accurate depictions of bandwidth, latency, et cetera.

Variables and Signals

All simulation models inherit from a parent `Node`. The child nodes regardless of their model respective implementation do support the following table of features as seen in table 6.2.

The core `Node` implementation also emits certain signals irrespective of the model implementation, as can be seen in table 6.3.

Network Topologies

The simulation models use two different topologies. The first are randomly generated mesh topologies described in the first section below. The second is a set of five topologies generated by Rønning [10], described in the second section.

The randomness in the simulation determines *the order of events*; each node shares its information and lets its neighbours know

Table 6.2.: Customisable parameters for models extending from `Node`

Feature	Variable	Values
Cluster size	`clusterSize`	`50..250`
Clustering timeout	`timeout`	`360s`
Number of nodes	`num`	`0..2000`

Table 6.3.: Available signals for models extending from `Node`

Feature	Variable	Values
Clustering complete	`clusterComplete`	Timestamps of completed clustering
Clustering timed out	`clusterTimedOut`	Timestamps of timed out clustering
Discovered nodes	`knownNodes`	Unique nodes discovered after useful update
Node connections	`nodeConnections`	Count of direct neighbours
Reports received	`rxReport`	Received a report
Reports transmitted	`txReport`	Sent a report
Observations received	`rxSize`	Received an observation
Observations transmitted	`txSize`	Sent an observation

that it is empathic at a random point in time in the interval $[0, 60]$. Furthermore, the same randomness determines when the next forwarding of received information will occur for staged flooding models in the same interval $[0, 60]$.

Random Mesh Topologies

The simulations are all based on a randomly generated *mesh topology*. The topology is drawn on a 2D plane and each node is given an X and Y coordinate that are drawn from the deterministic pseuedo-random number generator.[2]

2: See the NED function `uniform()` in OMNeT++ and the underlying function.

When the network is generated it creates one by one node. Each node is connected to all nodes within its range. To determine whether a node is in range of other nodes the range between each node is calculated and verified.[3] The mean number of connections each node has for each value of the parameter used, `range`, is illustrated in fig. 6.2. Where possible we will prefer to calculate the average number of neighbours per value for `range` and show that instead.

3: The NED language is not optimised for generating large topologies. An alternative to making the topologies dynamically is to generate them using a different tool and export them as static topology definitions.

Static Mesh Topologies

The set of five topologies generated by Rønning [10] were used in order to compare the results of the work in this thesis with the simulations by Rønning [10]. They have a higher average graph degree, or connectedness, with an average of 15 neighbours per node. This should result in a graph where the redundant transmissions that occur due to flood-based shortcomings, such as the many paths to the same nodes, should be extra prominent.

The topologies do not exhibit small-world phenomenon, as they are also generated by placing nodes on a two-dimensional plane and collecting nodes within a given range.

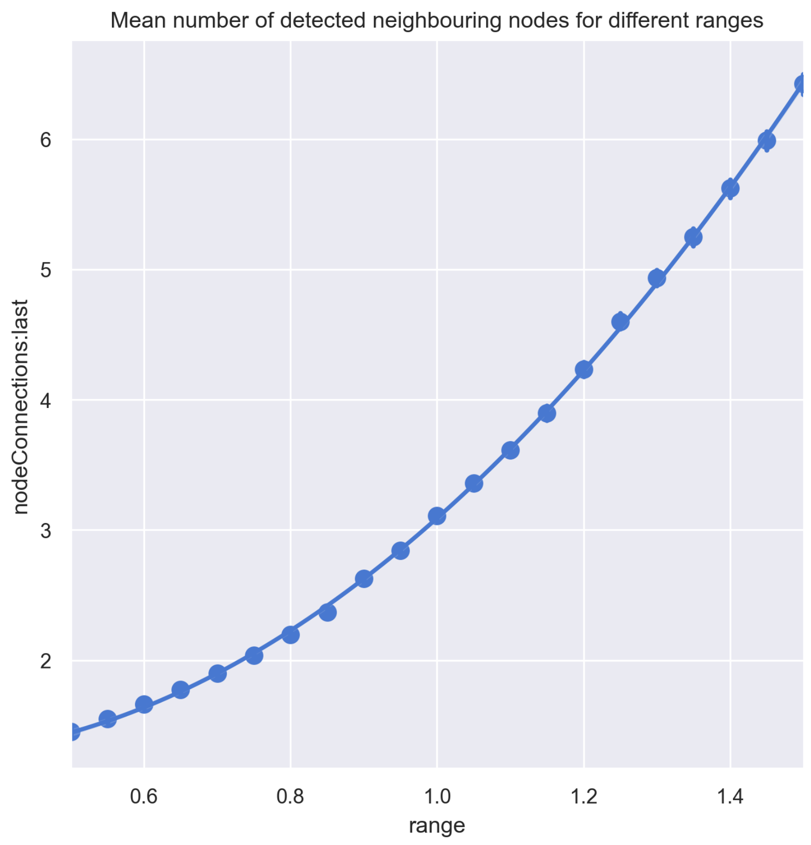


Figure 6.2.: The graph shows the mean number of connections each node has for different values of the parameter `range`.

The static mesh topologies were made with *five distinct random seeds*. These random seeds were shared across all runs, e.g. using the same seed for the 2nd replication of the 3rd and 4th topology.

Weights & Message Format

Weighted Edges

Weights, too, are drawn from a uniform random distribution, and are independent of the location of a node. Each channel connecting two nodes together consist of two weights. Upon initialisation the weights on the channel end of the first node to initialise is chosen as the weight for both ends. The same topology is accessed by the neighbouring node which updates its own internal representation of the weights to match that of the simulation topology.

Each individual node generates its own representative weight for each of its gates. As the nodes start up one of the two available weights in the graph are chosen. Once chosen, the weight is representative for the link between the two nodes. Additionally, that weight is visible in the graphical visualisation of the simulation, as in fig. 6.1.

Messages and Message Format

The messages used in the simulation are purposefully *simple*. Each message inherits directly from the `cPacket` module and provides exactly two fields, one of which has a variable length. This follows from our earlier discussion in section 4.1 where we limit ourselves to a simple message model in order to reduce implementation time and efforts.[4]

4: The addition of INET remains a suggestion highly recommended in order to perform accurate network simulations. Examples include establishing if, and how, delay-sensitive services may be impacted by backhaul communications in the event of synchronised information dissemination.

$$O() = O\left(n^2\right) \tag{6.1}$$

The first field is a time-to-live (TTL) field primarily used by the flood model in section 6.3. While the other models may use it, they both queue the information received for later without storing a TTL field specific to that report. As such, the TTL is not helpful for these specific models.

The second field is the collection of neighbour reports. The simulation simplifies the identifiers of the models to either unsigned shorts (16 bits) or to strings for ease of debugging. Besides the identifier, the structure maps to the information elements as described in eq. (5.2).

Clustering Implementation

The clustering implementation is a port of the original Python clustering module used by Rønning [10]. Because OMNeT++ is written in and for C++ [70], the clustering implementation has been ported and subsequently rewritten from its original Python implementation. The C++ version is changed from its original implementation primarily in how functionality is implemented, not in how the clustering itself is performed. The rewrite includes a simpler breadth-first search, as well as support for returning a list of partial nodes. These changes remain identical across the varied methods of information dissemination.

We propose that the establishment of a cluster is performed collectively and simultaneously with joining a cluster. While a performance benchmark is not a part of this thesis, it is clear that the C++ implementation performs clustering for values of $alpha$ so quickly as to be a cheap operation.

The first creation of an empathic cluster between two nodes happens when two empathic nodes are within range of each other and have exchange their first report. All models use a first or initial one-hop report before any further improvements to the dissemination begin. Whether a real-life implementation would use ResFi to establish if a neighbouring node is an empathic one before sending the first node is not of importance: even for high-density networks the amount of information in the first report is essential yet low with one AP's direct neighbours.

Confirming Clustering

One of the changes in the C++ implementation of the minmax-clustering compared to the original Python version is the inclusion of *partial nodes*.

The clustering implementation, both the original Python version and the C++ version, takes into consideration whether we *know* a node or *know of* a node. Confirming that a partition is correct is done by looking at *partial nodes* where we only *know of* them due to a *known node's* edge pointing to one. Partial nodes allow us to see identify that a clustering could be different, but this is unknowable until the node is upgraded to a known node.

The following describes the clustering process from the point of view of a single empathic node.

1. Add ourselves as a known node.
2. Add our neighbourhood as partial nodes.
3. Process report from neighbouring node.
   a) Convert partial node(s) to known node.
   b) Add new neighbourhood of known node(s) as partial node(s).
4. Perform clustering process.
5. Evaluate clustering results.
   a) Repeat step 3 and 4 if there are partial results.
   b) Finish process if there are *no* partial results.

Receiving Weights

The simulation sends a map of node to neighbour reports, where each report consists of neighbour observations in the form of a map of known neighbours and its weights to them. It represents a single node in the graph with its weighted vertices. By utilising every connected link for sending the message over the cryptographically secure backhaul, we flood our neighbours with the weight information. We refer to this as the *weight message*. Every recipient checks if `isNewWeightMessage(msg)`, and either discards the message or updates its local weight cache in the current interval[5] before retransmitting the message to $n-1$ neighbours.

5: The interval in which nodes exchange neighbour reports and perform the clustering.

Cut-Off Point or Termination Criteria

Any dissemination approach utilising a type of push-based dissemination approach needs a termination criteria for the information being spread out.

The termination in our model assumes *no information will be or is lost* in the network. Fault tolerance is thus not taken into consideration. This is a limitation of our current simulation, but can be improved on within OMNeT++ by utilising unstable communication channels and devising counter-mechanisms. A real-life implementation can take this into account by implementing a mechanism in which nodes directly contact each other to perform error or failure corrections.

Regardless of the cut-off methods employed, we can see that it is clear from these sections that any node will not receive neighbour reports from all the nodes in all the clusters reachable in the network, so long as the node is not actually connected to all the clusters in the network.

As discussed earlier in section 4.3, there are primary two methods this can be utilised in our simulation environment, and a third which should be α and model specific. Our implementation has support for the following three cut-off points or termination criteria.

Time-to-live

It is presumed and recommended that any real-life case also uses a time-to-live field to stop any messages from going more than the maximum cluster size α number of hops away from its origin. The TTL is set to the value of α, thus limiting its number of hops to a worst-case of $\alpha - 1$ unhelpful hops or α helpful hops if the node borders between two clusters.

Clustering Verified

The empathic cut-off point is when a node has been able to confirm that it has no alternative possible partitions, by gathering the observed information of minimum $\alpha + 1$ nodes and performing the clustering process.[6]

6: The relation between this and the set of useful information empathic models forward is discussed in more detail in section 5.2.

An exception to this exists within staged flooding in section 6.4 when a node V_i finds its cluster P_i before a neighbouring node V_j sends its initial neighbour report, as this means we must forward the information that V_j has thus far missed out on.

A useful message resulting in a confirmed empathic clustering will be forwarded as usual, but any future messages are discarded or not added to the queue of outbound reports. They can be discarded because all useful information has already been used to find the only possibly clustering.

To reiterate, the same applies in the sympathetic pruning model in section 6.5, as it is not possible that a first node V_i, having found its cluster P_i with the information it has received, has not also forwarded all information that would be useful for a node further downstream.

Timeout

The last fallback is that each node in the simulation has a maximum runtime. Timeouts are useful to prevent that a node proceeds beyond the limitation of the round, but the timeout implementation in the simulation model has only one purpose: stopping once enough time has passed and admitting defeat.

Timing out may not be equally sufficient for a flood-based implementation, however. In the best-case scenario a timeout for a flood-based implementation needs different timeout values as there is no timeout prevents the immediate cascade of information from spreading uncontrollably should there be bugs in the implementation of the blind flooding (definition 2.2.4).

Contrast that to the empathic implementation, or one based on another approach. Whether there are fixed or random intervals, a timeout value must be balanced with the amount of data yet-unfinished nodes could forward or produce in a subgraph of the network. It is important to balance the rate of dissemination with what is considered a realistic timeout value, but this is a separate evaluation that must be done on a per-α basis.

Alternatives

A model node could perform clustering for other nodes it knows of, or check whether a neighbouring node is a member of the same partition in the same round, and pause further dissemination downstream until the neighbouring node is included.

Initial start-up

Once a node in any of the models initialises itself it schedules a time for the initial report to be sent to adjacent nodes, as

```
Data: Immediate neighbours and measurement report
Result: All neighbours receive a copy of our measurement report
determine startup time delay in $[5, 60)$;
wait(startup time delay);
foreach neighbour in neighbours do
    sendMessage(neighbour, weightCollection);
end
```

Figure 6.3.: Pseudo-code of the initial dissemination of neighbour reports upon start-up

illustrated in fig. 6.3. Each node chooses a random time between 5 and 60 seconds after initialisation to send its inital report. This reduces the number of simultaneously communicated nodes over backhaul, while also staggering the communication for the non-flooding based models.

When the scheduled event fires it proceeeds to send its neighbour reports for its *one-hop neighbours* to all of its *one-hop neighbours*. The adjacency list allows for each node to perform *self-pruning*, which is relevant the staged flooding model in section 6.4 and its derivatives in section 6.5.

6.3. Blind Flooding Model

The blind flooding model (definition 2.2.4) is a naïve model. This is because the underlying approach is itself incredibly naïve, sending everything to everyone until its time-to-live reaches zero. Unfortunately, the simulation model has its limitations as well, and does not represent how a network would react during an actual broadcast storm.

Flooding is the fastest way to spread information across all nodes in a graph as it always takes the shortest path. It also takes all the paths, which is why it will also choose the shortest paths.

Figure 6.4 on page page 77 shows a high-level overview of the process within the blind flooding model.

Simulation Features

The simulation supports no additional features beyond the common functionality.

Messages that node V_i receives in the blind flooding model are—if we receive a new neighbour report and thus discover a new node in the graph (see definition 3.3.1)—immediately dispatched to neighbouring nodes.

Reactiveness

Initial versions of the staged flooding allowed to simulate between a *reactive* and *non-reactive* behaviour when receiving neighbour reports. This was removed, as reactiveness is inherent in the behaviour of the blind flooding mechanism. Reactiveness means that we do not wait when we receive information, but we immediately disseminate it to our neighbours if it passes through our pruning or control mechanisms. This significantly reduces the time it takes for information to disseminate in the given cluster, but it also increases the total number of messages sent, along with the simultaneous amount of information in the network at the same time.

Pruning Messages by Time-to-Live

Blind flooding has by definition few means of limiting its possibly infinite reach of its broadcasted messages. Due to its limited scope it is also the most restricted and least performant model in the simulation, despite the use of TTL.

The flooding model requires that a time-to-live (TTL) field is used in the same way that TTL is used in the TCP/IP stack. Every initial message contains a `timeToLive` field set to the value of `clusterSize`. The TTL is decremented by one when a message is received. If the TTL is now 0 the message is processed locally, then discarded. If the TTL is higher than 0 the message is flooded to all neighbours that are not V_t, the source node $\mathrm{N}(V_i) - V_t$.

Pruning forwarding upon certain clustering

In section 6.2 cut-off points, such as both time-to-live and certain clustering, are discussed more in detail.

The flooding model discards all messages it receives once it has validated its current clustering partition with $\alpha + 1$ reports available.

Resource Consumption

The blind flooding model uses no sustained amount of memory for its information dissemination approach beyond what the common model and the clustering implementation does.

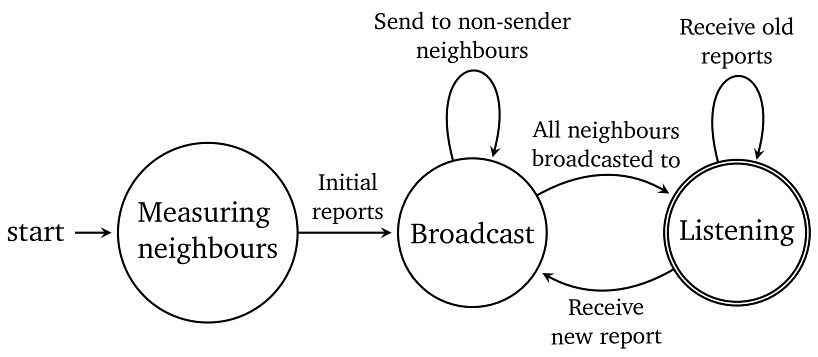


Figure 6.4.: A high-level deterministic finite automata for the blind flooding model.

Note that the steps "Send to non-sender neighbours" include decrementing the time-to-live field in the messages.

6.4. Staged Flooding Model

The staged flooding model is the foundation for the two enhanced models in section 6.5. It shares the same common traits as introduced in section 6.2, but adds additional behaviour.

Figure 6.5 on page 79 shows a high-level overview of the process within the staged flooding model.

Simulation Features

Based on staged flooding as introduced in section 2.2, it implements a model that supports *self-pruning* and the ability for other pruning mechanisms to be dynamically added to the simulation.

The implemented simulation supports the following features as seen in table 6.4.

The functionality of the received pruning is elaborated upon in section 6.4, and `pruneType` in section 6.4.

Feature	Variable	Values
Received Pruning	`pruneReceived`	Default: `on`
Self-pruning	`pruneSelf`	Default: `on`

Table 6.4.: Simulation features in the staged flooding model

Pruning Reports Received Before

The *received pruning* provided by the simulation parameter `pruneReceived` is a simple pruning mechanism that operates when reports first arrive. As the mechanism is simple and not related to any other mechanism, it does not have its own introduction elsewhere.

If a collection of neighbour reports do not a new neighbour report, allowing us to discover another node, the neighbour report is dropped before reaching the queue of reports.

Pruning Nodes Shared With Senders

The parameter `pruneSelf` allows us to enable or disable *self-pruning* (definition 5.3.5). It removes neighbour reports queued for a neighbouring node if that node is also a neighbour of the node that sent the neighbour reports to us.

Handling new information we receive Once a node V_i receives the report from adjacent node $V_{i,t}$ it attempts to update its local graph G. For each observation in the report a knowledge mapping is added between the transmitter V_t and the neighbours shared with V_i. This constitutes the set of removed forwarding nodes $R_{i,t}$ between V_i and V_t.

$$R_{i,t} = \left\{V_{i,j} | V_{i,j} \in \mathrm{N}(V_t) \cap \mathrm{N}(V_i)\right\}$$

Pruning reports when sending updates When our scheduled update event occurs we walk through our set of currently queued reports. The reports are filtered for each adjacent node $V_{i,0}, \ldots, V_{i,j}$ according to the map of reports that should be known for each of our neighbours.

As such, no node receives any reports that we received from the intersection of our neighbouring nodes $\mathrm{N}(V_i)$ and the nodes V_t neighbours $\mathrm{N}(V_t)$ which sent the report—or reports—to us.

Resource Consumption

The self-pruning implementation is not necessarily optimal in our simulation model. As the pruning of reports must be made when new neighbour reports are received it must store which neighbours are excluded from receiving which reports until the scheduled forwarding or update time.

Thus the memory consumption in the event all neighbours are shared is

$$|NR|_{V_i} \cdot |\mathrm{N}(V_i)|$$

It is the product of the number of received reports with the number of neighbours. An alternative implementation could tie the identifiers of each report received to the node that sent it, but this results in the same space complexity.

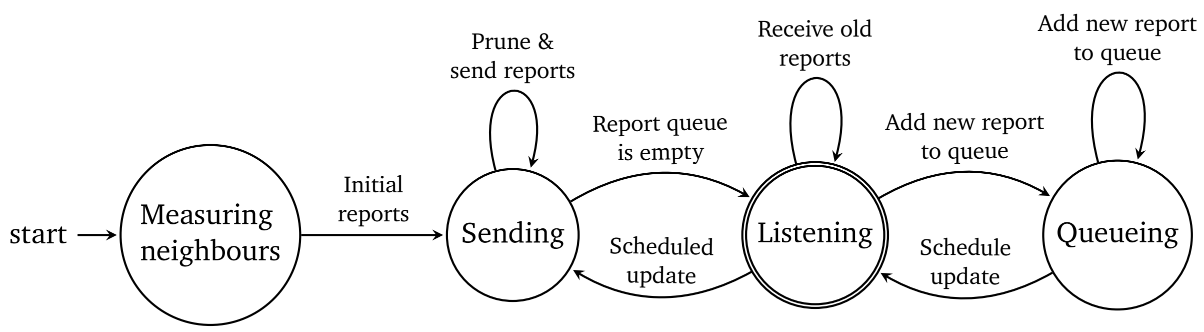


Figure 6.5.: A high-level deterministic finite automata of the staged flooding model.

The design is quite similar to the blind flooding model in section 6.3, but utilises as mentioned above a queue and a scheduled update time. Note that the steps "Prune & send reports" involve executing all of the enabled pruners for the queue of reports.

6.5. Projective & Sympathetic Flooding

This section describes *two* different models for the experiments with the same core implementation underneath the hood, but with different and crucial extensions to its dissemination mechanism.

The *staged sympathetic flooding* uses the core implementation of staged flooding in section 6.4 together with an implementation of sympathetic pruning as defined in definition 5.3.4.

The high-level state machine in fig. 6.5 in Staged Flooding Model represents these two models as well. The differences are found within each of the additional pruning mechanisms.

Despite the fact that these two models implement projective and sympathetic pruning from definitions 5.3.3 and 5.3.4 with the basis in the staged flooding model, the mechanisms can be employed for approaches including push-based gossiping (definition 2.2.7).

Simulation Features

The projective and sympathetic staged flooding models offer the same simulation feature parameters as detailed in table 6.4 on page 77. In addition the models implement the following feature detailed in table 6.5.

Remark 6.5.1 (Values of `pruneEmpathic`) The values `off`, `on`, and `extended` in table 6.5 respectively refer to staged flooding, staged flooding with projective pruning, and staged flooding with sympathetic pruning, but often shortened to the staged flooding model, projective model and sympathetic model.

Projective & Sympathetic Pruning

Both the projective and sympathetic pruners implement a pruner that we dynamically add to each node at the beginning of the

Table 6.5.: Showing the actual OMNeT++ simulation parameters and their respective values for the two different models implementations.

One model cannot have both projective and sympathetic pruning enabled. The simulation parameter `pruneEmpathic` allows to toggle between `off`, `on` and `extended`.

Feature	Variable	Value per Model	
		Projective	Sympathetic
Empathic Pruning	`pruneEmpathic`	`on`	`extended`

simulation.

In the pruning mechanisms they hook into the following points in the simulation:

`learnArrive()` At arrival of a message that contains hitherto unseen neighbour report

`prune()` When pruning a potential message of reports for a specific neighbour

`updateComplete()` When a scheduled update has been completed and all reports have been sent to their destinations

The models also hook onto `simulationComplete()` in order to submit statistics for how many unique nodes was accepted over the course of the simulation, and how many of those were superfluous nodes.

Furthermore, both of the mechanisms take the order of events into account when ensuring information is forwarded regardless of when empathic neighbours come online with their initial report.

We make the assumption that we do not know which of our neighbours are empathic nodes. While we would know in ResFi, we do not know until we receive a report from them in the simulation. More importantly we cannot perform minmax clustering with their node as the *seed node* without receiving their initial report. Thus we must queue potential reports even for those nodes we have not received information from yet.

The overhead of storing reports for nodes even after having scheduled an update is not considered problematic as it only requires storing a map or list of pointers to the vertices in the graph for each neighbour, or vice versa.

Projective Pruning

The implementation for projective pruning from definition 5.3.3 implements a pruner that 1. creates a queue for each queued report received where neighbours are inserted, 2. creates a set of useful nodes for our current node only, 3. updates the set of usefuls when the clustering changes.[7]

7: See definition 5.2.1 for more on the set of usefuls.

Resource Consumption The list of useful information is a map of sets mapping each queued report to a set of neighbour candidates. Storing the set of usefuls locally can be replaced by only retaining the set temporarily.

Sympathetic Pruning

The implementation for sympathetic pruning from definition 5.3.4 implements a pruner that 1. creates a queue for each queued report received where neighbours are inserted, 2. creates a map of neighbours identifiers to sets of useful nodes, 3. updates each neighbour's set of usefuls when a neighbour report from a useful node is received.

This means that each neighbour has its own set of usefuls calculated, which we the pruner uses in order to evaluate whether to retain or delete a report from the neighbour's queue.

Resource Consumption Compared to the projective pruner there is a much higher amount of information retain in memory throughout the simulation. This is because the map of useful information is a map of each neighbour to their own set of usefuls, although this can be reduced to a map of each neighbouring *cluster*.[8]

8: We believe multiple neighbours might be in the same cluster if the connectedness is high enough, as it increases the chances that neighbouring nodes are each other's neighbours as well.

However, the set of usefuls is replaced in its entirety for each neighbour when we update it.

Part III.

Results & Conclusion

Results and Discussion 7.

In this penultimate chapter we look closer at that which remains. We introduce results from our dynamically generated mesh topology in the first section where we primarily compare blind and staged flooding without empathic pruning, before moving to our results from the static topology used in Rønning [10] in the subsequent sections.

7.1. Blind & Staged Flooding

First we wish to compare the differences in utilising the blind and staged flooding. See fig. 7.1. We see that the number of reports received increases for the blind flooding model despite using TTL. As the number of reports received increases as the network increases, we have an indication that blind flooding is not tenable for use in EmpathicWiFi per our requirements, as this result suggests that reports move further away than acceptable.

Number of Reports Received

See the number of reports received in figs. 7.1, 7.3 and 7.4 to compare how the blind flooding and staged flooding differ as the values of alpha and network density changes. Unlike the blind flooding model which utilises the TTL field, staged flooding *without* empathic pruning has no control mechanisms except for discarding already known information.

First, in figs. 7.3 and 7.4 note how the number of reports received are for the majority of results not affected by alpha, but instead only impacted by the number of nodes. This supports the theory that both blind and staged flooding cannot be used in its simple form, unless it is enhanced with projective or sympathetic pruning.

Both models are affected by the connectedness in the network, as we assumed would be the case with a model based on flooding and without an optimal flooding algorithm in place. Yet, the number of received reports does not change significantly for the larger values of alpha as we increase the network size without increasing the density.

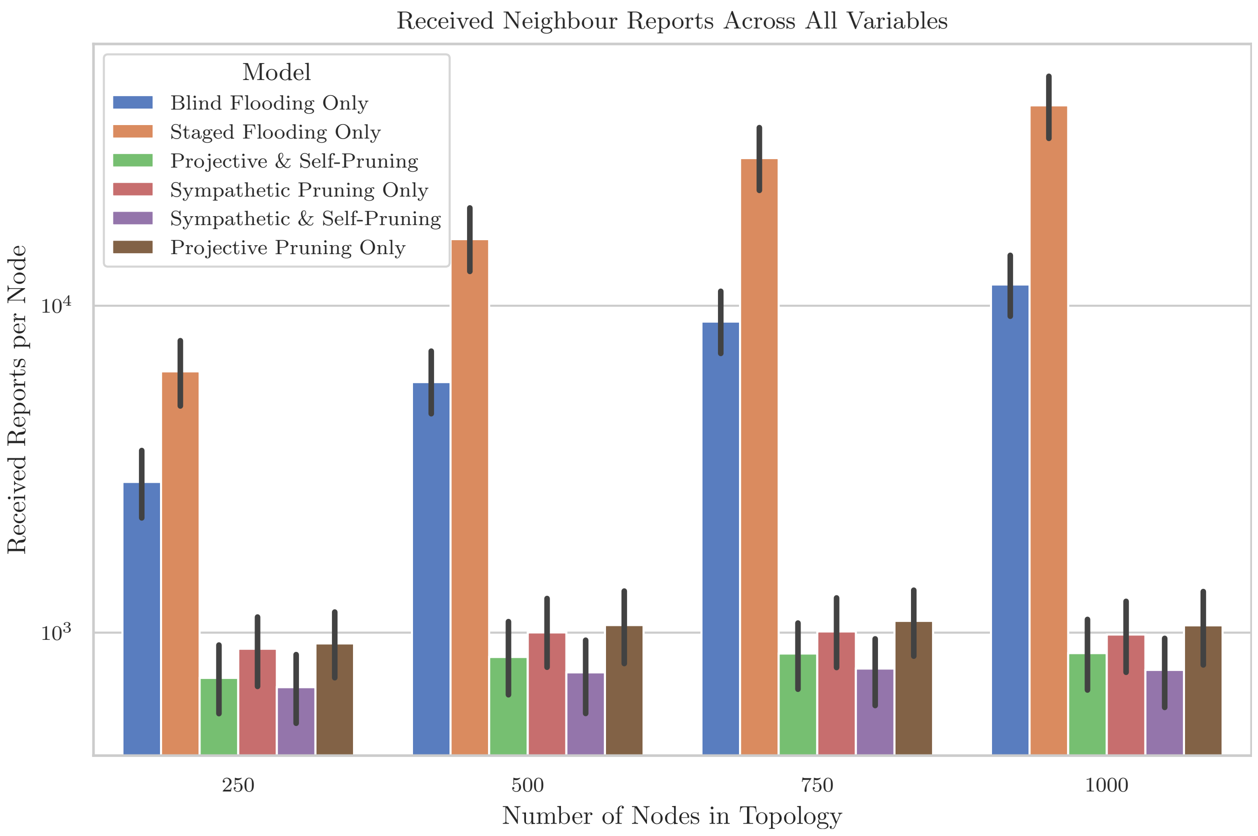


Figure 7.1.: The number of mean reports received per node across all variables, plotted logarithmically. This includes simulations for $a = 25, 50, 75, 100$, as well as multiple different values of neighbour density.

This is visible in fig. 7.2 in the first part of blind flooding. We can see that number of reports increases from $\alpha = 10$ until $\alpha = 20$, at which point it reaches a plateau. The small increase that is there can be explained by the set of additional nodes that now, rather than only having an edge inwards into the network, are surrounded by new nodes in a larger network. In other words, the increase in received reports can be attributed to all growth of all nodes with a big increase in edges due to the increased size. However, more work is required in order to prove that this increase is only due to the increased connectedness of border nodes. We will not categorically claim that this cannot be.

Impact by Change of Alpha

As shown in fig. 7.2 there are differences in how much the values of α impact the number of reports. We can observe that the blind flooding model is not impacted by different cluster sizes after $\alpha \approx> 20$, despite ceasing communication once a cluster has been reached.

Initially this was thought to be due to a lack of TTL, but the blind flooding model in section 6.3 is implemented with TTL in each message passed.

The broadcast messages in blind flooding are rebroadcast to other nodes, and the next nodes, and so on, before they eventually should cease. While the TTL field will stop broadcasting, it only does so for sufficiently small values of α and subsequently TTL. It seems our generated topologies are too dense to prevent a message from reaching all locations within α jumps.

The behaviour for staged flooding is different. Each increase of alpha results in a linear or near linear increase of received reports across all nodes, as seen in the changes in the columns with sufficiently high number of nodes in fig. 7.3 on 89.

The behaviour is consistent between blind and staged flooding models, until they reach their max as we can see in fig. 7.2.

Clustering Times

Figure 7.6 show how the time spent clustering is primarily affected by the cluster size α. Two other things are of particular note. First, the time is not particularly impacted by the increase in α; judging by the increase in time for convergence we could increase the value to 200 and the primary differentiator would be the length of the convergence for the higher values.

Secondly, the *projective* pruner seems to have a marginal mean improvement over the sympathic pruner. This could be due to the fact that it disseminates more information, thus also disseminating more useful information at a earlier time. That being said, the difference is hard to visualise here and we cannot claim that it is significant.

It is important to emphasise that the graph only contains a mean of the different empathic models, and that we have not included a proper confidence interval for this graph.

Impact by change of density

The impact on the number of messages is particularly prevalent when the `range` parameter is evaluated. This section uses a specific set of mesh network topologies where the amount of edges is $|V| = 250$ and the cluster size is $\alpha = 50$.

What causes the high number of reports? A caveat to any graphs illustrating the growth in traffic is that the actual number of reports also increase with the density. As the range or density of nodes increases so does the number of values per node that needs to be shared. Both the number of reports and the number of transmissions are affected by the connectedness of the graph.

The fig. 7.7 shows the number of neighbour reports sent and received relative to each node's count of neighbours. This illustrates how the relative amount of traffic increases with the density, but it also shows how some approaches fare better than others in such a scenario.

Firstly, we can see that the self-pruning variants fare better, but watch out. Notice in fig. 7.7 that staged flooding alone with self-pruning performs better to begin with for very sparse graphs. This changes when the average neighbour density approaches four, at which point projective pruning henceforth has a lower value of received reports.

This highlights the one of the benefits of self-pruning for sparse topologies where multiple paths to different nodes are less prominent.

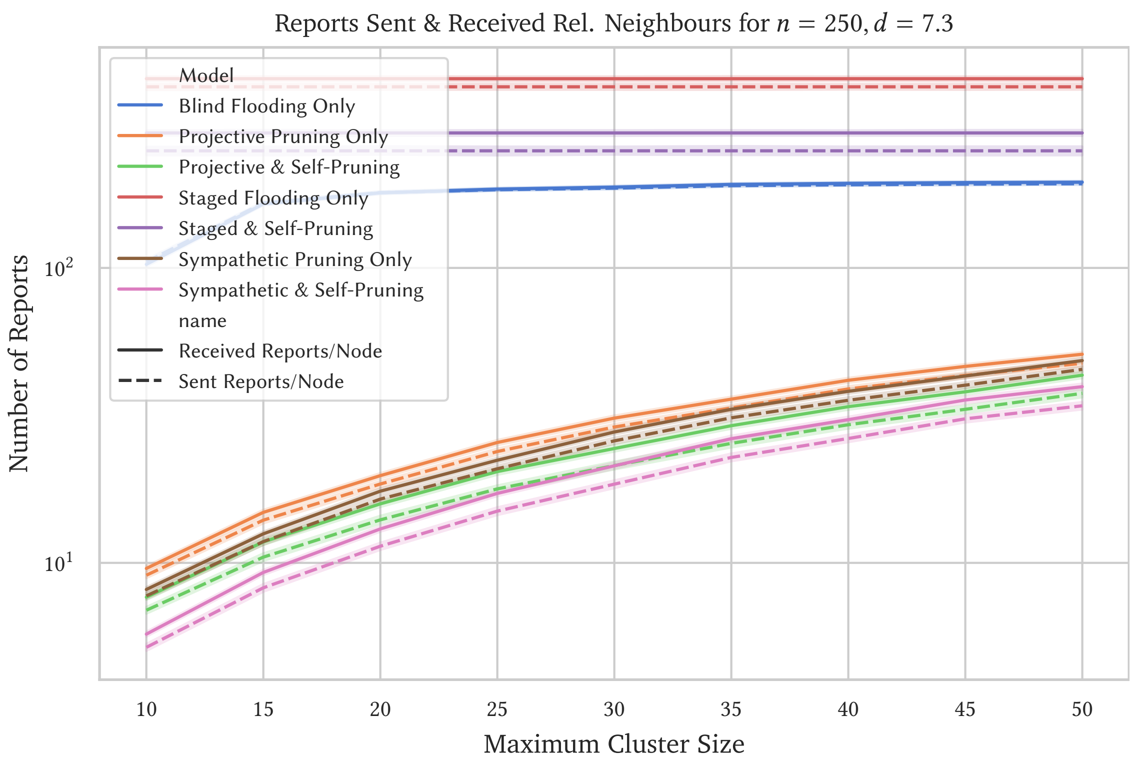


Figure 7.2.: Logarithmic graph over effect on the ratio of the total number of sent and received reports relative to value of $alpha$ for a fixed size of nodes $n = 250$, and one set density.

The graph illustrates the growth in reports as we gradually increase the size of the clusters.

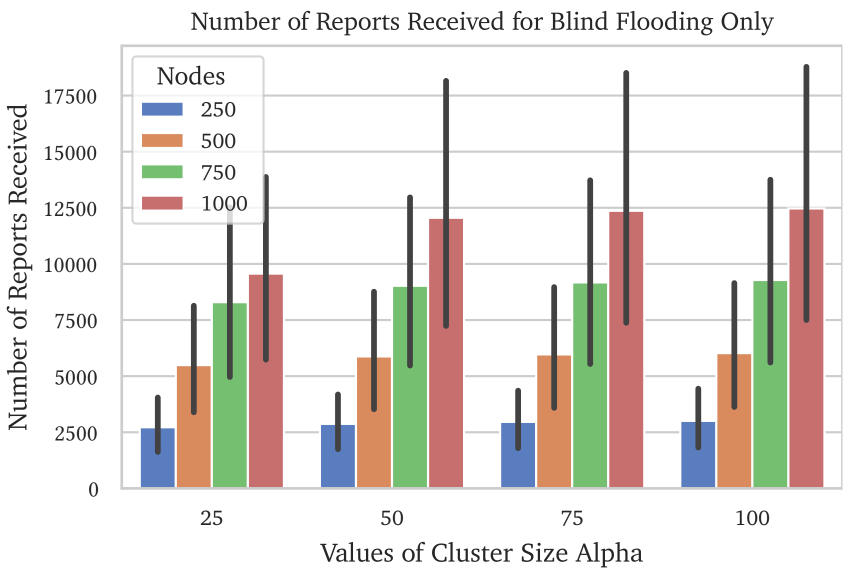


Figure 7.3.: Graph showing the impact on the total number of received reports for blind flooding only.

It indicates that the growth in reports received is a consequence of network size rather than cluster size. It also highlights that when the cluster size is sufficiently small and the network large enough it is possible to see that the TTL positively limits the number of reports, as we see for the bar where $\alpha = 25$ and the number of nodes is 1000.

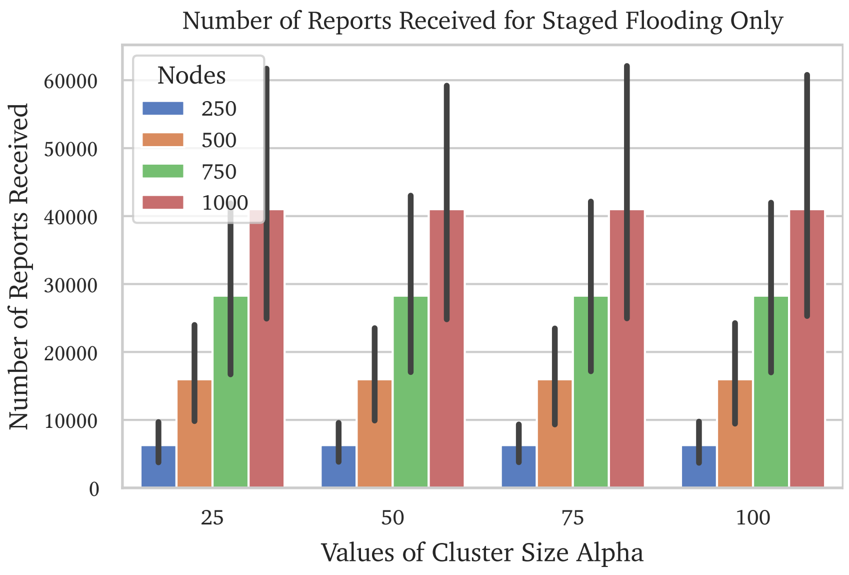


Figure 7.4.: Graph showing the impact on the total number of received reports for staged flooding only.

It shows that the lack of TTL makes blind flooding much better in terms of overall information transferred when we do not use empathic pruning, while also indicating that the growth of reports is a consequence of network size rather than cluster size.

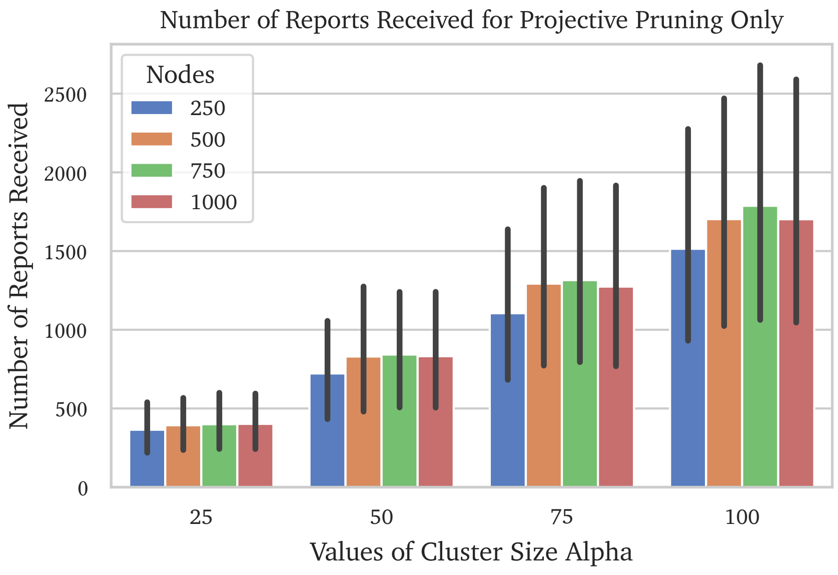


Figure 7.5.: Graph showing the impact on the total number of received reports for projective pruning only.

It indicates that the number of reports is primarily a consequence of cluster size, not network size.

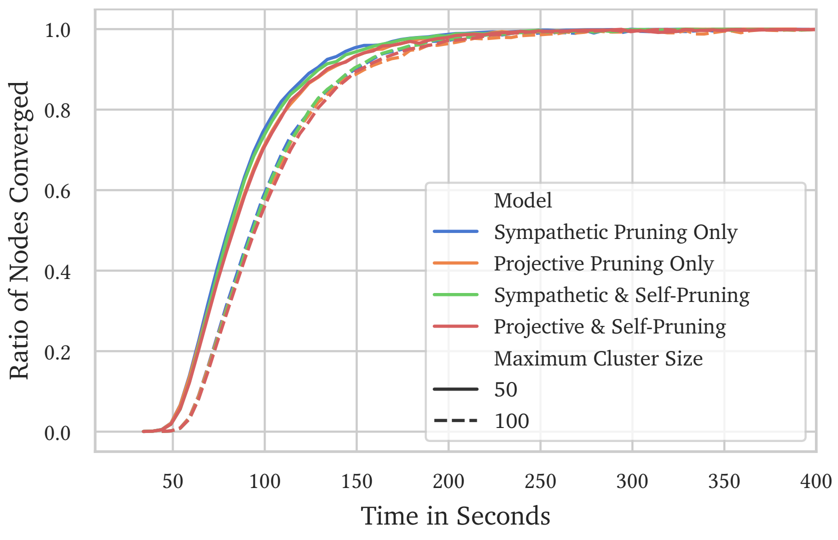


Figure 7.6.: Share of nodes that completed clustering over time categorised by time and values of α

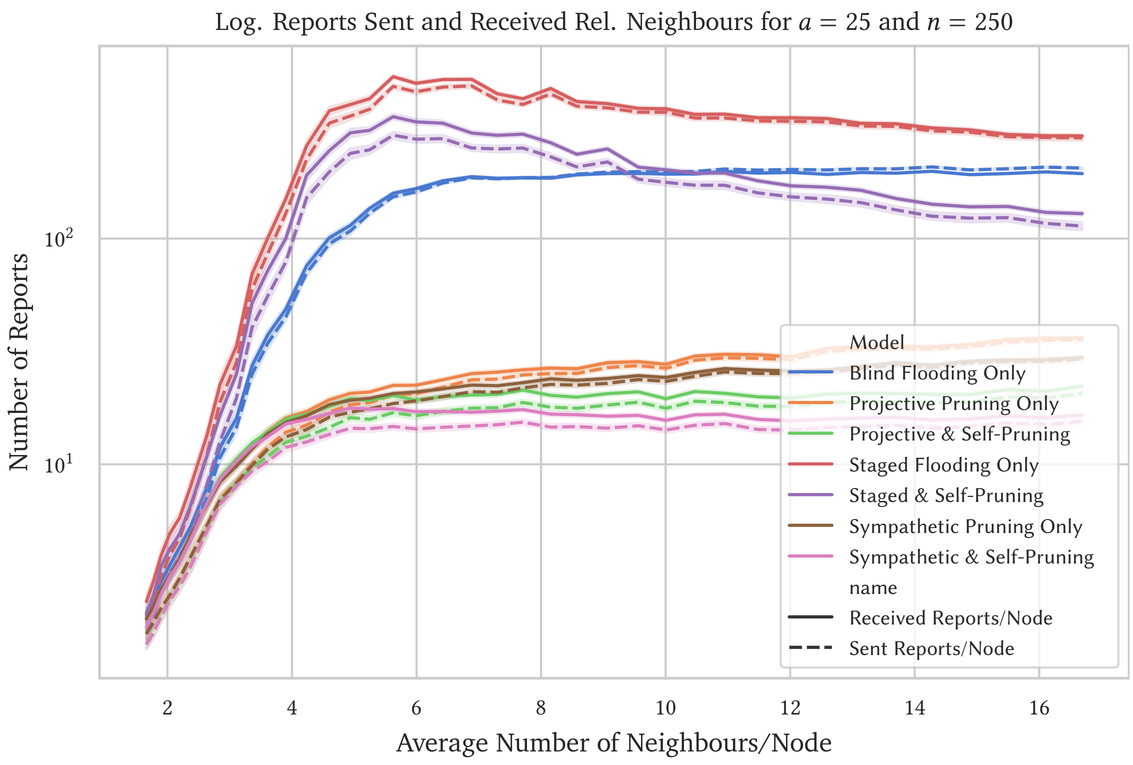


Figure 7.7.: Logarithmic graph over effect on the ratio of the total number of sent and received reports relative to the number of neighbours each node has.

The graph illustrates the growth in reports as the density of a network increases without increasing the complexity of the algorithm while taking into account the increased observations by each empathic node.

7.2. Relationship Between Traffic & Pruning

There are two different kinds of traffic in the results. In our results we cannot separate them, but we can point to them and explain how it impacts the results differently for the different types of pruning we have tested.

In order to explain the difference we will start with the key takeaway in terms of traffic in the networks.

Traffic In $n = 2000$ & $\alpha = 100$

Figure 7.15 is the key figure here. The bar figure highlights that the largest reduction in traffic sent is by utilising sympathic pruning and self-pruning in tandem. It shows that across the five static topologies which we ran five times each with different seeds, the total amount of traffic used in the network ranged between approximately 700 MB and 1400 MB.

Firstly, we believe this is a significant difference in terms of overall traffic between nodes in a simulation where all nodes still reach their clustering, but transfer less.

We do not compare the number of reports in fig. 7.15 to an implementation which only utilises staged flooding.

We will discuss the balance between projective and sympathetic pruning later, but a clear takeaway is that reducing redundant traffic is important.

Fewer Discovered Nodes is More

The first part of acknowledging the distinct difference in *how* we send less information is by asking what contributes to more information being sent. Neighbour reports are the key pieces of information that nodes need. If we can reduce the number of neighbour reports we pass from node to node, we reduce the total traffic both in terms of reports and sum of bytes.

In table 7.4 we can see the effect of both fundamental projective pruning (definition 5.3.3) and the sympathetic pruning (definition 5.3.4). By performing clustering on behalf of neighbours in sympathetic pruning we further reduce the amount of traffic. Sympathetic pruning alone yields an approx. 22% or 17% reduction in mean traffic depending on whether or not self-pruning is enabled.

How Come The Big Difference?

Why is the difference between the two so noticeably significant? A cluster of nodes will, as the cluster size increases, have more nodes within the partition that border to a majority of cluster members rather than other members. Any cluster member requires the same set of information in order to validate, and as such we should more often than not need the same information that our neighbours need.

The answer might lie in the relationship between the maximum cluster size α and the connectedness of the graph. If the connectedness in the graph is substantially high the majority of neighbouring nodes may not be members of the same cluster. Density of networks must, if this is the case, be considered when establishing which value of α to use.

An indication of this is present in fig. 7.8, where we can see the increase in reports for the empathic models as we gradually increment the values of α from 10 to 50.

A caveat to remember is that the order of information received impacts the results of the minmax clustering. It might be that any additional transmissions of information that occur due to projective pruning temporarily become part of the neighbouring node's cluster. Such an event could lead to multiple forwarded reports that were never necessary to begin with.[1]

1: The density in the graph further exacerbates the transmission of redundant information as self-pruning alone cannot eliminate two-hop neighbours. The combination of redundant and duplicate information may lead to an increased difference between the two models.

This underlines the importance of optimising the amount of information we forward, as any redundant information may yield multiples of duplicate traffic.

Distribution of Discovered Nodes

Let us examine fig. 7.9. The bivariate distribution per-node of discovered nodes[2] compared to the amount of payload-only kilobytes sent *and* received per node for the 0th static topology can tell us more.

2: See definition 3.3.1

The Y-axis representing discovered nodes allows us to see the distribution of nodes in the simulation with respect to how many nodes they had discovered by the end of the simulation. While $\alpha = 100$ we can see the number of discovered nodes going above 300 and 400 for a share of sympathetic and projective nodes respectively. The distribution on the Y-axis illustrates that the amount of knowledge gained of the network goes far beyond what is needed in order to confirm a cluster.

Remark 7.2.1 (Colourmap distribution) The bivariate distribution in figs. 7.9 and 7.12 has two colourmaps that each represent the distributions of their respective pruning mechanisms relative to themselves.

This means that information such as how narrow the histogram for discovered nodes is across the models must be evaluated in fig. 7.11.

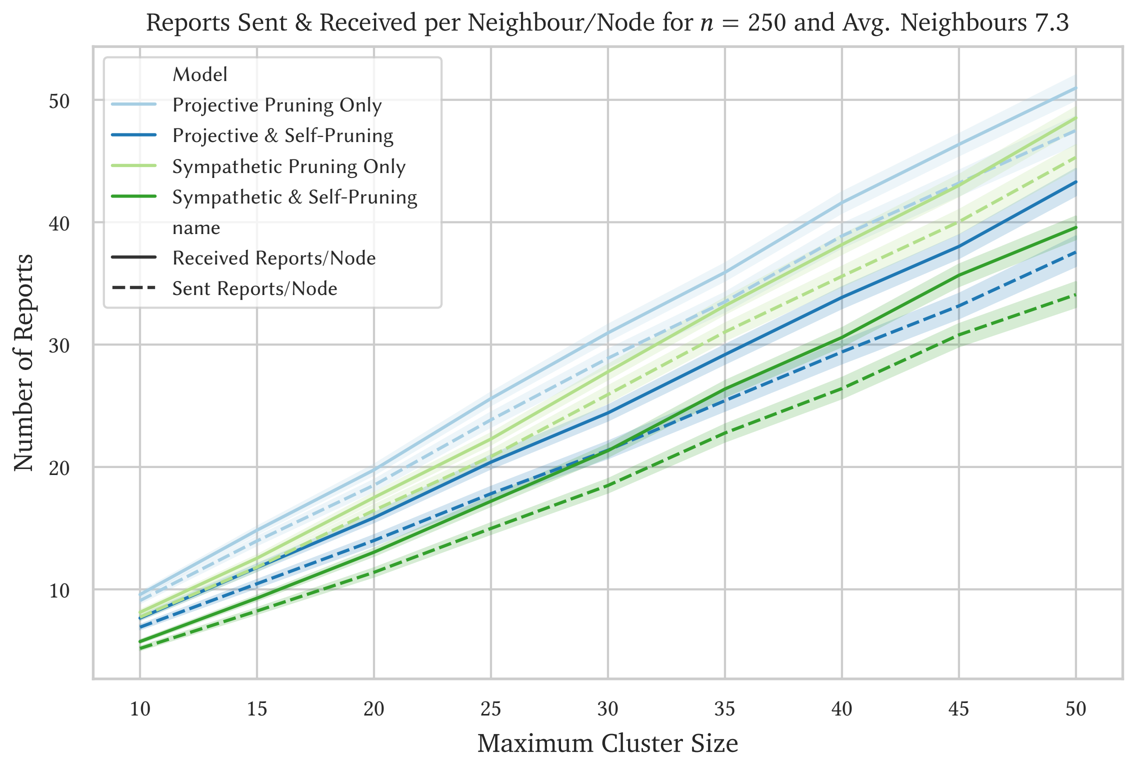


Figure 7.8.: Graph showing the effect on the total number of sent and received reports relative to value of $alpha$ for a fixed size of nodes $n = 250$ with average neighbour density at $7, 3$ for the empathic models.

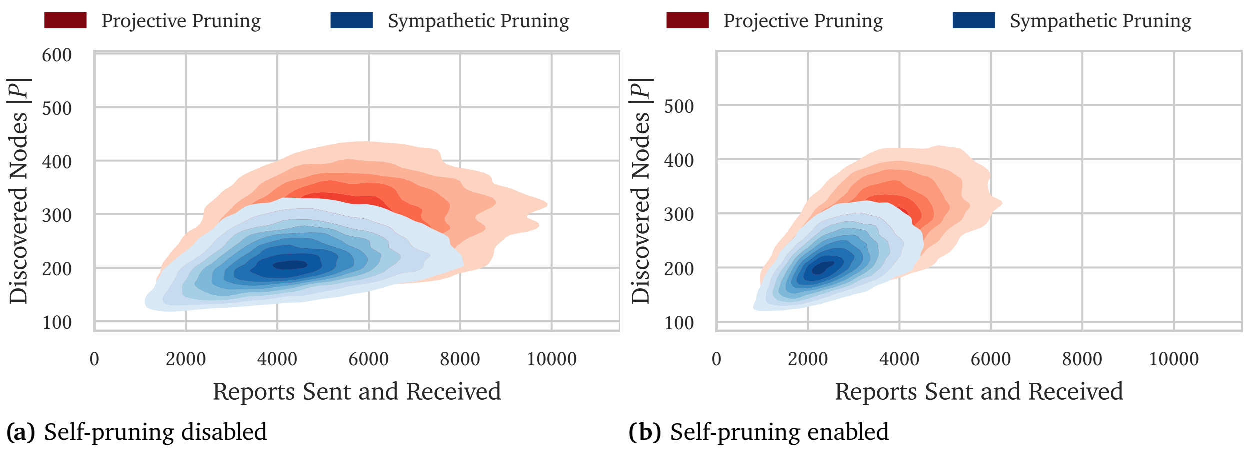
The graph illustrates the growth in reports as we gradually increase the size of the clusters. The distance between the two empathic pruning mechanisms grows gradually, becoming relatively larger to one another with the X-axis.

The addition of self-pruning in fig. 7.10b tightens both of the distributions of unique nodes, making the overall illustration thinner. This highlights that the number of unique nodes, although we have already seen this, is not affected by self-pruning, but by our own decisions of what information is useful to neighbours in EmpathicWiFi.

We can see the same in fig. 7.12, but we will not discuss traffic itself any more in this section.

2D Distribution

Lastly, we see that the distribution of known nodes for the different categories in fig. 7.11. The distribution shows sympathetic pruning contributing to a significant reduction in the total number of uniquely discovered nodes across the duration of the simulation. Its distribution, whether self-pruning is enabled or disabled, is more similar to that of a normalised distribution. In contrast to the projective pruning where we identify more of a long-tail phenomenon for both variants of self-pruning.



(a) Self-pruning disabled **(b)** Self-pruning enabled

Figure 7.9.: Distribution of Unique Nodes Discovered and Sum of Reports Sent & Received Per Node in 0th Topology. In fig. 7.10a the bivariate distribution is illustrated for a simulation with self-pruning disabled, whereas self-pruning is enabled in fig. 7.10b.

It is largely in line with the earlier comparisons between known nodes at the end of the simulation and the amount of traffic per node. It seems that the 3α average that Rønning [10] establishes with projective pruning is possible for us to replicate based on the distribution in fig. 7.11.

Sympathetic pruning yields a ratio of α to discovered nodes at $\approx 2\alpha$.

Zero-nodes in Static Topologies

The aggregated values for how many discovered nodes the 0th node has in the various static topologies are shown in table 7.1. This is included as a comparison to the results in Rønning [10].

Distribution of Reports & Bytes Transferred

When discerning the difference that comes from enabling self-pruning we must evaluate the differences between figs. 7.10a and 7.10b.[3]

The X-axis representing the distribution of total number of reports both received and sent shows us how much of a problem duplicate transmissions are in the simulation. Without self-pruning enabled we can see in fig. 7.10a that the core distribution of sympathetic pruning[4] seems to discover approx. 200 nodes, yet receives and sends a total of slightly above 4000 reports.

If we assume that the standard deviation of the number of neighbours per node is low, we can assume that each node roughly transmits and sends equally much, as they are equally connected.[5]

3: The same descriptions and conclusions apply to figs. 7.13a and 7.13b, as the sum of traffic is directly caused by sending reports in the simulation.

4: In blue gradient.

5: Due to the probabilistic order of events from the random intervals some nodes will naturally send a lot more than others, and some may barely send anything.

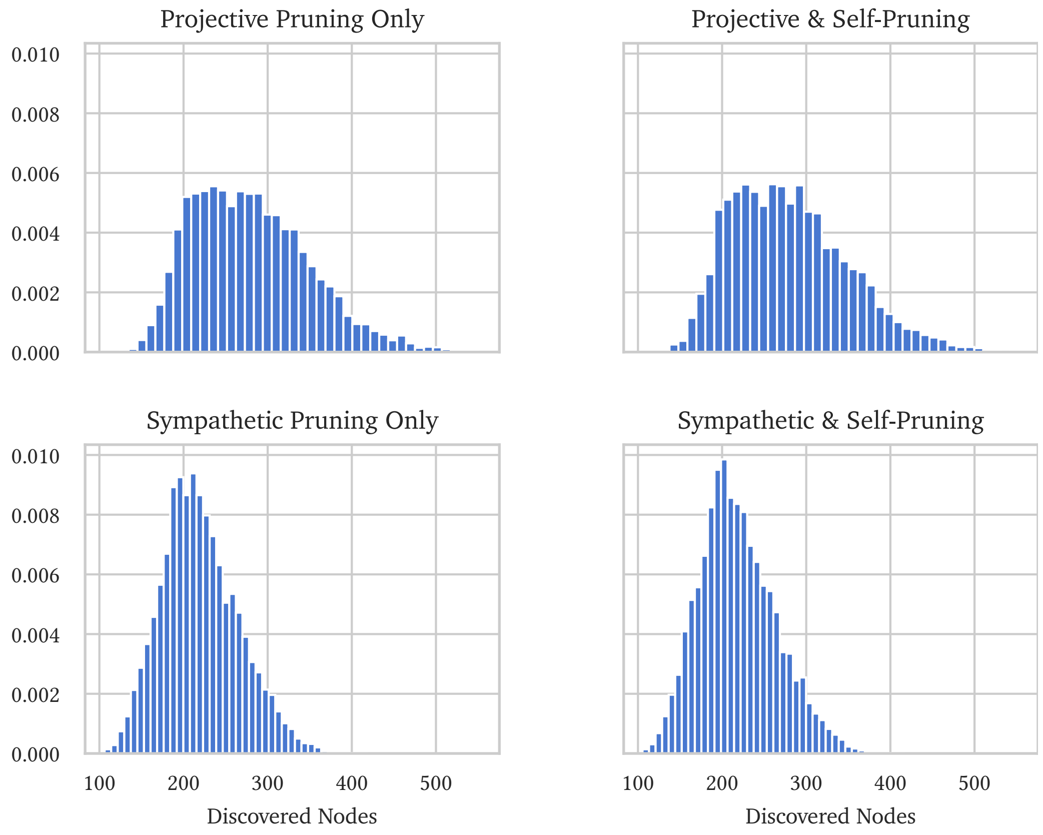
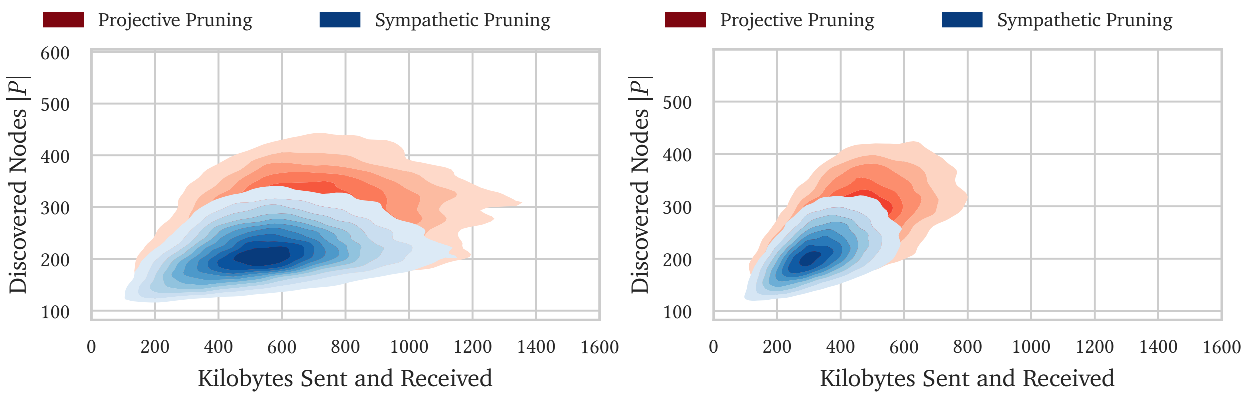


Figure 7.11.: Distribution of known nodes at the end of the simulation over all sets and all nodes

Table 7.1.: The mean, minimum and maximum of the th0 node in all five static topologies

		value		
		mean	min	max
set	experiment			
0	self-pruning off, empathic extended	266	241	287
	self-pruning off, empathic on	312	288	347
	self-pruning on, empathic extended	264	246	286
	self-pruning on, empathic on	302	285	323
1	self-pruning off, empathic extended	399	358	430
	self-pruning off, empathic on	481	469	489
	self-pruning on, empathic extended	403	362	445
	self-pruning on, empathic on	487	470	500
2	self-pruning off, empathic extended	191	176	223
	self-pruning off, empathic on	285	275	300
	self-pruning on, empathic extended	197	178	231
	self-pruning on, empathic on	309	292	323
3	self-pruning off, empathic extended	204	185	247
	self-pruning off, empathic on	381	367	397
	self-pruning on, empathic extended	193	179	210
	self-pruning on, empathic on	378	364	390
4	self-pruning off, empathic extended	169	146	192
	self-pruning off, empathic on	300	287	317
	self-pruning on, empathic extended	165	147	189
	self-pruning on, empathic on	310	297	327



(a) Self-pruning disabled **(b)** Self-pruning enabled

Figure 7.12.: Distribution of Unique Nodes Discovered and Sum of Payload Traffic Sent & Received Per Node in 0^{th} Topology. In fig. 7.13a the bivariate distribution is illustrated for a simulation with self-pruning disabled, whereas self-pruning is enabled in fig. 7.13b.

This results in a rate of duplicates of an order of magnitude with 10× the number of received reports to what a node needs.

Enabling self-pruning in fig. 7.10b significantly reduces the number of duplicate transmissions by placing the core of sympathetic pruning right above 2000 reports sent and received. When we make the same assumption as above we have an approximate rate of duplication at 5×.

The reduction in the median amount of traffic is almost at 50% when we look at figs. 7.13a and 7.13b.

2D Distribution

It is difficult to discern the distribution for projective pruning, except that it certainly has higher variance, so we turn to fig. 7.14 which shows the distribution of the sum of received reports per node as a histogram. It is clear that enabling self-pruning significantly tightens both the projective and sympathetic models, moving the median for projective pruning with self-pruning enabled to under 2000.

Our chaotic simulation with staged flooding and sympathetic pruning yields a lower distribution of discovered nodes, as well as gathering the amount of data transferred per node.

Here in table 7.2 we see that the 95% percentile for the number of uniquely known nodes across *all topologies* is reduced from approx. 400 to 300 by utilising sympathetic pruning. This constitutes a marked decrease for the vast majority of nodes in the topology, with 3α as the highest value for the 95^{th} percentile of sympathetic pruning nodes.

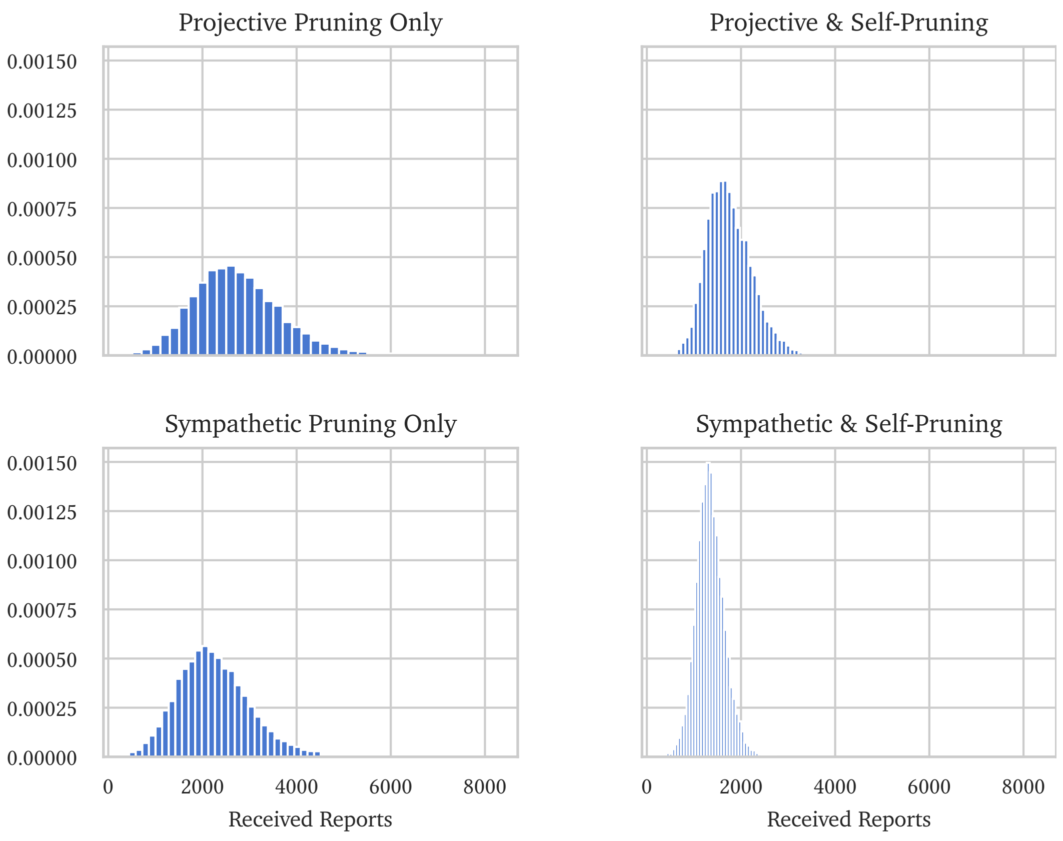


Figure 7.14.: Distribution of the number of received reports at the end of the simulation over all sets and all nodes

Table 7.2.: 95% Percentile for the Discovered Nodes for All Topologies.

The number in the upper right corner is the percentile as share of 1, 0.95.

experiment	0.95
Sympathetic Pruning Only	302.00
Sympathetic & Self-Pruning	300.21

Reducing Duplicate Transmissions

Figure 7.15 shows how self-pruning as defined in definition 5.3.5 helps reduce redundant information transfers, even when coupled with the neighbours' based empathic pruning. Note that the difference in traffic is significantly lower when coupled with sending all nodes that we consider to be useful.[6]

6: A caveat for the self-pruner is that it currently is too simple to ensure that it never prunes content it should not, but it does not seem to pose a problem in sufficiently dense topologies such as the 0^{th} static topology. By specifically validating if the node sending the reports are in the same cluster, it can avoid eliminating reports where we cannot know that they have sent it to the neighbours. This should eliminate the difference in number of unique nodes for self-pruning configurations at a small traffic increase cost.

We see that the use of self-pruning contributes to approx. 36% or 40% reduction in the mean traffic per node in table 7.5, as well as a multitude of other changes in tables 7.4 to 7.6.

A similar result is visible in table 7.3 where we see the 95th percentile in terms of traffic for the projective and sympathetic models, as well as self-pruning.

experiment	0.95
Projective Pruning Only	1166.092500
Projective & Self-Pruning	719.818750
Sympathetic Pruning Only	984.176602
Sympathetic & Self-Pruning	545.150391

Table 7.3.: 95% Percentile for Sum Kilobytes Sent & Received Per Node for All Topologies.

The number in the upper right corner is the percentile as share of 1, 0.95.

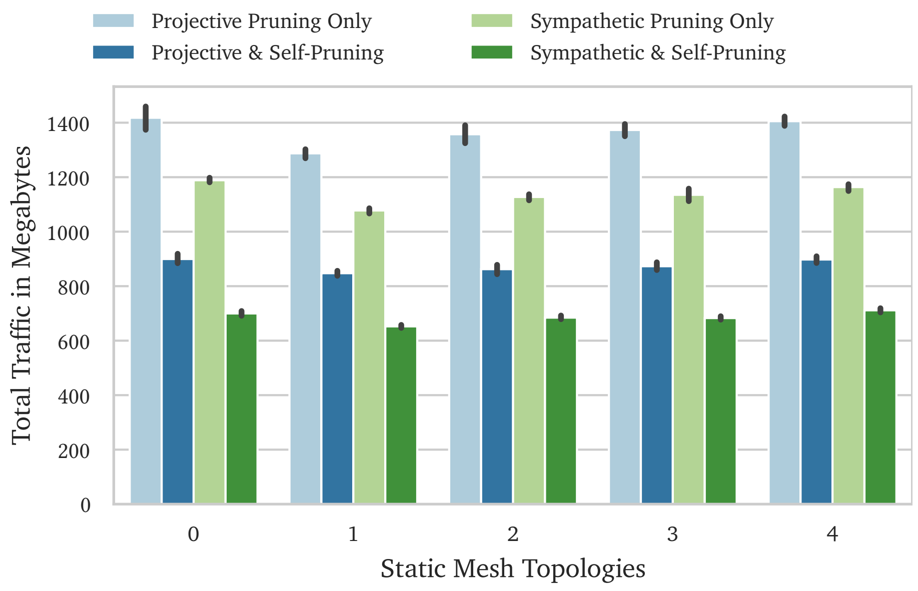


Figure 7.15.: Total Traffic Sent/Received in Static Topologies for $n = 2000$ and $a = 100$ Per Pruning Mechanism

Is It Too Much Traffic?

If the amount of traffic seems high for the information elements in question, we must take into consideration that the 2000 nodes in the topologies have a high degree of connectedness. Additionally, we suggest that the results we have fit with the discussion in section 5.4, where we hypothesise possible communication numbers for only a few nodes in a graph.

Based on the five topologies used, it seems that

$$\operatorname{avg}(|\mathrm{N}(V_i)|) \approx 15$$

Thus, the average degree or number of neighbours[7] in the graph is 15.

7: The average degree in a graph G for a vertex V is the average number of edges between vertices in the graph. In other words, the average degree is equal to the average number of neighbours each node has.

Note then that if the 14 example neighbours of V_i were to forward the same set to other members, say 11 or 12 other neighbours, then that is another 100, or 200 in terms of transmitted and received, added for every neighbour they forward it to.

Thus, we believe the numbers to be credible and understandable when the effect of network density is taken into consideration.

Table 7.4.: The aggregate values for traffic in kilobytes and number of discovered unique nodes in 0th static topology

		Discovered Nodes				Traffic in Kilobytes			
		mean	std	min	max	mean	std	min	max
Empathic	Self								
Projective	Off	279,9	70,2	108,0	579,0	700,6	279,3	39,6	2548,0
	On	280,1	70,1	119,0	574,0	448,6	167,3	43,9	1410,0
Sympathetic	Off	219,9	48,9	100,0	460,0	583,0	233,4	30,8	2005,1
	On	218,8	49,0	101,0	455,0	351,3	125,6	25,4	1220,2

Table 7.5.: The changes in traffic and number of nodes for self-pruning in 0th topology.

The table highlights the difference in percentage when going from self-pruning disabled to enabled.

		Discovered Nodes				Traffic in Kilobytes			
		mean	std	min	max	mean	std	min	max
Empathic	Self								
Projective	Off	—	—	—	—	—	—	—	—
	On	0.1%	-0.1%	10.2%	-0.9%	-36.0%	-40.1%	10.9%	-44.7%
Sympathetic	Off	—	—	—	—	—	—	—	—
	On	-0.5%	0.1%	1.0%	-1.1%	-39.7%	-46.2%	-17.6%	-39.1%

The high number of duplicate transmissions of reports over multiple hops underscore the need for a further optimisation of any flood-based approach, which exists regardless of further optimisations to the set of information we need to forward to neighbours.

As seen in the distribution of *only received* reports in fig. 7.14, together with the distribution of uniquely known nodes in fig. 7.9, the ratio of duplicated or redundant information is still quite high, but this must be taken viewed within the context of long hops of information.

Ways to avoid this include utilising *reduced broadcast*[8] , but this results in probabilistic flooding and adequate heuristics are required in order to have a high enough probability. This requires exploring how many devices that need to cooperate for the effects of empathic clustering and distribution channel allocation algorithms to have a significant enough positive effect on network metrics.

8: Use heuristics to reduce redundancies of the flooding by exploiting the node density and pruning information from paths.

Table 7.6.: Changes in traffic and number of nodes for types of empathic pruning in 0th topology.
The table highlights the difference in percentage when going from projective pruning to sympathetic pruning.

		Discovered Nodes				Traffic in Kilobytes			
		mean	std	min	max	mean	std	min	max
Self	Empathic								
Off	Sympathetic	—	—	—	—	—	—	—	—
	Projective	28.0%	43.1%	17.8%	26.2%	27.7%	33.2%	73.0%	15.6%
On	Sympathetic	—	—	—	—	—	—	—	—
	Projective	27.3%	43.4%	8.0%	25.9%	20.2%	19.7%	28.5%	27.1%

7.3. Applicability & Thoughts

In this section we briefly discuss the trade-offs between the various models from our work.

Blind Flooding

Blind flooding easily uses the least amount of memory and is arguably the least complex model of them all. Perhaps, however, its triviality and its traffic consumption prevents us from seriously considering it as an approach on its own. Broadcast storms are not acceptable in order to disseminate information as it considerably breaks with the requirement to not disturb the backhaul network per section 4.2.

With that said, we have not shown or provided results that highlight how blind flooding functions in an implementation where projective and sympathetic pruning are employed. Existing literature on the topic of the trade-off between latency and energy highlight, such as in Cheng, Niu, Luo *et al.* [40], indicate that our results would yield lower latency at the cost of an increase in the number of duplicate transmissions, and subsequently a decrease in efficiency. However, we have not tested this, and as such we cannot say to what extent a blind flooding model using projective or sympathetic pruning would be suitable or unsuitable when taking data caps, bandwidth or otherwise disturbance to the backhaul network into consideration.

Staged Flooding

The results from the earlier comparisons between blind and staged flooding support that making the trade-off between latency and communication costs can reduce network traffic consumption substantially.

While we have discussed how our approaches work with regard to the requirements in section 4.2, we have also confirmed that nodes can gather in clusters from scratch.

Furthermore, we have shown that in staged flooding we can still perform clustering within the given time intervals as shown in fig. 7.6, despite increasing latency.

The results as presented indicate that we can perform clustering in rather large networks without creating broadcast storms. Subgraphs within networks will overlap and neighbour reports will be transported beyond cluster boundaries, but not to a significant or otherwise disruptive degree based on our dynamic and static topology simulations.

The performance of staged flooding has not been closely evaluated, but its primary requirement is sufficient memory or storage space to retain the information we need in order to queue neighbour reports and schedule them for later forwarding.

Projective Pruning

A particular flaw in the flood-based experiments is the reliance on comparisons between blind and staged flooding without any form of pruning based on minmax clustering information. Since the introduction of minmax clustering in the EmpathicWiFi project there does not, in this author's opinion, seem to have been a point at which projective pruning was *not* meant to be utilised — regardless of its name or exact implementation.

Therefore, comparisons should be made primarily between projective and sympathetic pruning, when not discussing the interplay of projective pruning and staged flooding.

The trade-offs to projective pruning include the dissemination of more information than required, primarily when ratio of edges to cluster size α increases.

The primary benefits to projective pruning include its ease of implementation, especially when considering sets and information elements required.

Sympathetic Pruning

The trade-offs to sympathetic pruning include a more complicated—although not complex—implementation wherein multiple sets, maps or lists are required to perform the pruning.

The primary benefits to sympathetic pruning include its significant reduction of forwarded information, contributing to reduced traffic and memory requirements for nodes in terms of their node knowledge, as well as reducing the issues from utilising non-optimised flooding.

Self-pruning & Other Optimisations

The role of self-pruning in our results is to cast a shadow over the duplicate transmissions in our results, such as in fig. 7.14. There are many other protocols available that further reduce the forwarding of duplicates, and any real-life implementation of EmpathicWiFi owes to itself based on the results from self-pruning alone to further investigate these alternatives.

Self-pruning, as introduced in the section where we find definition 5.3.5, is of particular interest because it is a simple mechanism built on a small amount of information. It fits with our assumption that nodes must first forward their own neighbour report to their neighbours before forwarding anything else.

Unfortunately self-pruning is, depending on the exact threshold for per-node traffic costs, perhaps too simple. Other variants of self-pruning, such as *improved self-pruning* increases complexity, but does not increase the amount of information required to reduce forwarding of nodes [69].

While implementation complexity may increase, we argue that this is a price worth paying in order to make the use of EmpathicWiFi less noticeable in a network. Especially if providing updated releases of EmpathicWiFi is difficult, then more time must be spent reducing excessive communication.

When all is said and done, it is worthwhile to take a moment to reflect that 1400 MB is not a lot when you divide it by 2000 per interval. A rough average of 700 kB per node seems acceptable, but we must take notice to also reduce the deviation between nodes and minimise the values of those nodes that transfer the most. Table 7.4 highlights that some nodes transfer more than 2 MB per interval in order to perform the clustering without self-pruning.

However, if you wish to perform the process from scratch, four times a day for a month, then the amount per month is $1400\,\text{MB} \cdot 30 \cdot 4 = 168\,\text{GB}$ for an average of 84 MB per node/month. Self-pruning with projective pruning reduces the total amount per interval down to 897 MB, which is 108 GB for an average of 54 MB per node/month. The choice of dissemination mechanisms further emphasise that there is a trade-off between cluster responsiveness and sum of data transferred.

Self-pruning improves the network experience for nodes. And while good, as long as the network is *primarily transferring topology information, the* information that nodes need in order to make better routing decisions, it is possible to optimise the dissemination beyond our current results.

7.4. Threats to Validity

Herein follows a list of aspects of particular concern to us. These may reduce the strength of our results and our conclusions.

1. The generated mesh topology uses a uniform random number generator, and as such does not result in topologies with a distribution more similar to that of the Poisson distribution.
2. The simulation does not attempt to simulate multiple clusterings occurring over a set period of time with fixed intervals between them. Because of this there are no evaluations of how only transmitting adjusted observations can be taken into account and affect later clustering in different ways.
3. The simulation makes an assumption for a undirected graph when dealing with nodes that are $\alpha + 1$ hops away. A node v is in some cases excluded from the clustering solely on the basis of its neighbour u, without getting v's neighbour report. This is amendable in the C++ clustering implementation, but is not fixed due to time constraints and that the difference in results would be a marginal increase in sent nodes. The relative differences and strengths should not change as a consequence of this.

Memory Requirements

There are different times at which we are interested in the memory usage, as well as the circumstances at which it occurs. Primarily we are interested in the peak memory consumption. How much we consume at a max is important in order to determine the suitability of any of the models for implementation on embedded devices, as the max consumption is the one while the device is performing the clustering and needs to have all required data in volatile memory rather than necessarily cached on disk.

With that said, the thesis does not attempt to evaluate a specific implementation for a particular operating system for embedded devices in use on APs. The exact memory requirements will depend on the implementation language, platform and the balance between performance and memory requirements. More importantly, the memory requirements will depend on whether an implementation is designed for memory scarcity or developer convenience.[9] Beyond the space complexity described for each of the pruning mechanisms we do not further assess suitability in terms of memory questions.

9: We have attempted to prioritise a combination of readability and code cleanliness, resulting in code with room for further optimisations. These evaluations are solely based on the author's own subjective opinion and assessment of the code within this work.

Conclusions and Future Work 8.

This last chapter presents ideas for future work and concludes this Master's thesis.

8.1. Future Work

In chapter 3 we established that the amount of literature done within this particular area of empathic RRM and distributed sensing thus far is comparatively small when we look at the amount of literature and work in the other fields, such as information dissemination in MANETs.

The following are a few, select issues of interest, as we find it challenging to limit the number of possible avenues that could be explored in future work.

Multi-strategy dissemination This work only explored a push-based dissemination approach in detail. How suitable is a dissemination strategy consisting of both pushing and pulling? At which point should a strategy change from pushing information to pulling, and how is it affected by the cluster size α? Similar work is done on this for other kinds of dissemination protocols [34].

Allowing Multi-hop Communication The work presented in this thesis is based on a process of repeated one-hop communication, which converges towards having disseminated the information.

Can the gathered unique nodes, if their information was coupled with their public keys, be utilised to allow devices to directly communicate with and push updated interference reports to nodes when memberships change? Could the application of this allow for small-world communication and reduce the amount of traffic?

While coupling this information with security keys introduces a challenge of balancing the cost of additional space with the benefit of jumping multiple hops, it seems like an possible avenue to further reduce traffic should higher mobility or larger clusters be a goal.

Delivering software for embedded devices How can systems tailored to residential users be efficiently and iteratively delivered to embedded devices?

Modularity with respect to radio software components such as *hostapd* on Linux, allowing third-party services to extend and improve upon existing networking devices that end-users own, is an issue that touches on both design and technology interoperability.

Various approaches to development on embedded devices are assumed to be highly heterogeneous and inconsistent, especially for consumer-facing devices such as wireless routers or pure APs. Wireless routers, although well-known and commonplace in homes, are not established platforms for application development by third-party developers. How can we distribute and deliver software updates to a heterogeneous base of embedded devices?

Criteria to calculate interference for Any *channel allocation algorithm* requires both short-term and long-term radio sensing information. The slow-moving physical landscape of wireless networks and devices need to be factored in when optimising for channel selection. However, the slowly changing information must be coupled with the immediate and reactive properties of dynamic usage patterns. Modern networks and modern usage patterns require a balance between the low activity times and peak network usage.

Radio sensing is an issue of its own in academic literature, with continued research and development being poured into it [7], [12], [13], [28], [71], [72]. The physical limitations or requirements of radio devices also come with balancing challenges requiring careful design of sensing frequency coupled with optimal performance for end-users on networks. Due to throughput-intensive applications, this balance can have a significant effect on the interference that can occur in overlapping wireless networks. Parts of these considerations are further examined by Eide [73].

How does a system like EmpathicWiFi calculate interference values to neighbours, how often should it calculate interference, and when should clusters react? Can a empathic cluster dynamically mitigate in a fully distributed manner when network usage changes?

8.2. Conclusions

The aim of this research was to identify and subsequently evaluate a simple and effective information dissemination system, with the purpose of sharing radio sensing data in primarily residential networks and successfully performing autonomous clustering.

Based on both qualitative and quantitative analyses of dissemination mechanisms and strategies, we have selected and evaluated a simple baseline flooding approach, and introduced a staged flood-delay mechanism combined with two distinct variants of minmax clustering-based pruning. In evaluating these different approaches, we have implemented numerous simulation models and gathered metrics from their execution.

By combining the clustering algorithm with an actively developed simulation framework, we have laid the foundation for further optimisation work, development and investigation. As part of our work, an initial C++ implementation of the minmax clustering was developed that may also contribute to future development of an EmpathicWiFi agent system for embedded devices.

The simplicity of the push-based approach invites further investigation by utilising other general flood-based optimisation techniques that have been discussed in related literature. The benefit of utilising and simulating results with self-pruning, a small optimisation, is that it clearly demonstrates that further optimisations should be evaluated. This is also supported by the ratio of unique discovered nodes to transmitted neighbour reports, which changes from projective pruning with self-pruning disabled to self-pruning enabled.

Contrary to approaches that solely use blind flooding, our work shows that it is possible to use a simple flooding mechanism with queued forwarding, with both the existing and new empathic pruning mechanisms for cluster sizes of at least 100 nodes. The difference in traffic transmitted between blind and staged flooding, supported by the latency and cost trade-off, indicates that using staged flooding allows for a further increase in cluster sizes if the metrics of blind flooding simulations were previously defining the former thresholds.

For all metrics, our proposed dissemination and pruning mechanisms perform better than blind and staged flooding without pruning, with the caveat of a varying trade-off in processing time, space, and implementation complexity. Efficient, quick and robust information dissemination is important for a heterogeneous distributed system, and is also possible to implement while focusing on simplicity.

Furthermore, we have shown that the sympathetic pruning is an improvement to the projective alternative, as it successfully reduces the amount of information disseminated for high-density topologies with a connectedness of $p > 15$, which far exceeds our hypothesis. This allows for higher values of cluster sizes α and subsequently, enables larger empathic clusters of distributed channel allocation cooperation with a per-system and per-node communication cost that can still remain within the upper bounds of projective pruning. The combination of self-pruning and sympathetic pruning yielded an almost 50% reduction in the total number of neighbour reports disseminated across the static simulation topologies, thereby reducing the total payload transferred from 1.4 GB to 0.7 GB.

While we have achieved parts of our goals, unanswered questions do remain. We have not clearly identified a single approach to information dissemination, as many other approaches remain applicable and theoretically suitable, even if at a higher cost of complexity. Furthermore, the strength of our results can be improved upon with larger simulations, an increase in simulation iterations, TTL for staged flooding, and varying types of topologies.

In conclusion, a simple and sufficiently efficient information dissemination approach that is based on the combination of a staged flooding approach and the properties of the minmax clustering algorithm is feasible, applicable, and recommended for the purposes of distributed sensing in residential networks, given our defined requirements.

Index

FTTC	Fibre to the Curb	41
TDMA	time-division multiple access	22
DFS	dynamic frequency selection	14
CSA-IE	Channel Switch Announcement	14
802.11g	IEEE 802.11g-2003 (also known as Wi-Fi 3)	
802.11n	IEEE 802.11n-2009 (also known as Wi-Fi 4)	
802.11ac	IEEE 802.11ac-2013 (also known as Wi-Fi 5)	
802.11s	IEEE 802.11s (mesh networking)	

Bibliography

[1] T. Barnett, S. Jain, U. Andra and T. Khurana, 'Visual Networking Index (VNI) Global and Americas/EMEAR Mobile Data Traffic Forecast, 2017–2022', Cisco Systems Inc., Mar. 2019.

[2] T. Maseng, *EmpathicWiFi*, Mar. 2019.

[3] Korean Communications Commission. (18th Jan. 2011). 'KCC prepares guidelines to minimize Wi-Fi interference', [Online]. Available: https://eng.kcc.go.kr/user.do?mode=view&page=E04010000&dc=E04010000&boardId=1058&cp=2&searchKey=ALL&searchVal=wi&boardSeq=30836 (visited on 29/06/2020).

[4] M. Achanta, 'Method and apparatus for least congested channel scan for wireless access points', U.S. Patent 20060072602A1, 6th Apr. 2006.

[5] A. Mishra, V. Brik, S. Banerjee, A. Srinivasan and W. Arbaugh, 'Client-driven channel management for wireless LANs', *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 10, no. 4, pp. 8–10, 1st Oct. 2006, ISSN: 1559-1662. DOI: 10.1145/1215976.1215981.

[6] R. Dorne and Jin-Kao Hao, 'An evolutionary approach for frequency assignment in cellular radio networks', in *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, vol. 2, Nov. 1995, 539–544 vol.2. DOI: 10.1109/ICEC.1995.487441.

[7] A. Raschellà, M. Mackay, F. Bouhafs and B. I. Teigen, 'Evaluation of Channel Assignment Algorithms in a Dense Real World WLAN', in *2019 4th International Conference on Computing, Communications and Security (ICCCS)*, Oct. 2019, pp. 1–5. DOI: 10.1109/CCCS.2019.8888082.

[8] S. Zehl, A. Zubow and A. Wolisz, 'Practical distributed channel assignment in home Wi-Fi networks', in *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, IEEE, 2017, pp. 1–4.

[9] S. Zehl, A. Zubow, M. Döring and A. Wolisz, 'ResFi: A secure framework for self organized Radio Resource Management in residential WiFi networks', in *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, IEEE, Jun. 2016, pp. 1–11. DOI: 10.1109/WoWMoM.2016.7523511.

[10] M. Rønning, 'TBA: Distributed and Self-Reliant Channel Convergence', PhD diss. University of Oslo, Institutt for teknologisystemer, 2020.

[11] V. Rakovic, D. Denkovski, V. Atanasovski and L. Gavrilovska, 'Radio resource management based on radio environmental maps: Case of Smart-WiFi', in *2016 23rd International Conference on Telecommunications (ICT)*, May 2016, pp. 1–5. DOI: 10.1109/ICT.2016.7500414.

[12] T. Vanhatupa, M. Hannikainen and T. D. Hamalainen, 'Frequency management tool for multi-cell WLAN performance optimization', in *2005 14th IEEE Workshop on Local Metropolitan Area Networks*, Sep. 2005, 6 pp.–6. DOI: 10.1109/LANMAN.2005.1541512.

[13] S. Bi, J. Lyu, Z. Ding and R. Zhang, 'Engineering Radio Maps for Wireless Resource Management', *IEEE Wireless Communications*, vol. 26, no. 2, pp. 133–141, Apr. 2019, ISSN: 1536-1284. DOI: 10.1109/MWC.2019.1800146.

[14] S. Haykin, 'Cognitive radio: Brain-empowered wireless communications', *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 201–220, Feb. 2005, ISSN: 0733-8716. DOI: 10.1109/JSAC.2004.839380.

[15] A. Mishra, S. Banerjee and W. Arbaugh, 'Weighted coloring based channel assignment for WLANs', *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 3, pp. 19–31, 1st Jul. 2005, ISSN: 1559-1662. DOI: 10.1145/1094549.1094554.

[16] A. Baid and D. Raychaudhuri, 'Understanding channel selection dynamics in dense Wi-Fi networks', *IEEE Communications Magazine*, vol. 53, no. 1, pp. 110–117, Jan. 2015, ISSN: 1558-1896. DOI: 10.1109/MCOM.2015.7010523.

[17] Y. Matsunaga and R. Katz, 'Inter-domain radio resource management for wireless LANs', in *2004 IEEE Wireless Communications and Networking Conference (IEEE Cat. No.04TH8733)*, vol. 4, Mar. 2004, 2183–2188 Vol.4. DOI: 10.1109/WCNC.2004.1311426.

[18] Z. NUSS, A. B. Ami and M. Grayson, 'Self optimizing residential and community WiFi networks', U.S. Patent 10231140B2, 12th Mar. 2019.

[19] H.-J. Chen, C.-P. Chuang, Y.-S. Wang, S.-W. Ting, H.-Y. Tu and C.-C. Teng, 'Design and implementation of a Cluster-based Channel Assignment in high density 802.11 WLANs', in *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Oct. 2016, pp. 1–5. DOI: 10.1109/APNOMS.2016.7737200.

[20] H. Balbi, N. Fernandes, F. Souza, R. Carrano, C. Albuquerque, D. Muchaluat-Saade and L. Magalhaes, 'Centralized channel allocation algorithm for IEEE 802.11 networks', in *2012 Global Information Infrastructure and Networking Symposium (GIIS)*, Dec. 2012, pp. 1–7. DOI: 10.1109/GIIS.2012.6466657.

[21] S. Chieochan, E. Hossain and J. Diamond, 'Channel assignment schemes for infrastructure-based 802.11 WLANs: A survey', *IEEE Communications Surveys Tutorials*, vol. 12, no. 1, pp. 124–136, 2010, ISSN: 1553-877X. DOI: 10.1109/SURV.2010.020110.00047.

[22] J. K. Chen, G. de Veciana and T. S. Rappaport, 'Improved Measurement-Based Frequency Allocation Algorithms for Wireless Networks', in *IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference*, Nov. 2007, pp. 4790–4795. DOI: 10.1109/GLOCOM.2007.909.

[23] B. A. H. S. Abeysekera, K. Ishihara, Y. Inoue and M. Mizoguchi, 'Network-Controlled Channel Allocation Scheme for IEEE 802.11 Wireless LANs: Experimental and Simulation Study', in *2014 IEEE 79th Vehicular Technology Conference (VTC Spring)*, May 2014, pp. 1–5. DOI: 10.1109/VTCSpring.2014.7023003.

[24] T. H. Lim, W. S. Jeon and D. G. Jeong, 'Centralized channel allocation scheme in densely deployed 802.11 wireless LANs', in *2016 18th International Conference on Advanced Communication Technology (ICACT)*, Jan. 2016, pp. 249–253. DOI: 10.1109/ICACT.2016.7423348.

[25] Liverpool JohN moores University. (4th Sep. 2017). 'What to do With the Wi-Fi Wild West | Wi-5 Project | H2020 | CORDIS | European Commission', [Online]. Available: https://cordis.europa.eu/project/id/644262 (visited on 22/05/2020).

[26] L. J. M. University. (9th Mar. 2020). 'SMART Wi-Fi : Management of the Wi-Fi Spectrum and Performance | Smart-WiFi Project | H2020 | CORDIS | European Commission', [Online]. Available: https://cordis.europa.eu/project/id/947559 (visited on 22/05/2020).

[27] Y. Liang, Y. Zeng, E. Peh and A. T. Hoang, 'Sensing-Throughput Tradeoff for Cognitive Radio Networks', in *2007 IEEE International Conference on Communications*, Jun. 2007, pp. 5330–5335. DOI: 10.1109/ICC.2007.882.

[28] T. Yucek and H. Arslan, 'A survey of spectrum sensing algorithms for cognitive radio applications', *IEEE Communications Surveys & Tutorials*, vol. 11, no. 1, pp. 116–130, 2009, ISSN: 1553-877X. DOI: 10.1109/SURV.2009.090109.

[29] J. Mitola and G. Maguire, 'Cognitive radio: Making software radios more personal', *IEEE Personal Communications*, vol. 6, no. 4, pp. 13–18, Aug. 1999, ISSN: 10709916. DOI: 10.1109/98.788210.

[30] Y. Ma, G. Zhou and S. Wang, 'WiFi Sensing with Channel State Information: A Survey', *ACM Comput. Surv.*, vol. 1, no. 1, p. 35, Jan. 2019.

[31] S. Miyamoto and S. Sampei, 'Group-based centralized radio resource management strategies in wireless local area networks', in *2014 XXXIth URSI General Assembly and Scientific Symposium (URSI GASS)*, Aug. 2014, pp. 1–4. DOI: 10.1109/URSIGASS.2014.6929305.

[32] 'IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications', *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, Dec. 2016. DOI: 10.1109/IEEESTD.2016.7786995.

[33] S. Acharya, M. Franklin and S. Zdonik, 'Balancing push and pull for data broadcast', in *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '97, Tucson, Arizona, USA: Association for Computing Machinery, 1st Jun. 1997, pp. 183–194, ISBN: 978-0-89791-911-1. DOI: 10.1145/253260.253293.

[34] J. Leitão, J. Pereira and L. Rodrigues, 'Gossip-Based Broadcast', in *Handbook of Peer-to-Peer Networking*, X. Shen, H. Yu, J. Buford and M. Akon, Eds., Boston, MA: Springer US, 2010, pp. 831–860, ISBN: 978-0-387-09751-0. DOI: 10.1007/978-0-387-09751-0_29.

[35] M. Skjegstad, F. T. Johnsen, T. H. Bloebaum and T. Maseng, 'Mist: A Reliable and Delay-Tolerant Publish/Subscribe Solution for Dynamic Networks', in *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*, May 2012, pp. 1–8. DOI: 10.1109/NTMS.2012.6208757.

[36] J. Lipman, P. Boustead and J. Chicharo, 'Localised Minimum Spanning Tree Flooding in Ad-Hoc Networks', in *Advanced Wired and Wireless Networks*, ser. Multimedia Systems and Applications Series, T. A. Wysocki, A. Dadej and B. J. Wysocki, Eds., Boston, MA: Springer US, 2005, pp. 19–37, ISBN: 978-0-387-22792-4. DOI: 10.1007/0-387-22792-X_2.

[37] A. Saidi and M. Mohtashemi, 'Minimum-cost First-Push-Then-Pull gossip algorithm', in *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2012, pp. 2554–2559. DOI: 10.1109/WCNC.2012.6214229.

[38] R. Gupta, A. C. Maali and Y. N. Singh, 'Adaptive Push-Then-Pull Gossip Algorithm for Scale-free Networks', 22nd Oct. 2013. arXiv: 1310.5985 [cs].

[39] A. L. Liestman and D. Richards, 'An introduction to perpetual gossiping', in *Algorithms and Computation*, K. W. Ng, P. Raghavan, N. V. Balasubramanian and F. Y. L. Chin, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 1993, pp. 259–266, ISBN: 978-3-540-48233-8. DOI: 10.1007/3-540-57568-5_256.

[40] L. Cheng, J. Niu, C. Luo, L. Shu, L. Kong, Z. Zhao and Y. Gu, 'Towards minimum-delay and energy-efficient flooding in low-duty-cycle wireless sensor networks', *Computer Networks*, vol. 134, pp. 66–77, 7th Apr. 2018, ISSN: 1389-1286. DOI: 10.1016/j.comnet.2018.01.012.

[41] J. A. Cordero, P. Jacquet and E. Baccelli, 'Impact of jitter-based techniques on flooding over wireless ad hoc networks: Model and analysis', in *2012 Proceedings IEEE INFOCOM*, Mar. 2012, pp. 2059–2067. DOI: 10.1109/INFCOM.2012.6195587.

[42] K. Birman, 'The promise, and limitations, of gossip protocols', *ACM SIGOPS Operating Systems Review*, vol. 41, no. 5, pp. 8–13, 1st Oct. 2007, ISSN: 0163-5980. DOI: 10.1145/1317379.1317382.

[43] H. J. F. Nygårdshaug, 'Developing a Distributed Clustering Algorithm to Enable Self-managing Groups for 802.11 Access Points', Masteroppgave, University of Oslo, Oslo, 2018.

[44] M. B. Grønseth, 'Graph Coloring and Some Applications', Norwegian University of Science and Technology, Department of Mathematical Sciences, Jun. 2017, 65 pp.

[45] K. Illavalagan, 'TBA: Distributed DSATUR for CAPS', University of Oslo, 2020.

[46] P. Pawelczak, C. Guo, R. V. Prasad and R. Hekmat, 'Cluster-based spectrum sensing architecture for opportunistic spectrum access networks', *IRCTR-S-004-07 Report*, 2007.

[47] N. Ahmed, D. Hadaller and S. Keshav, 'GUESS: Gossiping updates for efficient spectrum sensing', in *Proceedings of the 1st International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking - MobiShare '06*, Los Angeles, California: ACM Press, 2006, p. 12, ISBN: 978-1-59593-558-8. DOI: 10.1145/1161252.1161256.

[48] D. Ongaro and J. Ousterhout, 'In Search of an Understandable Consensus Algorithm', presented at the USENIX Annual Technical Conference, 2014, pp. 305–319, ISBN: 978-1-931971-10-2.

[49] I. Minakov, R. Passerone, A. Rizzardi and S. Sicari, 'A Comparative Study of Recent Wireless Sensor Network Simulators', *ACM Trans. Sen. Netw.*, vol. 12, no. 3, 20:1–20:39, Jul. 2016, ISSN: 1550-4859. DOI: 10.1145/2903144.

[50] S. Surati, D. C. Jinwala and S. Garg, 'A survey of simulators for P2P overlay networks with a case study of the P2P tree overlay using an event-driven simulator', *Engineering Science and Technology, an International Journal*, vol. 20, no. 2, pp. 705–720, 1st Apr. 2017, ISSN: 2215-0986. DOI: 10.1016/j.jestch.2016.12.010.

[51] R. Ruslan, A. Shaqirra Mohd Zailani, N. Hidayah Mohd Zukri, N. Khairani Kamarudin, S. J. Elias and R. B. Ahmad, 'Routing performance of structured overlay in Distributed Hash Tables (DHT) for P2P', *Bulletin of Electrical Engineering and Informatics*, vol. 8, no. 2, pp. 389–395, 1st Mar. 2019, ISSN: 2302-9285, 2089-3191. DOI: 10.11591/eei.v8i2.1449.

[52] K. A. Ngo, T. T. Huynh and D. T. Huynh, 'Simulation Wireless Sensor Networks in Castalia', in *Proceedings of the 2018 International Conference on Intelligent Information Technology*, (Ha Noi, Viet Nam), ser. ICIIT 2018, New York, NY, USA: ACM, 2018, pp. 39–44, ISBN: 978-1-4503-6378-5. DOI: 10.1145/3193063.3193066.

[53] F. Paganelli and D. Parlanti. (2012). 'A DHT-Based Discovery Service for the Internet of Things', [Online]. Available: https://www.hindawi.com/journals/jcnc/2012/107041/ (visited on 22/06/2019).

[54] H. Zhang, Y. Wen, H. Xie and N. Yu, *Distributed Hash Table: Theory, Platforms and Applications*, ser. SpringerBriefs in Computer Science. New York: Springer-Verlag, 2013, ISBN: 978-1-4614-9007-4.

[55] L. Monnerat and C. L. Amorim, 'An effective single-hop distributed hash table with high lookup performance and low traffic overhead', *Concurrency and Computation: Practice and Experience*, vol. 27, no. 7, pp. 1767–1788, 2015, ISSN: 1532-0634. DOI: 10.1002/cpe.3342.

[56] M. F. Kaashoek and D. R. Karger, 'Koorde: A Simple Degree-Optimal Distributed Hash Table', in *Peer-to-Peer Systems II*, M. F. Kaashoek and I. Stoica, Eds., red. by G. Goos, J. Hartmanis and J. van Leeuwen, vol. 2735, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 98–107, ISBN: 978-3-540-45172-3. DOI: 10.1007/978-3-540-45172-3_9.

[57] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek and H. Balakrishnan, 'Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications', *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, Feb. 2003, ISSN: 1063-6692. DOI: 10.1109/TNET.2002.808407.

[58] Storj Labs, Inc., *Storj: A Decentralized Cloud Storage Framework*, 31st Oct. 2018.

[59] P. Maymounkov and D. Mazières, 'Kademlia: A Peer-to-Peer Information System Based on the XOR Metric', in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek and A. Rowstron, Eds., red. by G. Goos, J. Hartmanis and J. van Leeuwen, vol. 2429, Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 53–65, ISBN: 978-3-540-45748-0. DOI: 10.1007/3-540-45748-8_5.

[60] T. Li, X. Zhou, K. Brandstatter, D. Zhao, K. Wang, A. Rajendran, Z. Zhang and I. Raicu, 'ZHT: A Light-Weight Reliable Persistent Dynamic Scalable Zero-Hop Distributed Hash Table', in *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, Cambridge, MA, USA: IEEE, May 2013, pp. 775–787, ISBN: 978-1-4673-6066-1. DOI: 10.1109/IPDPS.2013.110.

[61] J. Gozdecki, A. Jajszczyk and R. Stankiewicz, 'Quality of service terminology in IP networks', *IEEE Communications Magazine*, vol. 41, no. 3, pp. 153–159, Mar. 2003, ISSN: 0163-6804. DOI: 10.1109/MCOM.2003.1186560.

[62] P. Richter, F. Wohlfart, N. Vallina-Rodriguez, M. Allman, R. Bush, A. Feldmann, C. Kreibich, N. Weaver and V. Paxson, 'A Multi-perspective Analysis of Carrier-Grade NAT Deployment', *Proceedings of the 2016 ACM on Internet Measurement Conference - IMC '16*, pp. 215–229, 2016. DOI: 10.1145/2987443.2987474. arXiv: 1605.05606.

[63] S. Kutten, G. Pandurangan, D. Peleg, P. Robinson and A. Trehan, 'On the Complexity of Universal Leader Election', *Journal of the ACM*, vol. 62, no. 1, 7:1–7:27, 2nd Mar. 2015, ISSN: 0004-5411. DOI: 10.1145/2699440.

[64] M. Brooker, *Leader election in distributed systems*, 2019.

[65] P. Chevalier, B. Kaminski, F. Hutchison, Q. Ma, S. Sharma, A. Fackler and W. J. Buchanan, 'Protocol for Asynchronous, Reliable, Secure and Efficient Consensus (PARSEC) Version 2.0', 26th Jul. 2019. arXiv: 1907.11445 [cs].

[66] S. Zehl, *ResFi Python Implementation*, in collab. with A. Zubow, M. Döring and A. Wolisz, version dce45e89, Nov. 2016.

[67] H. Lim and C. Kim, 'Flooding in wireless ad hoc networks', *Computer Communications*, vol. 24, no. 3, pp. 353–363, 15th Feb. 2001, ISSN: 0140-3664. DOI: 10.1016/S0140-3664(00)00233-4.

[68] H. Lim and C. Kim, 'Multicast tree construction and flooding in wireless ad hoc networks', in *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWIM '00, Boston, Massachusetts, USA: Association for Computing Machinery, 11th Aug. 2000, pp. 61–68, ISBN: 978-1-58113-304-2. DOI: 10.1145/346855.346865.

[69] R. Rab, S. A. D. Sagar, N. Sakib, A. Haque, M. Islam and A. Rahman, 'Improved Self-Pruning for Broadcasting in Ad Hoc Wireless Networks', *Wireless Sensor Network*, vol. 9, no. 2, pp. 73–86, 2 21st Feb. 2017. DOI: 10.4236/wsn.2017.92004.

[70] OpenSim Ltd. (Jan. 2020). 'OMNeT++ Simulation Manual', [Online]. Available: https://doc.omnetpp.org/omnetpp/manual/ (visited on 05/06/2020).

[71] O. Sallent, J. Perez-Romero, R. Ferrus and R. Agusti, 'On Radio Access Network Slicing from a Radio Resource Management Perspective', *IEEE Wireless Communications*, vol. 24, no. 5, pp. 166–174, Oct. 2017, ISSN: 1536-1284. DOI: 10.1109/MWC.2017.1600220WC.

[72] Y. Gu, W. Saad, M. Bennis, M. Debbah and Z. Han, 'Matching theory for future wireless networks: Fundamentals and applications', *IEEE Communications Magazine*, vol. 53, no. 5, pp. 52–59, May 2015, ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7105641.

[73] N. H. Eide, 'TBA: Radio Sensing for Autonomous Dynamic Distributed Radio Resource Management', University of Oslo, 2020.