

# NETWORK-LAYER GEOCAST

GEOGRAPHIC ADDRESSING AND  
ROUTING FOR FIXED NETWORKS

Bernd Meijerink





# **Network-Layer Geocast**

Geographic Addressing and Routing for Fixed  
Networks

Berend Jan Meijerink

## Graduation Committee

**Chairman:** Prof. dr. J.N. Kok

**Promoter:** Prof. dr. ir. G.J. Heijenk

## Committee Members:

Prof. dr.	J.L. van den Berg	University of Twente, The Netherlands
Dr. ir.	M.J. van Sinderen	University of Twente, The Netherlands
Prof. dr. ir.	B.R.H.M. Haverkort	Tilburg University, The Netherlands
Prof. dr.-ing.	L. Wolf	Technische Universität Braunschweig, Germany
Prof. dr.	E. Monteiro	University of Coimbra, Portugal

## Funding sources

EU FP7 SALUS – #313296

**UNIVERSITY  
OF TWENTE. | DIGITAL SOCIETY  
INSTITUTE**

DSI Ph.D. Thesis Series No. 19-019  
Digital Society Institute  
P.O. Box 217, 7500 AE  
Enschede, The Netherlands

ISBN: 978-90-365-4891-5  
ISSN: 2589-7721 (DSI Ph.D. Thesis Series No. 19-019)  
DOI: 10.3990/1.9789036548915  
<https://doi.org/10.3990/1.9789036548915>

Typeset with L<sup>A</sup>T<sub>E</sub>X. Printed by Ipskamp Printing.



This thesis is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.  
<http://creativecommons.org/licenses/by-nc-sa/3.0/>

# NETWORK-LAYER GEOCAST

Geographic Addressing and Routing for Fixed Networks

DISSERTATION

to obtain  
the degree of doctor at the University of Twente,  
on the authority of the rector magnificus,  
Prof.dr. T.T.M. Palstra,  
on account of the decision of the graduation committee,  
to be publicly defended  
on Wednesday the 11<sup>th</sup> of December 2019 at 14:45

by

Berend Jan Meijerink

born on December 20, 1988  
in Hardenberg, the Netherlands.

This dissertation has been approved by:

supervisor

Prof. dr. ir. G.J. Heijenk

---

## Acknowledgements

Writing a thesis, or more specifically the work needed to gather the material to write about, is a large undertaking. I have spent over 4 years (almost 5) of my life on the research that led to this book. I would not have been able to complete this thesis, or even start working on it, without the support of a large number of people.

I would like to thank my supervisor Geert, who offered me a PhD position while I was working on my master's thesis under his supervision. Geert is always available to answers questions and discuss research and non-research related topics. Next to his knowledge on the subject of vehicular communication and networking, he is also a great travelling companion. That last part might even be the most important when visiting conferences and project meetings in far away places. Without his guidance it would not have been possible to complete the research that led to this thesis.

I would also like to thank all my current and old colleagues of the DACS group. A very welcoming and fun group of people which I first got to know during the work on my master's thesis. In fact I had so much fun that when a PhD position was offered to me the group was certainly one of the main factors in me accepting that position. I especially want to thank everyone that helps make DACS more than just a work environment by organizing things like the somewhat regular movie night and having interesting (mostly completely ridiculous) conversations during the morning coffee break.

A special thank you to my paranymphs Aashik and Justyna, a current and a previous member of the DACS group, who are supporting me during the defence of this thesis. You are both great people to be around during and outside of working hours and I am happy to have you supporting me during my defence.

Also a huge thank you to my friends and family, I would not have been able to complete this thesis without your support. Many of you where always asking me what I was working on during the years. I have tried explaining my work to most of you, but I doubt I have managed to communicate what I was working on exactly. Despite this, all of you somehow managed to remain interested all this time.

Last, but certainly not least, I would like to thank my parents: Egbert and Eefke, and my sisters: Nanda and Gerinda. You have always encouraged and

supported me in whatever I wanted to do. Without your support I doubt I would have managed to start university at all, let alone start and finish all the work required for this thesis.



---

## Abstract

There is an increasing number of always connected devices in the world, of which many are mobile. A special case of these devices are connected vehicles, which can use their data connectivity to distribute data amongst themselves or even towards the wider Internet. Many of these devices can benefit from location-dependent data. In the case of connected vehicles examples could be local traffic information, accident notifications and weather information. In general, we can think of specific emergency information for all devices in a certain region. This information could be meant for a single intersection, a entire road or even a city. Some of the sources of this type of information could be local, but many would be located further in the network, or in a different network altogether.

In today's Internet, location-dependent data is either sent to a host by having the host actively poll for it, or by some sort of central authority (for a specific application) keeping track of the host's location and sending it the relevant data. All of this is achieved by using unicast communication. The result is that hosts close to each other will likely receive identical data, which is transmitted multiple times over the network. If data could be sent to a specific area instead of only to host-specific network addresses, we could reduce the load on the network for this type of communication. It also has the benefit that applications no longer need to keep track of device locations. Sending data to a location is generally referred to as geocast, or geographically scoped broadcast.

In this thesis we research, design and evaluate a system which could enable geocast both within and between networks. We do this in 4 steps: i) We design a addressing system that can address areas anywhere on the planet; ii) we evaluate different forwarding trees for their link usage and fairness when applied to geographically scoped destinations; iii) we design, implement a prototype and evaluate both a path-based and distance-vector-based geographic forwarding algorithm; iv) we design and evaluate a system that can help vehicles forward messages between themselves by using geocast-enabled infrastructure when available.

Our addressing system is based on addressing the world as four rectangles, which in turn each contain four rectangles. This pattern repeats recursively until the desired accuracy is reached. Due to the way we number these rectangles,

we can easily aggregate neighbouring areas into a single address. We design this system in such a way that we can determine overlap between addresses based on prefix matching, similar to the way routers in the Internet lookup addresses in their packet forwarding operation. We evaluate our proposal on how accurate it can represent arbitrary areas anywhere on the planet.

To find the most applicable forwarding tree for geocast, we evaluate a shortest path tree, minimum spanning tree and Steiner tree. We will test this by evaluating all three trees on a large selection of real-world network topologies. This test consists of selecting a source and a geographically scoped destination area and calculating all possible trees of the three types between them. The shortest path tree uses only slightly more links than the Steiner tree, but has the benefit of not requiring a complex computation in each router.

We design a distance-vector and a path-based forwarding algorithm that establishes shortest path trees specifically for the geographic routing use-case. The algorithms allow nodes to establish shortest path forwarding trees for any source destination combination without having full network knowledge. We show that the number of links used by the path-based algorithm is identical to the predicted link usage of the topology-based evaluation. The distance-vector-based algorithm performs slightly worse, but has the benefit of requiring less network knowledge and being less complex.

We also make prototype implementations of both algorithms, including a protocol for exchanging coverage and path information. We evaluate these implementations in a network emulator and show that they perform as expected in terms of links used to forward packets. We also evaluate the convergence properties of both implementations and show that they converge relatively quickly.

Finally, we propose a method for vehicular networks to use infrastructure with geocast capabilities to help forward messages with a geographically scoped destination. This proposal allows the infrastructure to ‘cancel’ ad-hoc forwarding of messages between vehicles, and instead relay them through the geocast enabled infrastructure. We evaluate this proposal in a simulator and show that it can significantly reduce wireless traffic, increase delivery rates and lower the delivery delay.

All of these proposals combined allow a packet to be routed from a source to all devices in a geographic area. This source can also be located in a different network, as our path-based routing proposal can potentially enable inter-network routing. The message could even traverse multiple networks before the networks that actually cover the destination are reached. We hope that all these proposals combined provide the building blocks that will eventually lead to Internet-wide geocast support.

---

## Samenvatting

Er zijn steeds meer apparaten in de wereld, waarvan een groot deel mobiel, die met het internet verbonden zijn. Een deel van deze apparaten bestaat uit voertuigen die hun dataverbinding gebruiken om gegevens met elkaar en het internet uit te wisselen. Veel van deze apparaten gebruiken locatieafhankelijke data, zoals weer- en verkeersinformatie. Een algemeen voorbeeld is het sturen van een noodbericht aan alle apparaten in een bepaald gebied. Deze informatie zou gericht kunnen zijn aan een enkel kruispunt, een straat of een complete stad. De bron van deze informatie zou lokaal kunnen zijn, maar zou ook verder in het netwerk kunnen zitten of op een ander netwerk.

Locatieafhankelijke data is al veel in gebruik in het huidige internet. Dit is meestal geïmplementeerd door applicaties specifiek data op te laten vragen bij de server, of de server de locatie van de gebruiker bij te laten houden en periodiek data door te laten sturen. Al deze communicatie verloopt via unicast. Apparaten die dicht bij elkaar zijn zullen in deze situaties waarschijnlijk allemaal individueel dezelfde data toegestuurd krijgen. Als data naar een gebied gestuurd zou kunnen worden in plaats van naar individuele apparaten dan zouden we het netwerk veel efficiënter kunnen gebruiken. Het zou ook niet langer nodig zijn om de locatie van alle apparaten bij te houden. Het sturen van data naar een gebied noemen we *geocast* of *geographically scoped broadcast*.

In deze thesis onderzoeken we *geocast* binnen en tussen netwerken. We ontwerpen en evalueren systemen die dit doel kunnen bereiken. We doen dit in 4 stappen: i) We ontwerpen en evalueren een adresseringssysteem voor gebieden. ii) We evalueren verschillende typen bomen op de hoeveelheden paden die ze gebruiken bij geografisch geclusterde locaties. iii) Wij ontwerpen en implementeren een prototype van een op paden gebaseerd en een op *distance vector* gebaseerd routeringsalgoritme voor *geocast*. iv) Wij ontwerpen en evalueren een systeem dat voertuigen kan helpen met het doorsturen van *geocast* berichten via vaste infrastructuur.

Ons geografisch adresseringssysteem werkt door de wereld in vier gelijke rechthoeken te verdelen. Elk van deze rechthoeken is zelf ook weer in vieren gedeeld. Dit patroon herhalen we tot de vereiste nauwkeurigheid behaald is. We nummeren deze rechthoeken op een manier waarmee het mogelijk is deze gemakkelijk te combineren. Met deze methode kan een enkel adres meerdere rechthoeken aanduiden. Ons systeem maakt het ook mogelijk om door middel

van *prefix matching* overlap tussen adressen te vinden, op een manier die vergelijkbaar is met de methode die IP routers gebruiken. We evalueren onze aanpak op de nauwkeurigheid van het adresseren van willekeurige gebieden.

We evalueren een *shortest path tree*, *minimum spanning tree* en *Steiner tree* op hoe effectief ze zijn in *geocast* situaties. We doen dit door ze te testen op een grote set van netwerktopologieën uit de echte wereld. Deze tests bestaan uit het evalueren van elk mogelijk (bron, bestemming) paar dat mogelijk is in een netwerk voor alle drie de bomen. We laten zien dat de *shortest path tree* maar een klein beetje slechter presteert dan een *Steiner tree* terwijl de *shortest path tree* een stuk minder complex is.

Wij ontwerpen twee *geocast* routeringsalgoritmen op basis van distance vector en pad informatie. Deze algoritmen bouwen *shortest path trees* voor een (bron, bestemming) paar zonder dat ze kennis van het gehele netwerk nodig hebben. We laten zien dat het aantal paden dat gebruikt wordt door ons op paden gebaseerde algoritme gelijk is aan het aantal paden uit de theoretische evaluatie. Het distance vector algoritme presteert iets minder maar heeft ook minder kennis van het netwerk nodig en is ook minder complex.

Beide algoritmen worden ook geïmplementeerd, inclusief een protocol om netwerk- en dekkingsinformatie uit te wisselen. We evalueren onze implementaties in een netwerk emulatie omgeving waar we gebruik maken van dezelfde netwerktopologieën die we eerder gebruikt hebben. We laten zien dat het aantal paden dat onze implementaties gebruiken overeenkomt met de theoretische evaluaties. We laten ook zien dat onze implementaties snel convergeren na veranderingen in het netwerk.

Als laatste onderdeel van deze thesis beschrijven wij een systeem dat voertuigen kan helpen met het doorgeven van *geocast* berichten. Met een Europese standaard voor voertuig communicatie kunnen voertuigen *geocast* berichten voor elkaar doorgeven om zo locaties ver van de originele zender te bereiken. Deze methode van berichten doorsturen werkt niet goed op tijden dat het minder druk op de weg is. Wij stellen voor om naast de weg aanwezige infrastructuur zo aan te passen dat het kan helpen met het doorgeven van deze berichten. Wij laten zien dat we met relatief kleine aanpassingen aan de infrastructuur, en geen aanpassingen aan de voertuigen, berichten betrouwbaarder en sneller af kunnen leveren.

De combinatie van de voorstellen die we in deze thesis presenteren maken het mogelijk om data te sturen naar een geografisch gebied. Dit systeem kan werken binnen een enkel netwerk maar kan ook geografische routing tussen netwerken mogelijk maken. Het doel van onze voorstellen is om het uiteindelijk mogelijk te maken om geografische routing in het gehele internet te ondersteunen.

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Geocast . . . . .	2
1.2	The Need for Geographic Routing . . . . .	3
1.3	Research Questions and Approach . . . . .	6
1.4	Scope of this Thesis . . . . .	8
1.5	Requirements for Network-Layer Geocast . . . . .	8
1.6	Contributions . . . . .	11
1.7	Thesis Structure . . . . .	12
<b>2</b>	<b>Background &amp; Related Work</b>	<b>15</b>
2.1	Introduction . . . . .	16
2.2	Geocast . . . . .	16
2.3	Routing . . . . .	22
2.4	Vehicular Networking . . . . .	25
2.5	Evaluation Tools . . . . .	27
<b>3</b>	<b>Geographic Addressing</b>	<b>31</b>
3.1	Introduction . . . . .	32
3.2	Related Work . . . . .	32
3.3	Addressing Requirements . . . . .	34
3.4	A Geographic Addressing System . . . . .	38
3.5	Accuracy . . . . .	46
3.6	Application to Network-Layer Geocast . . . . .	50
3.7	Summary & Conclusion . . . . .	51
<b>4</b>	<b>Geographic Forwarding Trees</b>	<b>53</b>
4.1	Introduction . . . . .	54
4.2	Related Work . . . . .	56
4.3	Evaluation Approach . . . . .	57
4.4	Evaluation Results . . . . .	65
4.5	Summary & Conclusion . . . . .	74

<b>5</b>	<b>Geographic Routing Algorithm Design</b>	<b>77</b>
5.1	Introduction . . . . .	78
5.2	Related Work . . . . .	81
5.3	Algorithm Design . . . . .	82
5.4	Evaluation . . . . .	99
5.5	Summary and Conclusions . . . . .	106
<b>6</b>	<b>Geographic Routing Implementation and Evaluation</b>	<b>109</b>
6.1	Introduction . . . . .	110
6.2	Implementation . . . . .	111
6.3	Evaluation . . . . .	114
6.4	Evaluation Results . . . . .	117
6.5	Open Issues . . . . .	127
6.6	Summary & Conclusion . . . . .	129
<b>7</b>	<b>Infrastructure Assisted Contention-Based Forwarding for Geo-</b>	
	<b>cast</b>	<b>131</b>
7.1	Introduction . . . . .	132
7.2	Contention Based Forwarding . . . . .	134
7.3	Multi-Hop Transmission Range . . . . .	135
7.4	Infrastructure-Assisted Geographic Broadcast . . . . .	140
7.5	Evaluation . . . . .	147
7.6	Open Issues . . . . .	156
7.7	Summary and Conclusion . . . . .	157
<b>8</b>	<b>Conclusions and Future Work</b>	<b>159</b>
8.1	Introduction . . . . .	160
8.2	Summary . . . . .	160
8.3	Conclusions . . . . .	162
8.4	Future Work . . . . .	164
	<b>Bibliography</b>	<b>167</b>
	<b>List of Acronyms</b>	<b>175</b>
	<b>Publications by the Author</b>	<b>177</b>
	<b>Open Data Management</b>	<b>179</b>

## Introduction

The world is becoming increasingly connected through the Internet. The last years have seen an increase in connected devices. Most of these devices are in the so-called Internet-of-Things (IoT) category, which encompasses devices not (traditionally) considered computers like lamps, thermostats and even cars. Many, but not all, of these devices are wireless and mobile. Multiple services used by these devices rely on the device's location. Locality can be important for many applications. Smart-phone users might be interested in having local weather information. Smart vehicles and the people inside them might be interested in traffic or emergency-services information, which are both highly location dependent [1, 2].

Traditional means of data routing require either these devices to actively request information for their location, or the current location to be known at some external entity so it can push the relevant data to the device. While to some extent current services do exactly these things to deliver location-dependent data, this method might not be scalable, or desirable due to privacy concerns, once a larger number of devices becomes connected. The largest problems will likely occur due to overhead in network communication for data requests and bookkeeping at a (de)central authority of all device locations.

The main problem with current solutions is that location-dependent data can only be sent to a device by a system that is aware of the device's location and its network address. The result is that a user either has to actively search for location-relevant data or becomes dependent on some entity that keeps track of the device. It might be beneficial if other devices could send messages towards certain locations without this need for a central authority or active data requests. For example: a police warning for a certain group of streets might be sent to only devices near those streets instead of a much larger area as is currently the case with cell-broadcasting as used in the Netherlands [3]. Another example would be a vehicle that sends a warning message to other local road users following an accident [4].

A possible solution to the location-dependent data delivery problem is geocast. Geocast is the concept of sending packets to a destination location instead of a static address [5]. Geocast would enable messages meant for a

specific area to be delivered to all devices in that location. Such a system would not only make existing applications that use location-dependent messages more efficient, but also allow for entirely new applications not dependent on a central authority.

In this thesis we will investigate network-layer geocast, including an addressing system, forwarding algorithms and application. In this introductory chapter we will introduce the concept of geocast and explain the reason network-layer geocast is needed. We will also introduce our research questions, our research approach, our requirements and our contributions.

## 1.1 Geocast

In the traditional Internet there are three main ways to forward packets: unicast, multicast and broadcast. Broadcast means that any device on the network receives the packet. In IPv4 networks broadcast is restricted to the local subnet. It is mostly used for initial configuration where the network address of other devices is not known. Broadcast was dropped in IPv6 in favour of multicast. Of the remaining two forwarding methods, unicast is by far the most used. Any interaction with what most people consider the Internet will almost certainly have unicast as the underlying delivery method. Unicast packets are routed towards a single destination, identified with a (for that network) unique address. Multicast is more often used for local applications. These applications range from service discovery to television streams. Multicast packets are routed towards a group of devices. In IP networks, a device subscribes to a multicast group, which is defined as a special IP address in a range reserved for multicast. Multicast requires each router to keep track of subscriptions to deliver these packets.

In addition to the traditional forwarding methods, unicast and multicast, there is a geographically-scoped delivery concept, called Geocast. Like multicast, geocast packets are forwarded to multiple destinations. Unlike multicast the geocast packets are sent to a certain geographic destination, instead of a pre-defined or changing set of hosts. This property allows geocast to function without a per-destination subscription mechanism, which is required for multicast.

In the literature geocast is often divided into geobroadcast, geounicast, and geoanycast [6]:

- *Geobroadcast*: A forwarding mechanism that sends data from a single point of origin to all nodes in geographical area. Sometimes also referred to as *Geographically-scoped broadcast* or simply *Geographical broadcast* in the literature. It could be used for data that the large majority of



nodes in an area is interested in. An example would be sending weather information to all nodes in a region.

- *Geounicast*: Forwards data from an origin to one specific node using geographic addressing. This type of geocast is sometimes also referred to as *Geographical unicast*. The type of geocast is comparable to ‘normal’ IP unicast but with geographical based addresses as opposed to IP addresses.
- *Geoanycast*: The forwarding of data to any single node in a specific area. Sometimes called *Geographically-scoped anycast* in the literature. It is comparable to the geocast mechanism only the data will not be forwarded further when it arrives at the first node in the destination area.

For this thesis we will exclusively focus on a *Geobroadcast* system as this is the most applicable for our use-cases. Geounicast makes little sense in most networking contexts, if we know the position of a single device we can just address it using a ‘normal’ unicast address. Geoanycast is a subset of geobroadcast where forwarding stops once a host inside the destination has received it. For this thesis we will mostly refer to *Geobroadcast* as simply geocast, unless comparison with one of the other geocast types is needed.

## 1.2 The Need for Geographic Routing

Geocast has important use-cases in the area of mobile systems and for vehicular networking in particular. Vehicular networks refer to any type of network in which vehicles, such as cars, participate. An important characteristic of these networks is that most of the nodes are mobile. A special type of vehicular network is the Vehicular Ad-hoc Network (VANET), in which vehicles can communicate with each other without using fixed infrastructure. In some systems vehicles will forward messages for each other to reach destinations outside their own transmission range.

There are several situations for these mobile systems, such as the transmission of traffic info, where it is unimportant to address specific nodes, but important that all nodes in a certain area receive a message. For example, cars on a specific road might need to know an ambulance is coming so they can make room, while a car on a parallel road does not need to receive this information. As such, geocast is a highly relevant forwarding mechanism for these networks [7].

A system based on IP multicast could provide similar functionality to a geocast system by, for example, specifying a multicast group for a specific region such as suggested in [8]. There are several disadvantages to such an approach:

- This multicast-based approach would certainly work for a small number of regions, but has a scalability problem once the number of regions starts to grow.
- Furthermore, this approach prevents the addressing of an arbitrary region by the system as all the groups have to be predefined, or a signalling system needs to be in place to make groups for a region on the fly.
- Multicast requires per destination subscriptions, which would not be needed for geocast as geographic areas are static.

The main problem of using an IP multicast-based approach is that any sending node would somehow need to know the multicast address of its destination region. In smaller static networks this could be done by pre-distributed lists with the regions and their corresponding multicast groups, but in larger or more dynamic networks some kind of lookup service (such as eDNS [9]) would be needed.

Many previous proposals for geocast such as GeoTora [10] and the Grid Location Service [11], which we will further describe in the next chapter, are focused either on ad-hoc wireless scenarios or overlay systems. Ad-hoc protocols mostly have a strong reliance on the relation between the geographic distance to a node and the distance to that node in the network topology. As a result, a common approach is to forward packets to neighbouring nodes which are located towards the general direction of the destination of the packet. This property allows such systems to route packets to their destination area without information on the path towards that area, as long as there are sufficient nodes on this path. There have been some proposals that can also handle ‘holes’ in the network and route around them [12], but they still rely on the basic principle of forwarding packets in the direction of the destination. While using this correlation makes sense in the case of an ad-hoc network with moving nodes, where it is very costly to keep track of forwarder locations, it does not in a wired-network scenario. Most wired networks are connected in patterns that are based on historical and financial reasons. There is likely some geographic component, but especially when routing between networks packets most often do not travel in a geographically logical fashion. Packets sent from one device to another nearby device using a different Internet provider might first have to pass through an Internet-exchange, possibly travelling hundreds of kilometres to arrive several meters away.

Another option for adding geographic routing to the Internet are overlay networks, like the original geocast proposal by Navas and Imielinski [5] and the overlay network for symbolically addressed geocast messages [13]. Overlay networks have the benefit that they can be deployed without changing the underlying infrastructure of the network. While this increases the chance of

the system being adopted it also comes with some drawbacks. The overlay network will have larger overhead compared to a network layer solution, it will also require extra devices in the network to make geographic routing possible.

It is also possible to provide geocast like capabilities on the application layer using unicast. This requires some sort of lookup service that can provide the sender with the unicast addresses of all devices in a certain area. One of these proposals is eDNS [14], extending DNS with the capability to lookup polygons and return the address of all devices inside that area. This approach has some problems associated with it. There would need to be a central or decentralized system that keeps track of all devices and their locations. Both central and de-central approaches have their up- and downsides, but they both increase overhead and are difficult to scale. Therefore, it is beneficial to not track vehicle locations, but to simply forward packets to their destination and deliver them to all hosts in that location.

### 1.2.1 Vehicular Networking Applications

One of the most interesting use cases for geocast lies in the area of vehicular networks. These networks are comprised of mobile devices such as cars and possibly fixed devices (Road Side Units (RSUs)) which might allow connections to other networks [6, 15]. VANETs are a type of vehicular network in which vehicles can communicate directly with each other, and depending on the standard also forward messages for each other without infrastructure. Many geocast proposals focus on this specific area. For wireless VANETs there is a strong correlation between vehicle location and the forwarding direction of a packet. An extra benefit is that vehicles are mostly bound to roads and as such their locations are predictable [2], which can greatly help with forwarding.

Vehicular networks can also receive messages from a fixed network through RSUs. These fixed networks can in turn be connected to the greater Internet. Our focus is on getting messages from these fixed networks towards the RSUs, or towards the network that contains these RSUs. Messages from external, non-vehicle sources can help with applications important for safety such as congestion avoidance, accident notifications and more [16].

Most devices in the vehicular network are likely interested in the same data, for example information on accidents and weather reports. It would be inefficient to send this type of data over unicast, as each device would receive the same data. Because devices in for example Enschede would not be interested in an accident occurring in Amsterdam the broadcasting of certain data can be limited to a geographical area. Specific data about accidents on high-ways could be broadcast country-wide as the amount of data is limited and the information is relevant to users in the entire region. Geocast is ideal for these situations, as it would significantly reduce network load compared to

unicast. Further, multicast-based solutions would have more overhead due to subscription management.

### 1.3 Research Questions and Approach

Based on the current lack of fixed network geocast we have formulated the main research question of this thesis:

*How can an efficient system for network-layer geocast be designed?*

To answer this broad question we further divide it into several sub-questions, with which we address specific problem areas. Our high-level approach to answer these questions generally consists of designing the relevant part of the geocast system and evaluating it in a manner that would match how the system would be used in the real world. We do this with a combination of graph based evaluations, evaluations of prototype implementations, and simulation environments.

Packets in a network are usually routed based on their destination address. We will need to map geographic areas to such an address. This leads us to our first sub-question:

*How can arbitrary-sized areas be efficiently addressed?*

We will answer this question by designing an addressing system for geographic areas. We will take into account that routers will need to perform fast lookups and that it should be possible to aggregate addresses for scalability reasons. We evaluate the efficiency of our addressing system by measuring how accurately it can describe an arbitrary rectangle anywhere on the planet. This evaluation is performed by generating a large set of random areas and looking up the corresponding address. We then calculate how much extra area the address contains compared to the original destination.

When sending packets from a source to a destination area they will likely have to be forwarded over multiple links to reach all devices inside the destination area. This process results in a tree with the source and all destinations as its leaves. The shape of, and the amount of links used by, such a tree greatly depends on the location of the destination devices in the network. We need to find the best forwarding tree for a group of geographically scoped destinations. Because of this, our second sub-question is:

*Which forwarding tree is the most efficient for geographically scoped destinations?*

To answer this question we evaluate three different mechanisms to generate forwarding trees. We do this by comparing them on a large set of graphs based

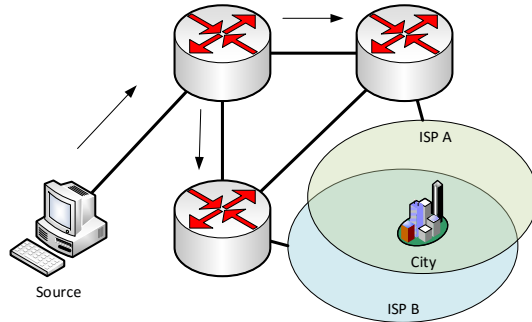


Figure 1.1: Scenario with multiple ISPs covering a city

on fixed real-world networks. This comparison is based on evaluating the total cost of the different trees in terms of link usage on all possible combinations of geographically scoped destination on the graphs.

Once we know the most appropriate way to forward packets through a network for geocast we can work on a geocast routing algorithm. Without full network knowledge a router can not know its position on the forwarding tree, making the forwarding decision non-trivial and possibly lead to redundant or even completely unneeded transmissions. Our third sub-question is:

*How can packets be efficiently routed towards a geographical area?*

The main challenge here is that geographical areas are likely covered by multiple routers or networks. One example is shown in Figure 1.1, where a city is covered by two ISPs, and a computer on a third network wants to send a geocast message to the city. To answer our question we will need to design an efficient geocast routing system for inter- and intra-network geographic routing. Our solution should efficiently solve the multiple destinations problem without resorting to multicast-like subscription mechanisms. We will evaluate our forwarding algorithms on the same set of graphs used previously for the forwarding trees. We evaluate the number of links used by the algorithm in a static situations. We will also develop prototype implementations of both algorithms and evaluate them in a emulated network environment of the same graphs.

Finally, we will need to apply our geocast system to a real problem to show the usefulness of geocast. To do this we will need to solve the last-hop problem of actually delivering a geocast packet to a host. As the vehicular network domain has many applications where network-layer geocast would be beneficial, we choose to focus on it with our fourth and final sub-question:

*How can fixed network geographic routing be applied to the vehicular networking domain?*

To answer this question we design a system that can help vehicular networks forward geocast traffic through fixed infrastructure. We will evaluate our solution in a simulator by constructing a highway environment with varying numbers of RSUs, giving us situations of full coverage to zero coverage. We will compare our proposals to the situation without infrastructure on the amount of wireless traffic and delivery rates.

## 1.4 Scope of this Thesis

A fully functioning geocast system would contain many different aspects, which we can not all explore in depth. For this thesis we choose to focus on a few specific areas that are crucial to a working geocast system. There are also a few areas on which we do specifically not focus, which we will explain in this section.

As mentioned before we will focus purely on a geobroadcast system. We do not believe geounicast will be widely used, as it requires the location of a device to be known beforehand, in which case it is likely we also already know its unicast address. The geocast scenario, where the destination is any node inside a certain area, could be achieved by using geobroadcast on the network layer and selectively delivering the message to end-hosts. We do not consider these two scenarios beyond this section.

For the majority of this thesis we will mostly consider the general requirements for geocast in a fixed network, and not focus on any specific application area. We focus on aspects related to delivering messages from their source towards a destination. As a result, most of our work is in the area of geographic addressing and geographic routing. For routing we focus on inter- and intra-network routing: the communication between routers and networks as a whole.

We do not focus on the last hop towards end-hosts in the general sense, we consider this to be mostly out of scope as it is highly dependent on the technology used. However, we will discuss an application for geocast specific to a certain vehicular networking system and show how the last hop could be implemented for such an application.

We will also not look into providing solutions to security related issues that might occur due to the use of geocast. While we will discuss possible issues in the later chapters of this thesis, solving these issues is considered out of scope.

## 1.5 Requirements for Network-Layer Geocast

To answer our research questions, and to be useful in practice, the design of our geocast system will have to fulfil certain requirements. Three things are needed

for such a system to be functional: A global addressing system for areas, a routing protocol to route the packets to the destination area and a system to deliver data to all end-hosts in an area. In this section we will describe the requirements for such a system given our focus area.

Here, we will define a set of requirements for a network-layer geocast system. We base these requirements in part on the IETF draft "Internet-wide Geo-networking problem statement" [1]. This document specifies several requirements an Internet wide geo-networking system would need to fulfil in order to provide the required functionality for vehicular networks. The document also takes geounicast and geoanycast into account. While we do not exclusively focus on vehicular networking applications and only take into account geobroadcast, this document does provide a set of requirements that are applicable to our geocast scenario. We define the following requirements which our geocast system should fulfil:

1. Minimal changes to existing routing infrastructure
2. Minimal changes to the IP layer in source and destination nodes
3. Inter-network geographic routing
4. Compatibility with IPv6

Minimal changes to the *existing routing infrastructure* allows a new geocast mechanism to function on existing networks. It also allows an eventual deployment and adoption of such a system to happen more easily, saving costs compared to a method based on a completely new routing system. Geocast packets should at least be formatted in such a way that a 'normal' IP router can parse and discard them based on their unknown address format.

Ensuring that there are no, or very few, changes in the *source and destination node IP layers* can ensure that a new geocast mechanism can be used without much effort. A small update of a host's network stack is ideally all that is needed for a geo-routing deployment. Ideally, this will also prevent nodes that have not been updated from crashing or other showing undefined behaviour when receiving a geocast packet.

Any system that is to gain wide acceptance should support *Inter-network geographic routing*, as this would allow any IP based network to fully support geocast. Geocast packets should not be limited to a single network, like multicast mostly is in practice, but should be able to reach locations served by other networks as well.

Our solution should be compatible with IPv6. Basing a possible solution on or at least supporting IPv6 will allow a new system to remain compatible with the Internet of the future.

We should note that while privacy and security are not a primary focus of this thesis, and thus not a part of our design requirements, we will try to take these aspects into account in our design. As any geocast system might potentially transmit or contain privacy or security sensitive information, such as the location of devices. We will try to design our system in such a way that keeping track of this information is not necessary.

### 1.5.1 Performance Requirements

Performance is an important aspect of any communications system. Data that takes too long to reach its destination might become as useless as data that never reaches its destination at all. For this reason, we will briefly discuss some performance specific requirements as an extension to our main requirements. While some of these requirements are implementation specific, our design should take these into account so that it is actually possible to implement our proposals efficiently.

We focus on three main performance requirements while designing our geocast system:

1. Low-latency / minimal forwarding delay.
2. Low signalling and routing overhead.
3. Scalability.

Geocast packets should be delivered with as low a latency as possible. Ignoring transmission delay, which is unrelated to the packet forwarding method, this implies that forwarding should happen with a very low delay. We should optimize our addressing system and forwarding algorithms for low lookup delays. As one of our use-cases is in the area of vehicular networks, this low-latency requirement is extra important [4].

For a geocast system to be deployed, the routing system should have a small overhead and signalling should most likely be minimized if possible. As we already noted, one of the implications is that our addressing system should allow aggregation to reduce overhead. This ensures that the system itself does not impact ‘normal’ traffic flowing through the same routers and links as the geocast traffic.

Scalability is also needed, as several use-cases for such a system require at least country-wide support for geocast. This means that inter-network routing should be supported, geocast traffic should not be limited to a single network. Another consequence is that our addressing system will likely need to be able to aggregate different areas into a single (or at least a smaller number of) area description(s).



## 1.6 Contributions

The main goal of this thesis is to present the design of and evaluate a geocast system that can deliver packets to all hosts in a given destination area from anywhere in a network. In this section we will describe the main contributions of our work.

We design and evaluate an addressing system that allows a source to address arbitrary sized rectangular areas anywhere on the planet based on nested rectangles. This system allows rectangular areas of different sizes, with a lower limit of 7.5 cm and an upper limit of the entire planet, to be addressed. Our system also allows the aggregation of the addresses of neighbouring or overlapping areas into a single address. This addressing system allows overlap to be calculated based on prefix matching, saving routers from computationally expensive geographic overlap calculations. We have performed an extensive evaluation of this addressing system, which shows that it can accurately represent areas of any size.

We perform an extensive evaluation of the applicability to geocast of three different forwarding trees: a shortest path tree, a Steiner tree, and minimal spanning tree. We evaluate how many links these different trees use and how fairly the traffic would be distributed over the network on a large set of real-world network topologies. Our evaluation shows that the shortest path tree and Steiner tree show similar link usage for geographically clustered destinations. Based on our results, we conclude that the shortest path tree is the forwarding tree best suited to geocast.

To forward geocast packets we design and evaluate two packet forwarding algorithms: a distance-vector based forwarding algorithm and a path-based one. These algorithms are designed to forward packets from a source router to all routers in a destination area over a shortest path tree. Neither algorithm requires per-packet or per-destination signalling like most multicast solutions, all data needed for routing is distributed through periodic messages exchanged between routers. We evaluate these algorithms on their link usage and show that our path-based approach indeed establishes shortest path trees from a source to a set of destinations. Our distance-vector approach uses more links, but does deliver to all destinations with a forwarding algorithm that is less complex and requires less network knowledge than the path-based approach. We also develop prototype implementations of both forwarding algorithms, including the route and coverage area distribution system. We evaluate our prototypes using a network emulator and show that they forward packets as expected. We also show that these algorithms converge relatively quickly following a link drop in the network and do so with minimal packet loss.

Finally, we have designed a system to forward geocast messages of vehicular networks through fixed infrastructure. One of the available forwarding

algorithms for vehicular networks is Contention Based Forwarding (CBF). Our system consists of a modified CBF algorithm for RSUs to help forward geocast messages from and towards vehicles. We specifically do not modify the CBF algorithm used by the vehicles. We have designed our modifications to help packet forwarding in areas where RSU coverage exists, but to have no negative impact in areas without coverage. We have evaluated our approach through a simulation study and show that it increases the probability of delivering packets while reducing the delivery delay. Our approach can also significantly decrease wireless channel usage when there is full coverage.

## 1.7 Thesis Structure

In this thesis we work towards a geographic routing system for Internet-wide geocast. The structure of the thesis is shown in a graphical format in Figure 1.2. This figure will be repeated every chapter, with the current chapter highlighted in green.

We will first present relevant background information, past geocast proposals and other related work in Chapter 2. We will also describe some of the tools used in our evaluation in this chapter. This information should give the reader enough background information to read the rest of this thesis.

Geocast requires a method to define a destination area and allow forwarding based on this definition. In Chapter 3 we will explain our addressing system and how it solves the addressing problems described in Section 1.5. We also evaluate our system on how accurately it can represent arbitrary areas of the planet.

Before we can think about routing we need to know how we can efficiently forward geocast traffic towards a destination area. We describe our search for the most appropriate type of forwarding tree for geocast routing in Chapter 4. We evaluate a shortest path tree, minimum spanning tree and Steiner tree for their applicability in geographic routing.

The design of two geocast forwarding algorithms, one path-based and another distance-vector-based, both based on the found forwarding tree type is described in Chapter 5. We describe the choices made during the design of these forwarding algorithms. We also evaluate both algorithms against each other and the results from the shortest path and Steiner tree from Chapter 4.

In Chapter 6 we describe an implementation of both our algorithm proposals. This chapter runs parallel to Chapter 5 in that we describe implementations of the algorithms described there. Our implementation includes route distribution, resulting in a complete routing protocol. We confirm the results of the evaluation done in Chapter 5 with the implementations and also evaluate both implementations on convergence time and packet loss during a link loss

scenario.

In Chapter 7 we will present a system that can take advantage of our proposed geographic routing system for vehicular networking. Our system allows geocast messages from vehicles to be forwarded through infrastructure instead of hop-by-hop.. This proposal aims to reduce wireless load, increase delivery rates and reduce delivery delay. We evaluate our proposal in a simulator to see how well it increases reliability, reduces end-to-end delay and reduces wireless load.

Finally, we summarise our work in Chapter 8. We also draw conclusions based on the results of the previous chapters. At the end of this chapter we look to the future and describe which areas will require more work to enable widespread, hopefully Internet-wide, use of geocast.

1. Introduction	
2. Background & Related Work	
3. Geographic Addressing	
4 Geographic Forwarding Trees	
5. Geographic Routing Algorithm Design	6. Geographic Routing Implementation and Evaluation
7. Infrastructure Assisted Contention-Based Forwarding for Geocast	
8. Conclusions and Future Work	

Figure 1.2: Thesis structure



---

## Background & Related Work

*In this chapter we will briefly discuss background information that is relevant for understanding geocast. We also discuss earlier work on geocast and show that these past proposals do not cover all the requirements we have for geocast. General routing concepts that the reader should be familiar with to understand this thesis are also discussed.*

1. Introduction	
2. Background & Related Work	
3. Geographic Addressing	
4 Geographic Forwarding Trees	
5. Geographic Routing Algorithm Design	6. Geographic Routing Implementation and Evaluation
7. Infrastructure Assisted Contention-Based Forwarding for Geocast	
8. Conclusions and Future Work	

## 2.1 Introduction

In this chapter we will present general background information regarding geocast that is relevant for this thesis. While we will discuss related work on the concept of geocast, specifics will be discussed in-depth in the appropriate chapters. There is also a short description on routing and routing protocols in general. We will also briefly discuss the tools we use to perform evaluation throughout this thesis. A reader familiar with these general concepts can safely skip this chapter.

This chapter is structured as follows: We describe the general concept of geocast and related work on the subject in Section 2.2. In Section 2.3 we describe the general concept of unicast and multicast routing. A short overview of some general vehicular network concepts, specifically the concepts used in Chapter 7, is given in Section 2.4. We give a short overview of the simulation tools we use throughout this thesis in Section 2.5.

## 2.2 Geocast

Geocast is the concept of sending data towards a geographical destination instead of a fixed address. Numerous papers have been published about geocast protocols in ad-hoc networks, allowing data to be sent to other location in the wireless ad-hoc network based on the location of the destination node(s). Papers describing geocast systems for large fixed IP networks are however, rare. In this chapter, we discuss several past proposals and background information of common technologies referred to throughout this thesis will be described.

In this section we will briefly describe three types of geocast, followed by a section on geographic addressing proposals. Finally we will describe past proposals for geographic routing and why they do not fulfil all the requirements we have defined in Chapter 1.

### 2.2.1 Types of Geocast

As we have noted in Chapter 1, in the literature three main types of geocast are mentioned [1, 6]:

- Geobroadcast
- Geounicast
- Geoanycast

All these types of geocast share the property that a packet is routed to a location. Geobroadcast and geoanycast share the property that a packet is

routed to an area. They differ in the way a packet is forwarded within that area. With geocast the packet is not forwarded once it has reached a node within the destination area. With Geobroadcast the goal is to reach all nodes in the destination area, and the packet is forwarded by nodes inside the area. Geounicast is the odd one out, in that only a single node is addressed.

For the reasons mentioned in Chapter 1, we will mostly refer to geobroadcast as simply geocast. Only when a comparison is needed between these three types of geocast, or some related work specifically includes them, will we specify the type.

### 2.2.2 Geographic Addressing

A geographic address is an address that somehow represents a geographic location. There are multiple approaches possible ranging from relatively straightforward to complex. In this section we will describe some relevant geographic addressing approaches and why we think they are not feasible for network-layer geocast. A more extensive description of these and more approaches can be found in Chapter 3.

One of the more straightforward approaches is to base geographic addresses on coordinates, such as done in [5] by using the WGS-84[17] coordinate system. Using this approach any shape, from a single point to complex polygons, can be represented given that there is enough ‘addressing’ space available to list multiple coordinates. The downside for routers using such approaches is that they need to perform relatively complex calculations to compute overlap between areas and the addresses of packets. The per-packet overhead is also relatively large if the address is complex enough, such as is the case with a complex polygon requiring multiple coordinates.

There is a 2010 Internet-Draft describing an IPv6-based geographic unicast and multicast address format [18]. This approach divides the globe into 4 (unequal) sections and can address positions with an accuracy of 0.00001 degrees. Regions can only be addressed with 12 different sized rectangles offset from the initial address position. This system is not flexible enough for our use-cases due to its focus on unicast and limit area sizes.

There are multiple proposals regarding the use of multicast addresses to represent geographic regions [8, 19]. In these proposals a multicast address maps to fixed geographic area. This has the downside of a fixed mapping, making it difficult to address areas partially crossing the border of these areas.

While not an addressing system for geographic routing we should note the existence of Geohash [20]. Geohash is a system that splits the world into increasingly smaller nested halves. This approach can be repeated for as long as needed to reach any level of precision. These halves are addressed in such a way that they mostly share the same prefix as neighbouring regions, although

this does not hold along the Greenwich meridian, the 180 degree meridian and near the poles. This addressing method is limited to a single fixed rectangle shape and as such not flexible enough to our goals.

### 2.2.3 Geographic Routing

Geographic routing is an essential part of any geocast system. It routes the packets from their source to their destination (area). In this section we will describe three different types of geographic routing protocols: those for fixed networks, DNS-based approaches, and ad-hoc network systems.

#### Fixed Networks

In this section we describe geocast protocols built for more or less fixed networks where nodes or routers are relatively static and the network topology does not change much. This does not necessarily only refer to a wired network, there could also be fixed wireless links in the network.

Navas and Imielinski proposed a system that adds an overlay network on top of the existing IP infrastructure to support geocasting [5]. This system specifies three types of devices: 1) geographically aware routers (GeoRouters); 2) special entry and exit points to the geographical routing network (GeoNodes); 3) geographically aware hosts (GeoHosts). Addressing in this system is based on the WGS-84 coordinate system used by GPS. GeoRouters form an overlay network by tunnelling over the traditional IP network, and route geocast messages between themselves based on the destination polygon in the packet header. Georouters form a hierarchy, where messages are sent to routers serving increasingly large areas until a router serving a smaller area containing the destination is found. The GeoNode stores geographic messages and will periodically multicast them on its downstream interfaces, which can be wired or wireless. Devices will have to subscribe to the relevant multicast group to receive geocast messages. A GeoHost is the geocast daemon running on end-devices. This GeoHost can send and receive geocast messages, and also keeps track of the host's current location. This system was published as an RFC [21] and received experimental status by the IETF, but has not have further updates since its publication. Problems with this approach are that they essentially form an overlay network and are locked in a strictly hierarchical structure, which might lead to problems when multiple networks cover the same area. Another problem is that the addressing system used by this approach is based on coordinates, of which we described the problems in the previous section.

The same authors proposed a system to use a worldwide geo-network to send and receive messages from geo-networking capable devices [8]. In this



paper, the authors propose to divide the world into 3-dimensional partitions with their own individual multicast address. Communication to and from these so called ‘dataspaces’ is based on multicast trees between responsible routers. This addressing approach would require a unique multicast address for each area. While this approach is certainly not impossible with the IPv6 addressing space, it would lead to a large number of routing table entries.

GPS-based Multicast, proposed in [19] by the same authors as [8], is based on smallest possible sections (an atom) that can be mapped to a multicast address. Each atom and each possible partition are assigned a multicast address. A sender would have to know the address of its destination area to target it. This system allows for a geocast system that can work in any IP multicast-capable network. The main addressing disadvantage is that the target region must be predefined.

Navas and Imielinski have also proposed a routing algorithm that uses full network knowledge to route geographic packets [22]. In this paper, the cost of calculating the forwarding links for a packet is evaluated. It is concluded that destination areas should be simplified in forwarding routers to reduce routing time at the cost of accuracy.

Durr and Rothermel propose another overlay routing system [13]. This system uses addresses where more specific destinations share a prefix with a larger area, so routing can be more efficient. They base their addressing system on symbolic names: countries contain smaller regions, containing cities and so on. Geocast routers are associated with a symbolic location as its designated router. This router is also responsible for all smaller locations if no more specific router exists. The routers form a hierarchy based on the hierarchy of the symbolic addressing system. Initially messages need to traverse this hierarchy towards their destination, but ‘short-cuts’ can be established to forward over a more direct path. The main downside to this approach is the initial need to traverse the hierarchy of routers. This also makes it problematic for such a system to support multiple networks in the same area, as this could break the strict hierarchy.

To the best of our knowledge, there are no more recent papers concerning geocast in fixed networks. More recent publications generally focus on the specific ad-hoc wireless use case, specifically vehicular networks or sensor networks.

### **DNS-Based Solutions**

Another approach to geographic routing is using an application layer approach where an application queries the Domain Name System (DNS) for a geographic area. The application can then use unicast to send a packet to a set of devices in the destination area.

RFC 2009 describes a solution where an application can query DNS for all base stations covering a certain area using the .geographic top level domain [21]. These queries could be either for a specific name such as the city hall of a city or for a region inside a city described by a polygon. Message delivery can be either through unicast towards each base station or by all base stations joining a temporary multicast group. A third options is that the system only returns a single unicast address for a centrally located base station, which in turn will deliver the message to other base stations in the area.

A later paper proposes a modified DNS system that can perform reverse location queries called eDNS [14]. The system can be queried for all records that are in a specified area. This system allows the implementation of an application layer geocast solution. A device first looks up all the addresses of the devices in the region it wants to geocast to. When the query returns the data can be sent to all those devices. This approach has the benefit that it can be deployed on top of the existing IP infrastructure and can work with both IPv4 and IPv6. The main downside of this approach is that every device in the target area needs to be addressed individually and all devices must periodically update their location in a central database. It was shown that this solution can work at scale [23], but the unicast overhead and requirement of a more or less central ‘bookkeeping’ system remain.

### Wireless Ad-Hoc Networks

Wireless ad-hoc networks are networks where connections are not necessarily fixed, but can change all the time. In these networks most, or even all, of the nodes are mobile. A general overview of geocast protocols focusing mostly on ad-hoc network can be found in [24]. Vehicular ad-hoc networks (VANET), where vehicles close to each other can communicate directly, are an example of an ad-hoc network. A large body of work exists describing geocast solutions specifically for this use-case, extensive overviews of geocast protocols specifically for VANETs can be found in [25], [26] and [27]. While we almost exclusively focus on fixed network in this thesis we will describe a few well known ad-hoc geographic routing protocols in this section and describe why they are generally not applicable to the fixed network use-case.

A well-known example of a geographic routing protocol for ad-hoc networks is GeoTORA [10]. When a node in the network needs to geocast a message it broadcasts a query with the request for the destination nodes. The destination nodes send a message back, allowing the original requesting node to know the forwarding hop towards the geocast area. The mobile nature of these ad-hoc networks makes this kind of signalling a necessity to reach any sort of efficiency. We do not have this problem in wired networks, allowing for the possibility of route distribution beforehand.

Another example is Greedy Perimeter Stateless Routing (GPSR) [12]. In this algorithm, traffic is routed to nodes that are located closer to the destination area than the transmitting node. This approach is seen often in geographic routing solutions for ad hoc networks. The downside for a wired environment is that the location of routers is not necessarily correlated with the direction a packet needs to travel to reach its destination.

An interesting grid-based ad-hoc routing system is the Grid Location Service (GLS) [11]. This system has nodes keep track of each others location in a distributed manner, where nodes closer to a node are more likely to know its location. The geographic routing layer of GLS addresses nodes based on their current location and uses a distance-vector protocol with two hop knowledge to route packets to their destination. This protocol allows nodes to lookup the location and send packets to a specific other node. This requires the location of all possible destinations to be known somewhere in the network, while we would like to address any number of nodes in a given area preferably without any knowledge (of the potentially large number) of nodes in this area.

The European vehicular networking standard ITS-G5 defines two modes of ad-hoc geographic forwarding: Contention Based Forwarding (CBF) and greedy forwarding [28]. In the greedy system a vehicle uses its knowledge of the locations of neighbouring vehicles to select the vehicle that has the best position to further forward the packet. The CBF method works by delaying transmission of received packets based on the distance towards the destination area, devices closer to the destination will have a shorter timer. Devices do not forward packets if they have received it twice, leading to a system that forward a packet ever closer to its destination. We will further explain the CBF method in Chapter 7. Due to their strong reliance on forwarder location and the delay introduced by CBF these methods are not suited for fixed network geocast.

The fundamental difference between wireless ad-hoc protocols described here and wired networks, is that in wireless ad-hoc networks there is a strong relation between the physical distance between two nodes and the network distance, e.g., number of hops between nodes, data rate of a link, or error probability of a link. In a fixed network this relation is only very limited. In a wireless setting, geo-routing protocols try to be ‘greedy’, routing packets to the neighbour node nearest to the destination such as in GPSR [12]. In a fixed network this correlation between the direction a packets needs to travel in to reach its destination and the physical location of the next hop router does not necessarily exist. For these reasons, the ad-hoc protocols described in this section are not suited for a fixed network deployment.

## 2.3 Routing

Routing refers to a collection of forwarding actions by multiple routers which together route a packet from a source to a, in the geocast case, group of destinations. Each router will need to have some knowledge about to which of its neighbours a packet needs to be forwarded to reach a destination. Based on this information it can make a forwarding decision. This process will ultimately deliver a packet to its destination.

In this thesis we will generally refer to routing as the entire process of forwarding a packet over multiple hops from its source to its destination(s). When we speak about forwarding we specifically refer to an action made by a single router on how to forward a packet towards one or more next hops.

In this section we will describe high-level general routing concepts. These will be needed to explain some of the design decisions we have made in Chapters 5 and 6. We will start by describing unicast routing, followed by multicast.

### 2.3.1 Unicast

Unicast refers to sending a packet from a single source to a single destination. It is the most used forwarding method in the Internet today, almost all Internet traffic in everyday use is unicast. Routers and other forwarding devices can simply forward these packets on the shortest path towards the destination. As packets are only forwarded on one link per forwarding device, the source is not relevant for the forwarding operation.

When multiple hosts need to receive the same information, the source will need to send multiple packets to all these destinations. When destination hosts share similar paths from the source this leads to multiple identical (apart from the destination address) packets traversing the same links, as shown in Figure 2.1a.

We describe unicast here as we will use some of the underlying mechanisms common to unicast routing protocols in our design of a geocast routing protocol in Chapter 5.

### Distance-Vector Routing

Distance-vector routing algorithms base their forwarding decisions on limited network knowledge. Routers using an distance-vector routing protocol generally only know the cost to reach other routers in the network and which neighbour to forward packets to.

In a simple distance-vector protocol, routers will exchange cost information with each other. After several rounds (depending on network size) of exchanging information, each router will know of every other router in the network and the cost to reach them. Note that routers have little information on the actual

network topology, they only know who their neighbours are and the cost to reach other routers through those neighbours.

Due to the periodic nature of information exchange between routers and the limited network knowledge changes in the network can take a long time to propagate. During this time it is likely that routers make incorrect forwarding decisions which could even lead to packet loss.

### Link-State Routing

Link-state routing protocols forward packets based on full network knowledge, each router has full topology information. Routers exchange so called link-state messages with their neighbours that describe the state of all links known to them. This eventually leads to all routers in a network knowing about all links in the network. Based on this information a router can compute an optimal route towards each other router in the network.

When a router receives a packet, it can simply lookup the shortest path next hop towards the destination and forward packets on the corresponding interface. An example of a link-state routing algorithm is Open Shortest Path First (OSPF) [29]. The downside of this type of routing algorithm is that full topology information needs to be distributed and kept up-to-date for all routers in a network. The benefit is that topology changes have less impact on routing performance compared to approaches based on less topology information.

### Border Gateway Protocol

The Border Gateway Protocol (BGP) is the protocol that connects different networks together to form the global Internet [30]. In contrast to the routing protocol concepts described before, BGP is designed to be an inter-autonomous system routing protocol. Its goal is to exchange route information between entire networks.

BGP carries information on the autonomous systems that need to be traversed to reach a destination. This gives each AS a path to all known destinations. This information allows packets to be routed without any loops towards a destination address.

#### 2.3.2 Multicast

In contrast to unicast, multicast is the sending of data towards a group of receivers. Multicast packets are sent from a single source to a so called multicast group. This group is identified by an IP address from a special IP address range (224.0.0.0/4 for IPv4 [31] and ff00::0/8 for IPv6 [32]). Multicast listeners will have to subscribe to the multicast group to receive it. This is a way to let routers in the network know they should forward packets with that

multicast group as a destination to that host. Among themselves routers build a distribution tree, how they do this depends on the implementation. The main benefit of multicast is that multiple destinations do not require multiple packets to travel over the same link. Routers can duplicate a single packet they receive to send it over multiple links when needed, as shown in Figure 2.1b.

Challenges for multicast exist mainly in the routing aspect. Routers need to keep track of which of its links require multicast traffic from a specific multicast address. Widespread use of multicast would also require coordination to ensure a multicast group is only used for a single application. Because of these problems multicast traffic is currently not routed over the global Internet. Multicast is mostly used internally in networks, for applications like video distribution (in IP-television systems for example).

Protocol Independent Multicast (PIM) is the most used multicast implementation in networks today. It has two modes, optimized for different amounts of multicast listeners in a network: Dense Mode (PIM-DM) and Sparse Mode (PIM-SM). PIM-DM is, as the name implies, optimized for a denser subscriber distribution.

PIM-SM is optimized for situations where relatively few routers in a network need to receive multicast packets. It functions by designating a router as a

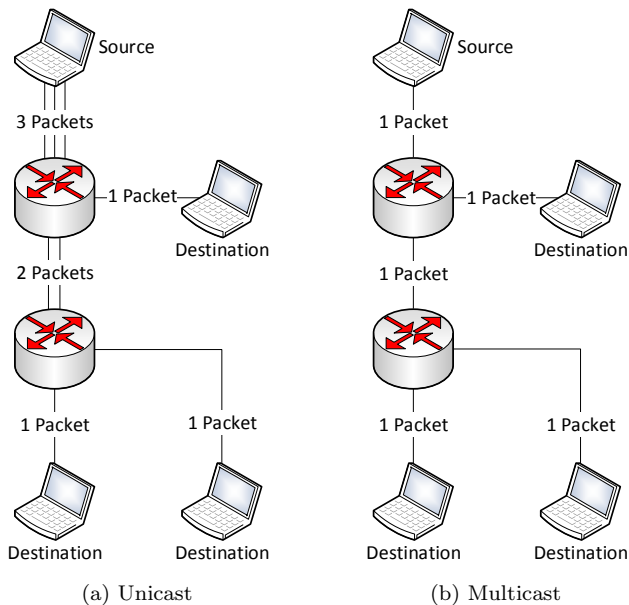


Figure 2.1: Unicast traffic vs multicast traffic

Rendezvous Point (RP) to which all traffic is initially sent [33]. This RP then forwards the packets towards the routers with subscribers. Over time the system switches to delivering packets directly from the source to a destination, leaving the RP based distribution tree.

PIM-DM works by initially flooding all multicast packets through the network [34]. Routers that do not have subscribers to a multicast group send a prune message upstream, which in turn stops the receiving router from sending the multicast packets with that group to that router. This establishes a shortest path tree from the source to all destinations.

While multicast might seem like a suitable candidate for geographic routing this is not the case. Normal multicast addresses are not flexible enough

## 2.4 Vehicular Networking

Vehicular networking refers to a collection of technologies related to networking vehicles. This ranges from systems that connect vehicles to the mobile network to systems that facilitate ad-hoc wireless communications between vehicles, so called Vehicular Ad-hoc Networks (VANETs). The most promising applications are in the safety and assisted driving area [4, 2], as vehicles will be able to share their speed and location with each other to increase safety and efficiency.

In general, communication in vehicular networks is often divided into three main categories:

1. Vehicle-to-Vehicle (V2V)
2. Vehicle-to-Infrastructure (V2I)
3. Infrastructure-to-Vehicle (I2V)

V2V refers to any communication that is between vehicles. An example would be a vehicle transmitting its current positions and speed to neighbouring vehicles. V2I refers to any form of communication where a vehicle sends data to some form of infrastructure. An example would be a vehicle notifying a central authority it was involved in an accident. I2V refers to the opposite situation where a message is sent from an infrastructure device to a vehicle. Notifying other road users of the accident would be an example. Sometime the term V2X is used to define communication from a vehicle to either another vehicle and/or infrastructure.

### 2.4.1 Vehicular Ad-Hoc Networks

VANETs are wireless networks consisting of vehicles and related devices. There is usually no (or limited) central authority but hosts decide between themselves

how to forward messages to their destination. There are currently two main (competing) standard for V2V communication: The North-American Wave [35] and the European ITS-G5. In both standards vehicles communicate using IEEE 802.11p [36], a standard specifically made for V2V and V2I communication.

### **IEEE 802.11p**

IEEE 802.11p is a wireless LAN standard used in, and specifically designed for, vehicular networking [36]. It is a standard in the well known IEEE 802.11 family of standards, better known as ‘Wi-Fi’ to the general public.

This standard allows vehicles to communicate with each other and other 802.11p compatible devices. They have their own reserved frequency range of 5.875 - 5.905 MHz in the EU [37]. The system uses 10 MHz wide channels and has a relatively low bit-rate compared to more well known 802.11 standards to increase reliability in the vehicular environment.

### **WAVE**

The US-based Wireless Access in Vehicular Environments (WAVE) standard uses IEEE 802.11p as its underlying transmission mechanism. WAVE only allows for relatively short ranged communications of up to two hops [35]. This design decision prevents vehicles from communicating with other vehicles that are far away through ad-hoc means.

Geocast through infrastructure could help route messages towards vehicles and at the same time extend the possible communication range. Extended communication range could be useful in situation like traffic accidents where traffic needs to be notified about the problem.

### **ITS-G5**

In the EU there is a competing standard: Intelligent Transportation Systems-G5 (ITS-G5). It uses the same link layer as WAVE (*IEEE 802.11p*), but does support multi-hop forwarding on the network layer including support for GeoNetworking [28, 38]. This standard supports geographic forwarding based on coordinates in all the categories mentioned before (geo-unicast, geo-multicast and geo-anycast). Multi-hop forwarding in this standard can be done through different methods but in the last chapter of this thesis we will focus on CBF, as this is most applicable for our application.

### **Road Side Units (RSUs)**

Road Side Units (RSUs) are devices with at least a wireless interface that are placed at or near a road (hence the name). These RSU can facilitate vehicular



communications in different ways. They can enable communications from devices further in the network towards the vehicles (I2V). An example would be traffic information from a centralized server that is sent to vehicles on a certain road. RSUs can also allow vehicles to send data to a system further in the network (V2I). An example of such communication could be that a vehicle reports a traffic accident it is involved in to a central authority, reachable on the network. RSUs can also act as forwarding points for V2V communication. In this situation the RSU will basically act as a static vehicle.

## 2.5 Evaluation Tools

In this section we will describe some of the tools we use in our evaluations. We describe the main source of the network-graphs we use: the Topology Zoo, the graph software used to manipulate graphs: NetworkX, and the emulators or simulators used to evaluate our proposals: Mininet, SUMO and OMneT++. We briefly describe these tools to give the reader an idea of how they are used throughout this thesis.

### 2.5.1 Topology Zoo

The Topology Zoo<sup>1</sup> is a collection of network graphs of real world networks collected from several sources [39]. The authors have collected these graphs from multiple freely available sources, as such a large majority, but certainly not all, of these networks are research networks.

This dataset is especially interesting for us as almost all nodes in the graphs have geographic coordinates of their actual locations associated with them. We use these graphs throughout this thesis to evaluate our solutions on geographically scoped destinations.

### 2.5.2 NetworkX

NetworkX<sup>2</sup> is a graph library for the python programming language [40]. It allow users to create different types of graphs (the graph theory kind, not charts). The software represents the vertices and edges of a graph as python objects, which allows the software to be used for all kinds of evaluations. NetworkX also comes with functions to calculate shortest paths and several vertex properties such as node degree and node centrality.

In this thesis we use networkX in the evaluations of Chapters 4, 5 and 6. In all these chapters we make use of the features to import different graph formats

---

<sup>1</sup><http://www.topology-zoo.org/>

<sup>2</sup><https://networkx.github.io/>

to import the TopologyZoo graphs. For our evaluations in Chapter 4 and 5, we specifically use the shortest path and minimum spanning tree functions. For our evaluation in Chapter 5 we extend the graph objects generated with NetworkX to perform our evaluation.

### 2.5.3 Mininet

Mininet<sup>3</sup> is a tool that allows a user to easily create a virtual network on a single machine with nodes running real kernel and application code [41]. Mininet can also add switches to a network based on open vSwitch. Mininet supports openflow for routing, but we do not use this functionality as we only connect hosts directly.

The most important aspect of Mininet for us is that it allows us to easily set-up a large number of different network topologies. We simply import the TopologyZoo graphs from NetworkX into our Mininet scripts.

Mininet allows each virtual host to run any application that can run on the host Linux system. The mininet virtual hosts can be connected through virtual network interfaces. We use this in Chapter 6 to test our user-space routing prototype.

### 2.5.4 SUMO

The Simulation of Urban Mobility (SUMO)<sup>4</sup> software can simulate road networks and the vehicles on them [42]. This software allows a user to create road networks, ranging from single streets to entire cities. Intersections can include traffic lights.

The roads in SUMO can be populated with a large number of vehicles, which can also include public transport vehicles. The vehicles characteristics such as speed and acceleration can be defines by the user.

We use SUMO in a rather limited fashion in Chapter 7. Our use is limited to the simulation of a simple two-lane highway environment on which vehicles travel in a single direction.

### 2.5.5 OMNeT++

OMNeT++<sup>5</sup> provides a discrete event simulation environment and comes with support for a wide range of network technologies, ranging from link layer to application layer, built in [43].

---

<sup>3</sup><http://mininet.org/>

<sup>4</sup><https://sumo.dlr.de>

<sup>5</sup><https://omnetpp.org/>

Specifically, we use the Veins<sup>6</sup> framework for OMneT++ to simulate a vehicular network [44]. Veins offers an implementation of the WAVE standard for vehicular networking, which we will extend in Chapter 7 of this thesis. This framework comes with tools to couple the network simulator to the traffic simulation of SUMO, to provide a realistic traffic environment in which to simulate vehicular networking.

---

<sup>6</sup><https://veins.car2x.org/>



---

## Geographic Addressing

*In this chapter we design and evaluate a geographic addressing system based on the requirements presented in Chapter 1. We introduce addressing-specific requirements and describe the design of our addressing system. Our addressing mechanism allows efficient referral to areas of arbitrary size. The binary representation of our addressing mechanism fits in an IPv6 address and can be used for route lookup with simple binary AND operations. We evaluate our system by addressing random areas and calculating how accurately our system can represent them. We show that our addressing mechanism can be used to address areas accurately enough to be used as a mechanism to route packets close to their destination. The contents of this chapter are based on the work presented in "An Efficient Geographical Addressing Scheme for the Internet" [45].*

1. Introduction	
2. Background & Related Work	
3. Geographic Addressing	
4 Geographic Forwarding Trees	
5. Geographic Routing Algorithm Design	6. Geographic Routing Implementation and Evaluation
7. Infrastructure Assisted Contention-Based Forwarding for Geocast	
8. Conclusions and Future Work	

## 3.1 Introduction

In the previous chapter we have described past proposals for geocast and how they relate to the current Internet infrastructure used around the world. As explained before, functional network-layer geocast could enable efficient geographically scoped broadcast, which in turn gives efficiency benefits to a whole set of applications, including applications specifically focused on vehicular networks. An example of such an application would be location-dependent weather warnings or notifications of accidents on the road ahead of a vehicle.

Geographic addressing can currently be done using overlay networks or specially designed multicast groups. These both have the downside of requiring extra infrastructure and extra overhead. This overhead comes in the form of extra signalling (in the multicast case) and reduced network capacity in the case of overlay networks. For geocast to be low overhead and low latency it should ideally be included in the network layer so that forwarding decisions can be made in routers.

In this chapter, we answer our first research question: *How can arbitrary-sized areas be efficiently addressed?*. We present an addressing method that can be used to efficiently address arbitrary areas with low computational cost. The resulting address fits in the IPv6 address space and can be used as a destination address for a packet and as a route entry for routers. Route lookup can be performed based on prefix matching which is similar to that already used in the Internet today.

The main contribution of this chapter is a geographic addressing method in which neighbouring areas can be specified with a single address. This addressing method has a binary representation that can be used to perform route lookups based on prefix matching, avoiding the need to do costly calculation on destination polygons [22].

The structure of this chapter is as follows: In Section 3.2 we describe previous work related to geographic addressing. We describe the requirements we designed our addressing system around and how they relate to the related work in Section 3.3. In Section 3.4, we describe the design of our addressing system. In Section 3.5 we evaluate how accurately our system can address arbitrary areas anywhere on the planet. Section 3.6 explains the applications of our addressing scheme in the Internet. Finally, we summarize and explore possible directions of future work in Section 3.7.

## 3.2 Related Work

The original geocast proposal addressed the destination of a packets based on their coordinates [5]. In a later paper the authors of the geocast proposal propose to divide the world into 3 dimensional partitions with their own

individual multicast address [8]. Communication to and from these so called ‘dataspaces’ is based on multicast trees between responsible routers. This addressing approach would require a unique multicast address for each area. While this approach is certainly not impossible with the IPv6 addressing space, it would lead to a large amount of route table entries.

GPS based Multicast is based on smallest possible sections (an atom) that can be mapped to a multicast address [19]. Each atom and each possible partition are assigned a multicast address. A sender would have to know the address of its destination area to target it. This system allows for a geocast system that can work in any IP multicast capable network. The main addressing disadvantage is that the target region must be predefined.

In 2010 an Internet-Draft was published describing a IPv6 based geographic unicast address format [18]. This document also specifies a geographic multicast format. The unicast format splits the globe in 4 sections, 3 within  $\pm 60$  degrees latitude of the equator (each 120 degrees of longitude) and the fourth covering the poles. This format uses 45 bits to address a region down to squares with edges of 0.000057 degrees (6.4 meters on the equator) with one corner specified to an accuracy of 0.00001 degrees, but it can only address such squares of 12 different sizes. The multicast format is based on the unicast format, using the unicast location part in the PI prefix to identify a region. Due to the limitation of to a set of fixed sized rectangles this system would not be accurate or flexible enough for our use-case.

The authors of [46] propose a system based on WGS84 coordinated and extending it with height information. They also propose adding identifiers to the destination to enable addressing down to individual rooms in buildings. This approach offers great flexibility but relies on routers calculating area overlap and comes at the cost of a relatively large packet overhead that depends on the complexity of the destination.

While not replacing an ip address directly, the eDNS platform is a modified DNS system that can perform reverse location queries [14]. The system can be queried for all records that are in a specified area. This system allows the implementation of an application layer geocast solution. A device first looks up all the addresses of the devices in the region it wants to geocast to. When the query returns, the data can be sent to all those devices. This approach has the benefit that it can be deployed on top of the existing IP infrastructure and can work with both IPv4 and IPv6. The main downside is that every device in the target area needs to be addressed individually and all devices must periodically update their location in a central database. This solution requires the DNS infrastructure to handle more queries and to be updates regularly, which will likely lead to scalability issues.

Geohash [20] is a method for efficiently indexing geographical coordinates with arbitrary precision. While Geohash is not directly related to geocast

routing, it is an interesting geographic addressing approach to look at. The Geohash system splits the world into halves based on latitude and longitude and represents them in a short form. Locations near to each other share a common prefix although this does not work near the Greenwich meridian, the 180 degree meridian and the poles (due to the closeness of the longitude coordinates). Due to the nature of Geohash to represent nearby coordinates with identical prefixes and the ability to do this with arbitrary precision. The downside is that this addressing method is bound to fixed-shape rectangular areas and does not share the same prefix for all neighbouring areas. Despite these downsides, we feel that Geohash is an interesting concept to explore for geographic addresses. We will partially base our addressing system on increasingly smaller nested areas, somewhat like Geohash.

### 3.3 Addressing Requirements

To deliver packets from a source to a destination the packet must either contain the destination or some other form of identifier that corresponds to this destination. In IP-based networks such as the Internet this identifier is the host's IP address, which uniquely identifies a host. A special case is multicast traffic, which uses a reserved range of addresses. Packets with such an address as their destination are forwarded to a group of hosts that have subscribed to that address. Note that multicast is not supported Internet wide and is mostly used within networks for specific features such as television distribution over IP. To deliver geocast functionality we need to be able to efficiently address any region in a network. The specification of the destination should be as specific as possible. In this section we will describe the high-level requirements for our addressing system. This is followed by a discussion on different addressing formats and methods for packet identification, for which we both make a choice to base our addressing system on.

We consider the following requirements for such a system based on the requirements we previously specified in Chapter 1:

- **Accuracy.** The proposed system needs to be relatively accurate in large as well as small areas. Inaccuracy would cause messages to be delivered to locations that are not within the destination area, or in the opposite case not to be delivered at all.
- **Minimal delay.** Routers will need to be able to quickly forward packets, preferably by a system similar to currently used unicast longest prefix matching. Large tables containing complex forwarding entries should be avoided.



- **Scalability.** Scaling to a worldwide system should be possible without impacting routing performance. It should be possible to aggregate addresses to minimize the growth of route table entries.
- **IPv6 compatibility.** Ideally we would not make any major changes to the existing Internet routing infrastructure. Defining an address format that is compatible with IPv6 would allow an addressing system to function while only modifying forwarding behaviour in routers. This will likely limit our addresses to a 128 bit addressing space.

### 3.3.1 Choosing an Addressing Format

Geocast addresses should somehow represent the geographic destination area of a packet. To fulfil the requirements we have, the most straightforward addressing solution is to base the destination on geographic coordinates. With such a system a point is represented as a (latitude, longitude) pair. To address the entire planet with any accuracy (in the meter range) we would need 8 bytes of storage space for both latitude and longitude (16 bytes per coordinate).

Depending on the specifics of the destination area this information describing the area can take up a significant amount of space in a packet. We have to take the minimum MTU size of the Internet into account when deciding on a format. According to RFC 791 the minimum MTU any IPv4 host should support is 576 bytes [47], no minimum link MTU is mentioned. The minimum path MTU for the IPv6 Internet is set at 1280 bytes [48]. In the rest of this document we will only consider the IPv6 Internet, as IPv6 support was a requirement. For efficiency reasons the destination area should not be significant compared to the 1280 bytes, otherwise the possible throughput for geocast traffic would become low compared to other routing methods.

Some examples of added header size using simple coordinates to define a destination area:

- **Circle:** A circle can be encoded as a centre point with (latitude, longitude) and a radius. This would result in 16 bytes for the centre and 4 bytes for the radius, not problematic for our size limit.
- **Rectangle:** A rectangle can be encoded with two coordinates (either set of opposing corners). This would require 32 bytes to accurately address. We could also choose to only use one coordinate and add a 'height' and 'width', requiring 24 bytes.
- **Polygon:** A polygon can be encoded as multiple points with the assumption it is closed. A sufficiently complicated polygon could cause significant overhead.

If we assume a traditional IP header is still needed to route a geocast packet over the internet all location information is extra overhead. Compared to a IP header size of 20 bytes (IPv4) or 40 bytes (IPv6) even the circle representation causes relatively significant overhead. Even if we ignore the overhead caused by this extended header we are still left with extra overhead in the forwarding operation. routers will have to calculate overlap between these representations and the type of area in the routing table. This could introduce a significant per packet forwarding delay.

Another option would be to somehow ‘abstract’ the location to some non-coordinate based format. This can be done in different ways such as making a pre-defined mapping from an identifier to a actual location. Such an approach was investigated in [8]. These approaches require that the mapping is know at routers and end-hosts, which in turn requires some form of central authority. It does have the benefit of allowing arbitrarily shaped and sized areas to be addressed.

Another interesting abstraction is GeoHash [20]. Geohash is a method for efficiently indexing geographical coordinates with arbitrary precession. While Geohash is not directly related to geocast addressing, it is an interesting approach to look at. The geohash system splits the world into halves based on latitude and longitude and represents them in a short form. Locations near to each other share a common prefix although this does not work near the Greenwich meridian, the 180 degree meridian and the poles (due to the closeness of the longitude coordinates). Due to the nature of Geohash to represent nearby coordinates with identical prefixes and the ability to use arbitrary precision, we feel that Geohash is an interesting concept to explore for route-able geographic addresses.

Using some form of abstraction that uses less space than the actual coordinates will always come with a cost. A shorter representation will always be less accurate or offer less flexibility in the shape and exact position of the addressed areas. Ideally we should come up with a system that finds a middle ground between these trade-offs wile keeping into account the requirements.

### 3.3.2 Choosing a Packet Identification Method

There are multiple ways in which we can link a packet to a destination. Based on the requirements discussed Chapter 1 it is likely best to address each packet individually like in IP routing. For completeness we will briefly discuss this option, and two other possible options in this section:

1. A Unique identifier that corresponds to a predefined destination. In this approach packets are send to a limited set of pre-defined regions.
2. A Unique identifier and the first packet of a stream also contains the

destination area. The first packet in a stream contain a description of the destination area plus some unique identifier for the following stream. Further packets are forwarded based on the destination described in the first.

3. The destination area is encoded into every packet. Similar to ‘normal’ IP addresses, each packet contains a description of the destination area.

The first option corresponds to the predefined multicast scenario, such as GPS based Multicast [19], mentioned in Section 3.2. A unique identifier is matched to a specific destination region. The solution would require each router to have a table mapping the identifier to the correct location. This solution is inflexible and not very scalable unless some smart encoding method is found for the identifier.

The second option would require an initial packet containing either the exact destination area or some abstraction plus an identifier that uniquely identifies the packet stream. Following packets can then only carry this identifier. This would require a router to keep track of all current streams, but the possibility of high accuracy with limited overhead might be worth it. The overhead for this method would depend on the number of packets in a stream. For streams containing a low number of packets the overhead would relatively be large, but streams containing many packets would have small overhead.

The third and final possibility that is discussed here, is to always include the destination region in all packets. This allows the route taken to the destination to change for a data stream which is impossible in the first approach described. It is the option that is closest to the way the IP protocol functions. The downside is the extra overhead that results from possibly complex addressing system. If overhead becomes a problem it might be needed to choose a simpler destination specification such as a circle. The downside would be the decrease in accuracy [22].

Based on the different approaches to addressing we have seen in the related work and the short summary of the options available we believe that a approach with per packet addressing and a compact addressing format would be best.

Directly embedding the destination in the packet has some benefits over predefined addresses such as addressing in data spaces [8]. The set of destinations would be limited and adding new destinations (a new road for example) would require some sort of central authority. Another consideration is the flexibility of the addressing, a destination is unlikely to be always neatly overlapping with a pre-defined area. Ideally an geocast address should just replace the IP address in the packet header, which would lead to less overhead compared to other geographic addressing systems or equal overhead to multicast addressing.

Due to the above reasons we will design our addressing system for the situation where each packet contains the full destination specification in its

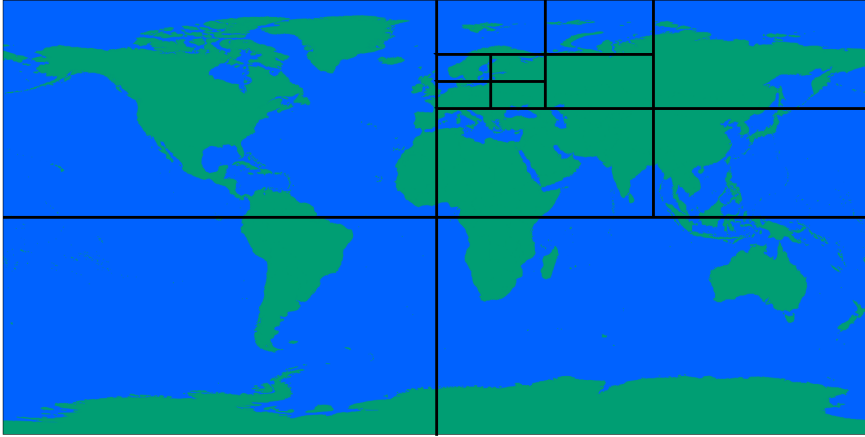


Figure 3.1: Rectangles enclosing northern Europe

header. This is also the case in most of the previously discussed related work. If we make use of the IPv6 addressing format there should be no extra overhead and we keep the option to change routes and will not be affected by loss of the initial packet.

### 3.4 A Geographic Addressing System

In this section, we will describe the method we use to address sections of the planet. This method is designed to be as accurate as possible while allowing routers to calculate coverage in an easy way equal to prefix matching.

To represent a single section on the planet we divide the Earth into four sections based on the latitude and longitude of the World Geodetic System 1984 (WGS 84) projection [17]. The latitude ranging from 90 to -90 (North to South) and longitude from -180 to 180 (East to West). The resulting four rectangles (which we refer to as level 1) meeting at (0,0) are themselves divided into four sections. This process continues for the number of steps that are needed for an optimal description of the area that we are interested in. Due to the nature of the WGS 84 projection, the rectangles in our system have half the height of their width in terms of degrees. Figure 3.1, shows a overview with 4 levels of these rectangles enclosing northern Europe.

The major difference between our approach and GeoHash [20] is in the way we will number these rectangles. As we will examine in Section 3.4.2, it is possible to combine neighbouring rectangles into a single area description. This allows us to address arbitrary areas on the globe based on these rectangles.

With the ability to combine neighbouring rectangles into a single description we can resolve one of the main limitations of addressing single rectangles.

We will further explore our addressing system by explaining the different sizes our rectangle descriptions can have. This is followed by describing the method we use to find the rectangle description that contains a single point, followed by a description of how we address multiple rectangles at the same time. We end the description of our system by explaining our binary representation.

### 3.4.1 Size of the Rectangles

Rectangles that are split into four identical sections logically have half the height and width of their parent rectangle in terms of degrees. We define the level of a rectangle as the number of times we need to split the initial rectangle to reach it. With this definition there are 4 rectangles at level 1, 16 at level 2 and 64 at level 3 (Figure 3.3). The number of rectangles in a certain level is thus given by Equation 3.1. In this equation,  $N$  is the number of rectangles that are present at a certain *level*.

$$N(\text{level}) = 4^{\text{level}} \quad (3.1)$$

The resulting height and width of a rectangle at a specific level depends on the size of the earth and the location of the rectangle on it. Due to the size in meters depending on the latitude, all calculation related to size are done in degrees. Equation 3.2 gives us the width of a rectangle at a specific *level* where  $W$  is the width. Equation 3.3 gives us the height of a rectangle at a specific *level*, where  $H$  is the height.

$$W(\text{level}) = \frac{360}{2^{\text{level}}} \quad (3.2)$$

$$H(\text{level}) = \frac{180}{2^{\text{level}}} \quad (3.3)$$

As noted, the actual size of a rectangle depends on the latitude of the rectangle. Assuming the earth has a circumference of 40,075 km this gives us an upper bound on the longitudinal size, or width, of a rectangle as  $40,075/360 = 111$  km per degree. In practice, the width of a rectangle will be smaller per degree unless one of the lines is exactly on the equator. The latitudinal size, or height, of a rectangle is consistent over the globe.

### 3.4.2 Finding the Rectangle Enclosing a Single Point

If we want to be able to represent any area in a unique way we need to be able to identify all rectangles. We do this by numbering them. We number the

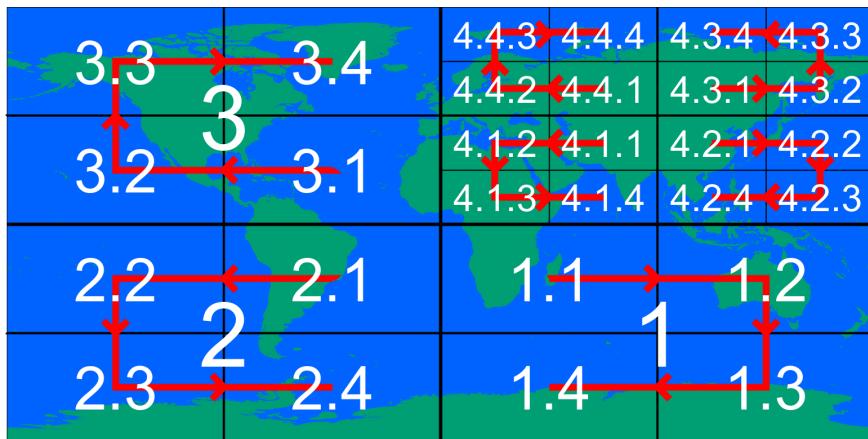


Figure 3.2: Numbering the rectangles

rectangles that result from the process described before in a way resembling a horseshoe. We start numbering from the centre of the parent rectangle, such that the new rectangles are numbered *1*. We then move to the side with *2*. Number *3* is the rectangle below (in this case) and we move back to the inside to place number *4*. The initial rectangle (that covers the entire planet) is numbered as if a parent rectangle exists of which it is on the top left. We show the first 3 levels of this numbering system in Figure 3.2, with the red line of the horseshoe pattern and arrows showing the direction. Note that we omitted the line for the top-level rectangle 4, so we can clearly show the pattern repeating inside it. We separate the number of each level with a dot in such a way that the lowest level rectangle is represented by the right-most number.

This system can be described mathematically by using Equation 3.4, 3.5 and 3.6. We input the *longitude* and *latitude* of a location in Equation 3.4 and 3.5 respectively, together with the *level* we want to know the number of. We now use the resulting *x* and *y* values in the table *L* given by Equation 3.6 to find the corresponding number. Table *L* gives a rectangle number based on an *x* and *y* value where  $0 \leq x \leq 3$  and  $0 \leq y \leq 3$ . Note that the numbering pattern visible in the lookup table of Equation 3.6 is reflected in Figure 3.2.

$$x(lon, level) = \left\lfloor \frac{lon + 180}{W(level)} \right\rfloor \pmod{4} \quad (3.4)$$

$$y(lat, level) = \left\lfloor \frac{90 - lat}{H(level)} \right\rfloor \pmod{4} \quad (3.5)$$

$$L = \begin{array}{cc|cc} 3 & 4 & 4 & 3 \\ 2 & 1 & 1 & 2 \\ \hline 2 & 1 & 1 & 2 \\ 3 & 4 & 4 & 3 \end{array} \quad (3.6)$$

To find a complete representation of a rectangle with a certain level of precision the equation will have to be used once for each level of accuracy. As noted earlier, this description contains the number of each rectangle until the lowest level separated by a dot. A formal method to find a complete description of a point can be found in Equations 3.7 and 3.8. In Equation 3.7,  $S_{level}(lat, lon)$  gives the number on a specific level, with  $L$  the matrix in Equation 3.6 and  $x, y$ , given by Equations 3.4 and 3.5 respectively, determine the row and column in  $L$ . Equation 3.8 represents the entire description of a rectangle. In this equation  $lat, lon$  are the latitude and longitude and  $level$  is the maximum level. Algorithm 1 provides the pseudo-code to perform this lookup using Equations 3.2, 3.3, 3.4, 3.5 and 3.6.

$$S_{level}(lat, lon) = L_{x(lon, level), y(lat, level)} \quad (3.7)$$

$$S(lat, lon, level) = (S_1(lat, lon), S_2(lat, lon), \dots, S_{level}(lat, lon)) \quad (3.8)$$

The benefit of this numbering scheme is that rectangles on the same level will neighbour rectangles with the same number in neighbouring parent rectangles.

---

**Algorithm 1:** Coordinate lookup algorithm
 

---

**Input** :  $lat$  and  $lon$  (Coordinates of point);  $maxLevel$ ;  $L$  (Lookup matrix)

**Output** : List  $result$  with rectangle representation of length  $level$

```

1 List  $result$ ; // Initialize list
2 for  $level \leftarrow 1$  to  $maxLevel$  do
3    $x \leftarrow (lon + 180)/(360/2^{level}) \pmod{4}$ ; // Equations 3.4 and 3.2
4    $y \leftarrow (90 - lat)/(180/2^{level}) \pmod{4}$ ; // Equations 3.5 and 3.3
5    $number_{level} \leftarrow L[x][y]$ ; // Equation 3.6 lookup
6    $result.add(number_{level})$ ;

```

---

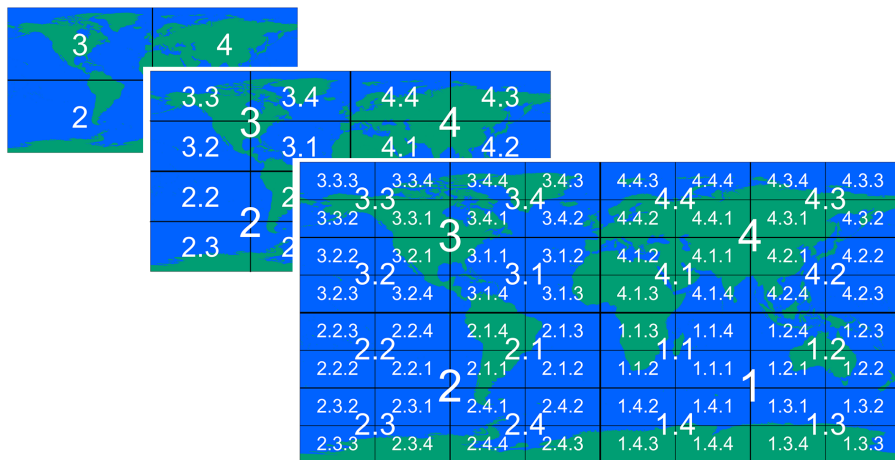


Figure 3.3: World map with rectangles up to level 3

As we will show later in this chapter, this feature is helpful in aggregating rectangles to cover arbitrary sized areas.

As mentioned before, we have chosen to represent a rectangle in this quaternary notion by separating each level with a dot. For example, a rectangle on level 2 with number  $3$  that has a parent rectangle with number  $1$  as its parent is represented by  $1.3$ . Figure 3.3 shows rectangles with their number up to level 3 drawn on a map of the world. Note that on each level rectangles with similar suffixes neighbour each other when their parent rectangles are neighbours.

### 3.4.3 Addressing Areas Using Multiple Rectangles

Enclosing arbitrary areas in just a single rectangle would lead to a very inefficient system as it would likely enclose much more than just the requested area. To solve this problem, it would be beneficial to address multiple lower level rectangles that together better describe the area. To do so we need a method to address multiple rectangles at once. Because the way the rectangles are numbered it is relatively easy to address neighbouring rectangles at once. Imagine an area we want to address stretching from the Netherlands to the centre of Russia. In Figure 3.3, we can see that we would likely need the rectangles  $4.4.2$  and  $4.4.1$ . We can now describe this in a single address as  $4.4.[1,2]$ . Numbers enclosed in brackets mean that both those rectangles are addressed on that level.

An area that crosses parent rectangles, such as the one covering the Nether-



lands and the UK would be described as [3,4].4.2. This notation means that we want to address both rectangle 3.4.2 and 4.4.2. We describe these rectangles as having level 3. We describe the number of levels that a rectangle description has more than the first level containing multiple rectangles as its depth: [3,4].4.2 has depth 3, while 4.4.[1,2] has a depth of 1. A formalised expression of combining two rectangles can be seen in Equation 3.9, where  $\oplus$  denotes the operation of combining two areas.

$$(S_1^A, S_2^A, \dots) \oplus (S_1^B, S_2^B, \dots) \rightarrow (S_1^A \oplus S_1^B, S_2^A \oplus S_2^B, \dots) \quad (3.9)$$

Due to the nature of this method to ‘mirror’ lower level areas into higher levels, the description regularly incorporates more areas than just the components it is constructed of. An example would be extending the area covering the Netherlands and the UK eastward. This would add 4.4.1 to the group leading to the area [3,4].4.[1,2], but this also includes the 3.4.1 area. However, the numbering method has been designed to avoid this as much as possible.

As we have shown, combining neighbouring rectangles that share a parent rectangle is trivial. It is possible to simply combine the addresses of two rectangles to get an address that contains both. However, this method does not guarantee that we also address the space between those rectangles. A complete address also needs to address all the rectangles between them.

A single edge of a rectangular area (a line) can be described by two points on the same latitude or longitude. As shown in Section 3.4.2, we can find the representation of these points at any level. To complete this line representation, we will however also need to include the rectangles that are between the rectangles that represent our start and end points. Consider the line between the points (60,-55) 3.4.1 and (60,55) 4.4.1. As we can see in Figure 3.3 the rectangles 3.4.2 and 4.4.2 are between them, to accurately represent the line we need a method to find these rectangles. To extend this idea to areas that also differ in latitude we also need to take the rectangles in that direction into account, instead of focussing on just the longitude.

We can modify Algorithm 1 to accept two coordinates and calculate the area between them. The distance of the two coordinates in the lookup matrix (Equation 3.6) is calculated by subtracting the values of Equation 3.4 and 3.5 for both coordinates from each other per level. We can now simply ‘walk’ over the matrix within the calculated range for each level and add the found numbers to our value for that level. Algorithm 2 shows the procedure in pseudo code. Note that we make use of Equations 3.2, 3.3, 3.4 and 3.5 again. It is important that coordinate 1 is the north-western corner of the rectangle and coordinate 2 the south-eastern.

An extra check is added to the algorithm to see if the parent rectangles do not border in the East - West (*line 9*) or North - South (*line 14*) direction.

**Algorithm 2:** Rectangle lookup algorithm

---

```

Input :  $(lat1, lon1), (lat2, lon2)$  (2 coordinates);  $maxLevel$ ;  $L$  (Lookup matrix)
Output: List result with rectangle representation of length level
1 List result; // Initialize list
2 for  $level \leftarrow 1$  to  $maxLevel + 1$  do
3    $x1 \leftarrow (lon1 + 180)/(360/2^{level})$ ; // Equation 3.4 and 3.2
4    $y1 \leftarrow (90 - lat1)/(180/2^{level})$ ; // Equation 3.5 and 3.3
5    $x2 \leftarrow (lon2 + 180)/(360/2^{level})$ ; // Equation 3.4 and 3.2
6    $y2 \leftarrow (90 - lat2)/(180/2^{level})$ ; // Equation 3.5 and 3.3
7    $dX \leftarrow (x2 - x1)$ ;
8    $dY \leftarrow (y2 - y1)$ ;
9   if  $|dX| \geq 4$  then // Are parents East-West neighbours?
10    |  $dX \leftarrow (dX \bmod 4) + 4$ ;
11  else
12    |  $dX \leftarrow dX \bmod 4$ ;
13  if  $|dY| \geq 4$  then // Are parents North-South neighbours?
14    |  $dY \leftarrow (dY \bmod 4) + 4$ ;
15  else
16    |  $dY \leftarrow dY \bmod 4$ ;
17   $temp \leftarrow 0$ ;
18  for  $y \leftarrow y1$  to  $y1 + dY + 1$  do
19    for  $x \leftarrow x1$  to  $x1 + dX + 1$  do
20      |  $temp \leftarrow temp \vee L[x \bmod 4][y \bmod 4]$ ; // Bitwise OR as
21      | shown in Section 3.4.4
22  result.add(temp);

```

---

When this is not the case (distance is greater or equal to 4) we need to make at least one ‘loop’ in that direction over the matrix in Equation 3.6 to ensure we cover all rectangles between the two points at that level. If we do not do this, rectangles on that level between the points might not be included in our description.

Algorithm 2 can give us the address of any region on the planet. We have made an online tool<sup>1</sup> with which the user can select any area on a map and get the generated address. This tool also allows the user to visualize the addressing system, displaying each level of a rectangle description as a coloured box.

<sup>1</sup><https://berndmeijerink.nl/geolookup/>

### 3.4.4 Binary Representation

One of the goals of our addressing format is to be compatible with IPv6. To meet this goal we need to be able to encoded our addresses into an IPv6 address. This format should allow routers to perform route lookups based on our rectangle representation and allow addresses to be aggregated into a single address. For these reasons we need to have a binary notation for our addressing system, which we will describe in this section.

Each level in our addressing format represents four possible rectangles. Because we want to address destinations that possibly overlap multiple rectangles, we cannot simply represent the numbers of the rectangles in a binary fashion. We represent each level as a 4 bit block: 1 = 1000, 2 = 0100, 3 = 0010 and 4 = 0001. Example: 2.4.2 = 0100.0001.0100. We show every possible single rectangle at level 1 in Table 3.1. Note that these rectangles are numbered as if the parent rectangle has its centre at the lower right corner. Lower level rectangles might be numbered starting from another parent rectangle, as explain in Section 3.4.2.

This example can still be represented with two bits per level, but once we need to combine multiple rectangles the four bit system becomes needed. If for example we would want to address the region surrounding (0,0) at level 3, this would require us to address the following rectangles: 1.1.3 (1000.1000.0010), 2.1.3 (0100.1000.0010), 3.1.3 (0010.1000.0010), and 4.1.3 (0001 1000 0010), or [1,2,3,4].1.3 (1111.1000.0010). The binary representation of rectangles that can be combined is a binary OR over the individual rectangles that cover the area. We show two level 2 examples in Table 3.2. Again note that the image is correct starting at level 1, and is not necessarily correct when used on lower levels.

With this binary representation it becomes possible to map the rectangle addresses to IPv6 multicast addresses. If we take the 16 bit multicast prefix into account, we will be left with 112 bits for addressing. With 4 bits per level this allows us to address a total of 28 levels. At the equator 28 levels corresponds to a rectangle of 14.9 by 7.5 cm. As this is a unrealistic small area to address it is likely that fewer levels can be used, saving space for other information in the address. We will evaluate the number of levels (and thus bits) needed for realistic scenarios in Section 3.5.

IPv6 geocast format														
Geocast prefix		Rectangle description (up to 18 levels)										Some useful data...		
FF	..	1,2	3,4	5,6	7,8	9,10	11,12	13,14	15,16	17,18	..	..	..	..

Figure 3.4: Suggested IPv6 addressing format














Rectangle	Bin	Dec	Hex	Image
[1]	1000	8	8	
[2]	0100	4	4	
[3]	0010	2	2	
[4]	0001	1	1	
[1,2]	1100	12	C	
[1,4]	1001	9	9	
[2,3]	0110	6	6	
[3,4]	0011	3	3	
[1,2,3]	1110	14	D	
[2,3,4]	0111	7	7	
[1,3,4]	1011	11	B	
[1,2,4]	1101	13	D	
[1,2,3,4]	1111	15	F	

Table 3.1: All single level rectangle representations

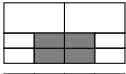
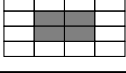
Rectangle	Bin	Dec	Hex	Image
[1,2].[1,4]	1100.1001	12 9	C9	
[1,2,3,4].[1]	1111.1000	15 8	F8	

Table 3.2: Multi-level rectangle examples

### 3.5 Accuracy

To determine the accuracy of the system we performed an evaluation with random areas on the world map. We generated a set of 10,000 random coordinates  $(lat_N, lon_N)$ ,  $N \in [1, 10000]$ . These coordinates have formed the basis of several sets of rectangles. These rectangles all have one of the generated coordinates as their north-western corner. The south-eastern corner is generated based on a random value between 0 and  $s$ . Sets  $((lat_N, lon_N), (lat_N + \Delta lat_{N,i}, lon_N + \Delta lon_{N,i}))$  were created with  $\Delta lat_{N,i}$  and  $\Delta lon_{N,i}$  randomly generated with a uniform distribution between 0 and  $s_i$ . For the values of  $s_i$  we use  $s_1 = 0.001$ ,  $s_2 = 0.01$ ,  $s_3 = 0.1$ ,  $s_4 = 0.5$ ,  $s_5 = 1$ ,  $s_6 = 2$ ,  $s_7 = 5$ ,  $s_8 = 10$ ,  $s_9 = 15$  and

$s_{10} = 20$  degrees. These ranges were chosen as they would cover most realistic scenarios we can envision. As reference: on the equator 0.001 degrees equals 111 meters and 0.1 degrees equal 11 km.

Additionally, 14 sets were generated with separate latitude and longitude ranges to ensure wide and tall rectangles. These values were as  $(\Delta lat_{N,i}, \Delta lon_{N,i})$  in degrees:  $s_{11} = (0.001, 0.0001)$ ,  $s_{12} = (0.01, 0.001)$ ,  $s_{13} = (0.1, 0.01)$ ,  $s_{14} = (1, 0.1)$ ,  $s_{15} = (3, 1)$ ,  $s_{16} = (5, 2)$ ,  $s_{17} = (10, 2)$  and the inverse sets. For reference, the smallest rectangles have a maximum size of 111 meters by 11 meters. The reason for these sets is that rectangles with a large difference between length or width are more difficult to fit into a single rectangle without sacrificing accuracy.

To calculate the accuracy we find the area covered by our generated rectangle and divide it by the area covered by the most accurate rectangle we can find in our numbering scheme. The result is a number between 0 and 1, 1 being exact coverage by the calculated rectangle. We refer to this value as the accuracy of the coverage area.

### 3.5.1 Lookup Level Accuracy

To find the optimal level at which the system can accurately describe most areas, we calculate the accuracy at different levels for each rectangle in our test set. We start at level 1 (One or multiple rectangles of 180 by 90 degrees), and go up to level 28. To give some context, a single level 28 rectangle measures about 14.9 cm by 7.5 cm on the equator. A level 18 rectangle would measure 152 by 76 meters on the equator.

We show the coverage accuracy for randomly generated rectangles of different sizes in Figure 3.5. Most tested sizes converge to an accuracy of around 0.32 before level 18. Only rectangles with a maximum edge size of 0.01 degrees (1.11km at the equator) and 0.001 degrees (11 meters at the equator) converge later. We can also see that larger areas need fewer levels before the accuracy does not increase any more, compared to smaller areas that require more levels to reach the same accuracy. This is a logical result of the amount of space a rectangle of a certain level covers. We can also conclude that the maximum accuracy seems to be 0.3, meaning that the target area covers 30% of the most accurate rectangle our system could generate.

Because one of our use-cases is vehicular networking we also look at rectangles that have a height that significantly differs from their width. We show these results in Figure 3.6. The results do not significantly differ from the more uniform rectangles. For the smallest possible rectangles we test here (with a maximum size of 111 by 11 meters on the equator) we observe that we need almost 28 levels to accurately describe it.

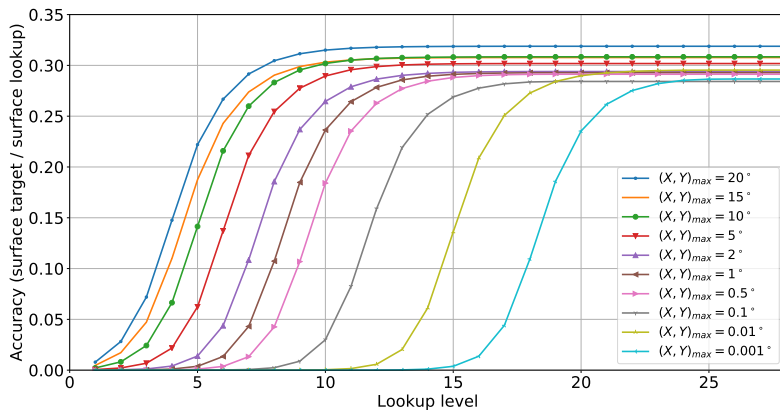


Figure 3.5: Accuracy per level of different sized rectangles

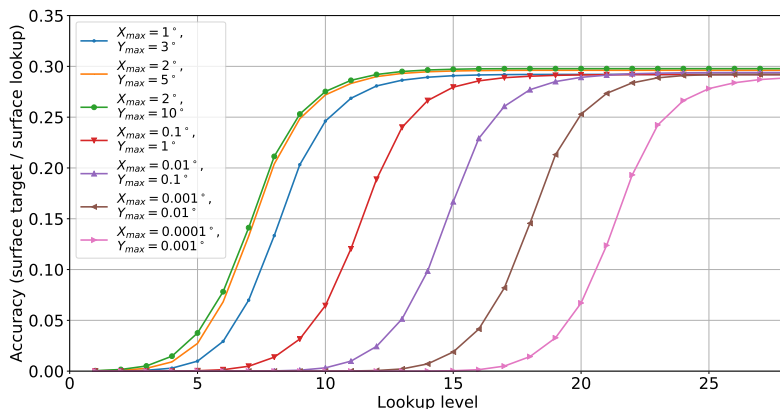


Figure 3.6: Accuracy per level of different sized long rectangles

### 3.5.2 Lookup Depth Accuracy

We look at the accuracy in term of level depth (the amount of levels containing multiple rectangles) to see what the accuracy gains are compared to simply using one rectangle. In Figure 3.7, we examine the depth needed to accurately represent an area. As mentioned before, we define the depth as the number of levels below the point at which the description starts covering multiple rectangles. At depth 0 the address represents a single rectangle, at depth  $n$  at most  $4^n$  rectangles appear of that depth. Figure 3.7a shows that a depth of 12 is sufficient to accurately cover even small areas. In the case of larger areas the

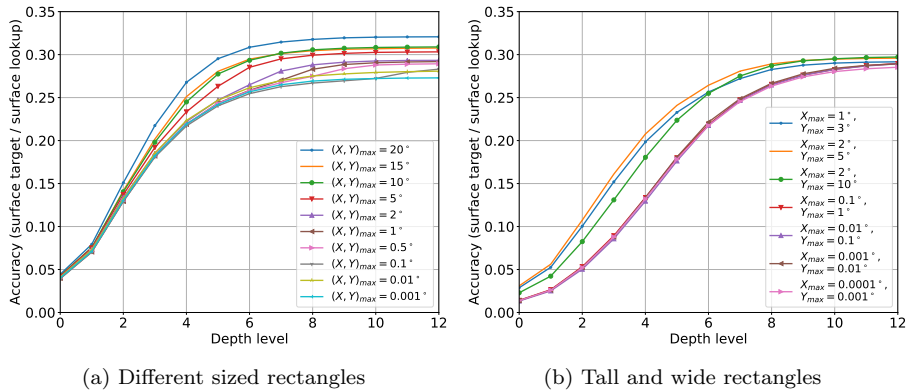


Figure 3.7: Accuracy per lookup depth

lookup can be limited to a depth of 8.

As mentioned, we have also specifically looked at very tall and wide rectangles to see their accuracy. Figure 3.7b shows that the performance of the description for these areas is equal to that of fully random areas. We need a few more depth levels to achieve the same accuracy in these cases, likely due to the fact that these areas cover many rectangles at the lower levels in one direction.

Based on Figure 3.7a and 3.7b we can conclude that calculating rectangles after depth 10 does not result in much further gain. We can also see that the tall and wide areas need on average a greater depth to gain the same accuracy compared to the completely random ones. This effect is caused by the fact that these areas require more smaller rectangles to accurately represent them.

### 3.5.3 Discussion

The results of our evaluation show that on average a single address in our addressing system can represent any area with 30% accuracy. This means that 30% of the area actually addressed by our system is part of the original destination. While this might not seem very accurate, keep in mind that if needed multiple addresses can be used to represent a complex region if needed. While this will cause some extra overhead on the network, it might be worth it if the region is significantly complex.

If we compare our solution to similar approaches such as Geohash, that can only represent a single rectangle, these will perform similar to depth 0 in Figure 3.7a and 3.7b, as it can only represent a single rectangle.

Significant gains can be made by addressing multiple rectangles, accuracy

is increased by a factor of 6 as compared to the single rectangle case. More accuracy can likely still be achieved by using multiple addresses at the cost of using multiple packets, which will likely share a mostly similar route, to reach a destination area.

### 3.6 Application to Network-Layer Geocast

In this section, we explore the applicability of our addressing approach to network-layer geocast routing and forwarding. As an example we will take a packet addressed to the country of the Netherlands.

A route entry can be represented by a single address, similar to the destination address of a packet. When a router receives a packet it will have to check if the packet's destination address has overlap with any of its route entries. This can be done through a simple bitwise AND operation on the destination address and the route entry. If the result of this operation has at least a single bit set to one in every block of four bits there is a match and the packet should be forwarded.

We take the country of The Netherlands as an example: The country covered by the rectangle with the North-West corner (3.3750933,53.6724828) and South-East corner (7.2230957,50.6266868). It is contained in the level 7 rectangle [4].[4].[2].[3].[2].[1,2,3,4].[1,4]. At level 7 these 8 rectangles represent an area 5.625 degrees wide and tall. The binary representation of this rectangle is 0001.0001.0100.0010.0100.1111.1001. This can be represented by the IPv6 address *ffxx:1142:4f90::*, where the initial two bytes (*ffxx*) should be replaced with a geocast-address identifier.

Now consider a router that has a route entry to this exact area. Also consider a packet addressed to the northern part of the country with the following destination address: [4].[4].[2].[3].[2].[1,2,3,4].[1], 0001.0001.0100.0010.0100.1111.1000 in binary and an IPv6 address of *ffxx:1142:4f80::*. The router can now perform the bitwise AND operation on the address and entry to obtain the overlap:  $0001.0001.0100.0010.0100.1111.1001 \wedge 0001.0001.0100.0010.0100.1111.1000 = 0001.0001.0100.0010.0100.1111.1000$ . The resulting value has at least a single bit set to 1 in each group of four bits, so the entry is a valid forwarding route for this address.

This approach puts the burden of most calculations at the sending system. This system will have to calculate the destination address of its packets based on Algorithm 2. Routers must initially calculate (or be provided with) the rectangle description of the area they cover, but no computationally intensive operations are necessary during forwarding.

We determined in the previous section that a description of level 18 can accurately cover most areas. A rectangle of level 18 can be encoded into 72



bits. This means that it can be used in a IPv6 multicast address while leaving 56 bits unused.

### 3.6.1 Route Aggregation

Due to the nature of our addressing system, multiple neighbouring addresses can be aggregated into a single address that covers a larger area. This allows a network to advertise the coverage area of its routers as a single address.

We can aggregate addresses in the manner we show in Equation 3.9, simply combining the addressed rectangles on each level. While this process works for any set of rectangles, it only makes sense if the rectangles overlap or neighbour each other. If they don't, the size of the addressed area would be much larger than the original rectangles.

Assume we have two routers that cover the campus of the University of Twente. One covers the west side of the campus which has address 4.4.2.3.2.1.1.2.4.2.1.4.2.[2,3], other the east side which has address 4.4.2.3.2.1.1.2.4.2.2.4.2.[2,3]. The university could advertise its coverage as 4.4.2.3.2.1.1.2.4.2.[1,2].4.2.[2,3]. In binary notation we can simply OR the two individual descriptions to obtain the combined rectangle description:

$$\begin{aligned} &0001.0001.0100.0010.0100.1000.1000.0100.0001.0100.1000.0001.0100.0110 \oplus \\ &0001.0001.0100.0010.0100.1000.1000.0100.0001.0100.0100.0001.0100.0110 = \\ &0001.0001.0100.0010.0100.1000.1000.0100.0001.0100.1100.0001.0100.0110 \end{aligned}$$

Networks will be able to advertise their coverage as an aggregated address instead of all individual coverages of their service area. Like traditional IP addresses this will greatly reduce the amount of information that has to be exchanged between networks for routing. This has the benefit that route tables are smaller, which ideally results in faster lookups and lower communication overhead between routers.

## 3.7 Summary & Conclusion

In this chapter, we have presented a method for addressing geographic regions using rectangles. Multiple rectangular regions can be addressed with a single address. This system allows areas of any size to be specified with relative accuracy. We show our approach is in some cases a factor 6 more accurate than other methods such as Geohash. The binary representation of our addressing method can fit within an IPv6 address with space to spare, while maintaining good accuracy.

Our representation can also be used for route lookup in an IP-based network. We show that our approach has potential as a system to transmit geocast packets close to their destination, where a more accurate but computationally expensive routing method can take over. We have also shown that our addressing format

can easily aggregate regions together into larger regions, which will allow a reduced set of routes, or more accurately coverage areas, to be advertised between networks.

The main requirements for our system are: 1) Accuracy, 2) Minimal delay, and 3) Scalability. As we have shown in our evaluation our addressing system has an accuracy of 30% (percentage of chosen area in the addressed area). We can match addresses based on simple bitwise AND comparisons which in turn allows minimal lookup delay. Due to the hierarchical nature of our addressing system it is also very scalable, allowing networks to advertise their coverage in a single address and in turn decreasing the size of lookup tables in routers.

Now that we have an addressing system we can move forward to actually routing packets. In the following chapters we will work towards a routing system based on the addressing system presented in this chapter.

---

## Geographic Forwarding Trees

*Before we can use our addressing mechanism in a routing protocol we need to find the most suitable forwarding tree for geocast. In this chapter we perform a comprehensive evaluation of different forwarding tree approaches. We evaluate these trees on their efficiency for routing geocast packets. We evaluate the Shortest Path Tree, Minimum Spanning Tree, and a Steiner-heuristic-based forwarding tree for geocast packet distribution on real world networks and random graphs. We compare the results to those for multicast routing for which such evaluations have been performed in the past. Our results show that due to the correlation of geographic distance and network distance in most wired networks, Shortest Path forwarding efficiency can come close to an ideal Steiner Tree. We will be able to use these results in the coming chapters to design a geographic routing system. The contents of this chapter are based on the work presented in "Evaluation of geocast routing trees on random and actual networks" [49].*

1. Introduction	
2. Background & Related Work	
3. Geographic Addressing	
4 Geographic Forwarding Trees	
5. Geographic Routing Algorithm Design	6. Geographic Routing Implementation and Evaluation
7. Infrastructure Assisted Contention-Based Forwarding for Geocast	
8. Conclusions and Future Work	

## 4.1 Introduction

In the previous chapter we have defined an addressing system that can be used for geocast. An addressing system alone does not allow us to route packet to a geographic area, the next challenge is that of geographic routing. To make a geographic routing algorithm we first have to decide which type of forwarding tree would best help us meet our requirements. In this chapter we explore several types of forwarding trees that could be applicable to geocast to find the one that is best.

The routing method most used, and in fact the only method supported globally, in the Internet today is unicast. Unicast routing usually uses the shortest path, based on some cost metric (like least hops), between the source and destination hosts. While this is a logical approach for a unicast system, as there always exists an obvious optimal path between any source and a single destination, it does not necessarily make sense for a system which has multiple destinations. With geocast, the destination likely consists of several devices that might be served by multiple routers. Each of these routers will need to receive a packet from the source, leading to a situation where a single path is not enough but we need some form of tree.

As mentioned in the previous chapters, geocast can be compared to multicast in several ways. Both geocast and multicast transmit packets to multiple destinations. They also share forwarding characteristics in that packets are only duplicated when the path in the network diverges. Unlike multicast the destination of packets in geocast share a geographical region and they are not distributed throughout the network. Furthermore, unlike multicast a device cannot simply subscribe to a group to receive a geocast packet, it has to be present in a certain area. The geocast packet is transmitted to all devices on a network in a specific geographic region. These characteristics are especially beneficial for transmission towards vehicular networks, where nodes are mobile and keeping track of membership information and location is inefficient [1].

Due to the geocast specific requirements, forwarding requirements for geocast differ from multicast in several ways:

1. There is a logical and fixed correlation between the geocast address we defined in Chapter 3 and the area a packet needs to be forwarded to. A address describes exactly one region, and similar addresses represent similar regions.
2. Forwarding is based on a geocast address, not membership information. Routers will need to know which link(s) lead to an area, instead of keeping track of membership information.

We can already fulfil part of the first requirement through the addressing system presented in Chapter 3. For our routing system the consequence is

that destination routers will be geographically close to each other, but that does not guarantee they are close to each other in the network. Devices in the same area might connect to different networks (such as different ISPs or mobile networks), each of these devices should get the same geocast packets. The second requirement will depend on the routing algorithm used for which we need to find a suitable forwarding tree. The consequence for a routing system is that no per packet or per destination signalling is needed.

In a multicast scenario the routers that need to be reached can be distributed throughout a network. In the geocast case, these routers would be located close to each other geographically. While geographic distance does not directly correlate to network distance, a strong link between both of them can be observed in a large number of real world networks. Our hypothesis is that this geographic clustering will lead to a situation where a geocast source has an obvious forwarding path to the destination routers. This could result in a significant portion of shortest path routes from the source to the destination being shared. Using unicast will lead to redundant packets travelling over the same link, while using multicast would add signalling overhead. A new set of routing algorithms specifically designed for geocast is needed to provide an effective geocast solution in Internet-scale networks [1].

In this chapter we answer our second research question: *Which forwarding tree is the most efficient for geographically scoped destinations?* The answer to this question will help us to design an efficient geocast routing algorithm. Our hypothesis is that more optimal methods like Steiner trees are not as relevant when destinations are located close to each other and simpler but computationally less expensive methods such as naive shortest path forwarding are more attractive. Our assumption is that routers that are responsible for areas in close geographical proximity, are also close to each other in the network with a small number of hops between them.

The main contribution of this chapter is to select the most suitable forwarding tree to use in the design of an efficient geographic routing algorithm. We do this by performing an extensive evaluation of different forwarding trees in different geocast scenarios. We use the average cost and path utilization over multiple (source, destination) pairs as our main metrics. We compare the results with results from multicast based evaluations. The multicast case has been extensively researched in the past [50, 51], but the effect of geographical clustering on the forwarding tree efficiency is an open question. This information will be used in Chapter 5 to design an efficient routing system for geocast traffic, which can be used in combination with the addressing system we presented in Chapter 3.

This chapter is organized in the following way: In Section 4.2 we explore previous work on the topic of multicast shortest paths and random graphs. Section 4.3 explains our evaluation approach and which metrics we use. The

results of our evaluation are described and discussed in Section 4.4. Finally we summarize our results and draw conclusions in Section 4.5.

## 4.2 Related Work

Previous papers have explored the benefit of using different forwarding mechanisms for multicast traffic. The authors of [50] show that a naive shortest path tree from the source is not that much more inefficient than a Steiner tree heuristic method. Their evaluation focuses on multicast performance in Waxman graphs. This result might mean that the same could hold for geographically scoped destination.

In [51] the authors evaluate different multicast trees for their properties in overall cost and delay. Like [50], they show that a shortest path tree based approach can come close to the Steiner tree heuristics in terms of performance.

The authors of [52] compare multicast path cost (in hops) to the unicast cost for the same destinations on a set of different networks. They note that on average multicast distribution costs 0.8 times the number of hops compared to unicast routing to the same destinations for an unsaturated (not all routers have multicast subscribers) network. The authors also note that this correlation holds over different types and sizes of networks, but due to the comparison to the unicast tree in the same networks these results lose network specific information.

More recently the focus of this kind of evaluation has been in the realm of ad-hoc wireless networks. In [53] Nguyen et al. show that shortest path trees provide benefits over minimum cost trees in wireless ad-hoc networks. According to the authors these benefits outweigh the downside of higher tree cost.

Knight et al. have published a database of public network topologies at the PoP level [39]. They perform a statistical analysis on the data and map the properties such as node degree of these network. We use the real world networks published in this Topology Zoo in our evaluation and we will use the statistical data to generate random geometric graphs.

Constructing a Steiner tree over a graph is an NP-complete problem. Kou et al. have presented a fast Steiner heuristic algorithm [54]. We use this algorithm to find the Steiner tree for our route evaluations. This allows our evaluation to contain a larger number of graphs than would otherwise be possible. It also has the benefit of being more close to a solution that could realistically be used in an actual router for tree construction.

## 4.3 Evaluation Approach

In this section, we will explain our approach to evaluate three different types of routing trees. We will first present the trees with their advantages and drawbacks, followed by a short presentation of the tools and sources used. To perform a fair evaluation of the three different routing tree approaches in a geocast scenario we will use two graph models. We will generate random geometric graphs to create a set of networks on which we can perform evaluations, and we will use actual network topologies used in the real world. Information relevant to these graphs and assumptions we make about them will also be presented in this section. We will follow this by our destination set selection method. At the end of this section we will explain our evaluation metric and give some examples based on a simple graph.

### 4.3.1 Routing Trees

We evaluate three methods of geocast and multicast trees that can be realistically used for routing:

1. shortest path tree from source,
2. minimum spanning tree,
3. Steiner tree from source.

Each of these approaches have different benefits and drawbacks that will make them more or less suitable for geographic routing depending on the goals of the network administrator or even the layout of the network. We will explain all three methods using a real world network shown in Figure 4.1. In this figure each node represents a router positioned at their geographic location relative to the other routers.

#### Shortest Path Tree

The shortest path tree is simply a combination of all shortest paths from the source to all the destination nodes. We count a link that is used multiple times as one usage. We assume that with link overlap an underlying routing protocol can prevent duplicate packets over the same link. For example: Router  $A$  needs to forward a message to a specific area which includes router  $B$ ,  $C$  and  $D$ . The shortest path tree would be the union between the shortest paths  $(A \rightarrow B)$ ,  $(A \rightarrow C)$  and  $(A \rightarrow D)$ .

Consider an example situation using Figure 4.1: Using node 6 as the source and nodes 8, 9 and 10 as destinations the shortest path tree would consist of  $6 \rightarrow 7 \rightarrow 8$ ,  $6 \rightarrow 10 \rightarrow 9$  with a total cost of 4. This approach requires a

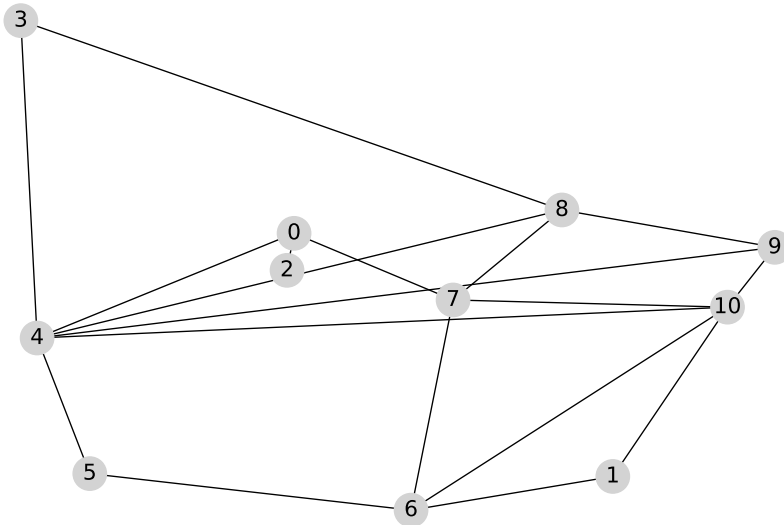


Figure 4.1: A real world network graph

per (source,destination) pair forwarding calculation for each router. A simple per destination forwarding calculation, as would be the case for unicast, is not possible. Such a simple forwarding mechanism would lead to routers on the path to a destination to forward the packet on the shortest paths to each destination, which is likely not the desired effect. As it is probable that the destination area includes multiple routers, a forwarding router needs at least some knowledge of how it fits in the distribution tree to make an efficient forwarding decision. We suspect that this approach will be efficient for geocast as the geographic closeness of destination likely correlates to closeness in the network to some extent, leading to a large number of shared links.

### Minimum Spanning Tree

For the minimum spanning tree (mst), we simply calculate the minimum spanning tree of the network (based on hop count). This sub-graph is used to reach all destination nodes from the source. This approach has the benefit that the distribution tree for any geocast (or other forwarding type such as multicast) message can be precomputed. The major downside is that a number of links will carry all the traffic, while others are never used. This approach will also not lead to the lowest overall path cost as the most efficient route will almost never be used in most networks. It can however, perform equal to



the Steiner tree in situations where the source and destination nodes are ideally distributed on the minimum spanning tree. An example of such a situation is a network where all nodes are on a line, only a single spanning tree can exist. However, this situation is not likely to occur often and will be offset by all the destinations that are not ideally distributed on the minimum spanning tree in most network types.

### Steiner tree

A Steiner tree is the least cost tree between source and destination nodes. Because this is a NP-complete problem we use a well known heuristic algorithm [54] to construct it. This algorithm works by first finding the metric closure of the nodes we are interested in. The minimum spanning tree is calculated over the metric closure graph and we map this back to the actual network. This approach will lead to a close to optimal graph, but like the shortest path approach we need to compute a tree for each (source, destination) pair, with higher computational overhead. Using Figure 4.1 as an example again, with node 6 as the source and nodes 8, 9 and 10 as destinations: The Steiner tree would consist of  $6 \rightarrow 10 \rightarrow 9 \rightarrow 8$  with a total cost of 3 (one less compared to the shortest path tree). As mentioned before, the Steiner tree is the least cost tree but has the downside of requiring more overhead to compute compared to the other two trees. In the geocast scenario a forwarding router would need knowledge of the source router and all destination routers to know its place in the ideal forwarding tree.

### Computational Complexity

The computational complexity of these three approaches differ widely. We will do a short comparison on the complexity of all three options when used to make a forwarding decision. We will discuss each tree option in the order of lookup complexity.

A minimum spanning tree can be computed once for any given network (given there are no topology changes) and used for every forwarding decision from then on, but does require full network knowledge at each router to be computed. Once a router has computed the minimum spanning tree it knows the next hop for every other router in the network. This can be trivially used for geocast distribution as there are no loops on the tree by definition. As a result the lookup complexity is extremely low, at the cost of a initial tree computation which has to be repeated any time the network topology changes.

A shortest path tree as we describe it requires full network knowledge like the minimum spanning tree. Without full network knowledge routers can not know their position on the tree and in turn limit the destinations they forward the packets to. As the network is not already limited to a tree as with the

minimum spanning tree the forwarding next hops depend on both the source and destination. This requires each router to calculate the shortest path from the source to all destinations and forward only on branches of the tree it is part of. This calculation is significantly more complex than the minimum spanning tree.

A Steiner tree has similar limitations as the shortest path tree, including the requirement for full network knowledge. The computation of a Steiner tree is however significantly more complex [54], as each router receiving a packet needs to compute the Steiner tree for each (source, destination) pair and only forward if it is on one of the branches of the tree. This method has the highest forwarding complexity by far.

### 4.3.2 Tools and Sources

To perform our evaluation, we used several pre-existing tools. We use two different sources for networks: real world and randomly generated networks. All the real world graphs we evaluated are taken from the Topology Zoo [39]. The randomly generated graphs used are generated using the NetworkX package [40] for the Python programming language. We also use this package to import the Topology Zoo graphs and to help us with performing the actual evaluation. A more in-depth description of these tools can be found in Chapter 2.

### 4.3.3 Networks

For the rest of this chapter, we will refer to a network as a graph  $G = (V, E)$ , with  $V$  the vertices or nodes (representing routers),  $E$  the edges (representing links between the routers). Routers (represented by the vertices) are placed at a geographic location. We assume the networks we evaluate are static, there are no topology changes during the evaluation. We also assume that all links in the network have an identical cost of 1. We use both real world networks and randomly generated graphs in our evaluation, which we will now further explain.

#### Real Networks

To perform a fair evaluation of the different approaches we need to consider real world networks, both as a control sample and as a validation of the random geometric graphs. A computer network is by definition a designed system that is built in a certain way for specific reasons such as cost, performance or necessity. This also means that nodes close to each other are not always directly connected due to various reasons.

To evaluate against real networks we use several network graphs that have been made available through the Topology Zoo project [39]. We import these

graphs and remove all nodes that are not connected to other nodes. When the resulting graph is still disconnected, we take the largest sub-graph as the graph to run our evaluation on. After these operations we are left with a total of 226 graphs. In the majority of cases the graphs can be imported without these operations. We place nodes at the geographic coordinates they are labelled with in the Topology Zoo. One example of such a graph is the one depicted in Figure 4.1. This graph will be used later to explain our evaluation process.

### Random Geometric Graphs

To supplement the actual networks used and provide a basis for more general conclusions we have also generated a set of random geometric graphs to run our evaluation on. We chose to use random geometric graphs because the presence of edges between vertices is based on geometric distance. This property is helpful in geocast evaluation as it provides a strong correlation between network distance and the relative distance between nodes. We acknowledge that a random geometric graph may not represent an actual network with high accuracy, but the set of actual networks should sufficiently cover this, allowing the random networks to focus on an ideal geocast case.

We generate our graphs using the *random\_geometric\_graph* function of NetworkX. We generate the positions of nodes using a normal distribution with a standard deviation of 2 for both the x and y coordinates. Vertices are connected by an edge if their distance is less than 1.8. Generated graphs are accepted as valid based on three criteria:

1. The graph is connected (every node is reachable by every other node),
2. The average betweenness centrality is similar to the studied real networks with values between 0 and 0.5. The betweenness centrality is a measure of the importance of a node, it is the fraction of shortest paths between node pairs that pass through it [55]. The average gives an indication of how centralized a network is.
3. The average node degree is distributed with values between 2.5 and 3.5, such that they resemble the real network. The node degree represents the number of links a node has. The average node degree we use is the average of the node degree of all nodes in a network ( $2|E|/|G|$ ).

Graphs that did not meet these requirements were discarded.

For the majority of random graphs, we choose to generate them in such a way that they closely resemble values from the real world networks. As noted above, these values are comparable to real networks found in the Topology Zoo [39]. We have also generated some special graphs, such as fully connected graphs and graphs that resemble a star topology to evaluate those specific scenarios.

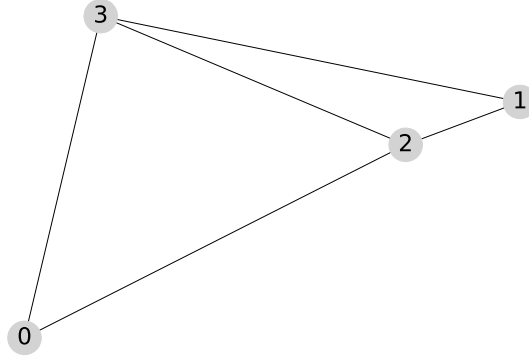


Figure 4.2: 4 node example graph

### 4.3.4 Destination Selection Method

To evaluate multicast and geocast destinations in the graphs we use different node selection methods. Both methods share the source node selection. Every node in the network is selected exactly once as the source for every possible destination set, the set of source nodes is equal to the set of nodes  $V$ . Runs are done for all destination sets containing 1 node, 2 nodes, up until the total number of nodes in the network, excluding the source. The destination set generation method differs between the multicast and geocast case.

#### Multicast Destination Selection

In the case of multicast, the destinations are every possible combination of all other nodes in the network. We define this set as  $DS_{|V|-s}^s$ .

$$DS_{|V|-s}^s = \{\{d_1\}, \dots, \{d_1, \dots, d_{|V|-1}\}\} | d \in (V - s) \quad (4.1)$$

The destination set  $DS_{|V|-s}^s$  is the set of all possible combinations of set  $V$  without the source node  $s$ :  $(V - s)$ . The maximum length of a destination set is  $|V| - 1$ , all nodes except the source node. If we take the example network given in Figure 4.2, using node 0 as the source, the destination set would be  $\{1,2,3, (1,2), (1,3), (2,3), (1,2,3)\}$ .

#### Geocast Destination Selection

The geocast evaluation selects each (non source) node as destination once and selects extra nodes that are geographically closest depending on the number of destinations required. For each of these destination nodes, 0 to  $N - 1$

extra nodes are selected. The extra nodes are always selected based on their geographical distance to the initially selected destination, the first node added is always the closest, the second node is the second closest and so on. Destination sets are distinct, generated sets that are identical to already existing sets are ignored as they would represent the same geocast area. We define this set of geographically scoped destinations as  $GS_{|V-1|}^s$ .

$$GS_{|V-1|}^{s,d} = \{d, v_1^d, v_2^d, \dots, v_{|V-1|}^d\} | d, v \in (V - s) \quad (4.2)$$

$$GS_{|V-1|}^s = \bigcup_{d \in (V-s)} \{GS_{|V-1|}^{s,d}\} \quad (4.3)$$

In these equations  $GS_{|V-1|}^{s,d}$  represents the geographic destination set with  $d$  as the initial destination and  $s$  the source,  $v_n^d$  are the other nodes in the network sorted by their geometric distance from  $d$ .  $GS_{|V-1|}^s$  is the set of all distinct destination sets for source node  $s$ . In the example network shown in Figure 4.2 this would be  $\{1,2,3,(1,2),(3,2),(1,2,3)\}$  for source node 0. Note again that we do not use the same destination set twice here. In this case node 1 is also the closest other node to 2, we do not include (2,1) as this will replicate (1,2).

### 4.3.5 Evaluation Metrics

To perform our evaluation we use the same evaluation metrics for both network sets, and the three routing trees. We evaluate the performance of the three different routing trees using the following metrics:

1. Path cost: the total cost of reaching all destinations,
2. Edge usage: the total number of times a specific edge (or link) is used in graph over multiple evaluations.

These metrics are important because they represent how efficient a forwarding tree is. In general, a lower path cost is considered better, due to a lower amount of transmissions. A more equal edge usage distribution can be beneficial as traffic is more evenly distributed throughout the network, reducing the chance of bottleneck links.

We also vary the link costs in the networks we evaluate to include situations where all link costs are equal to a situation where there is a variation in link costs. From the literature we know that in the case of multicast traffic, links are likely to have the identical costs associated with them [52].

To present the way we will interpret our graphs we will use the network in Figure 4.1 as an example. This network has 11 nodes and 18 links.

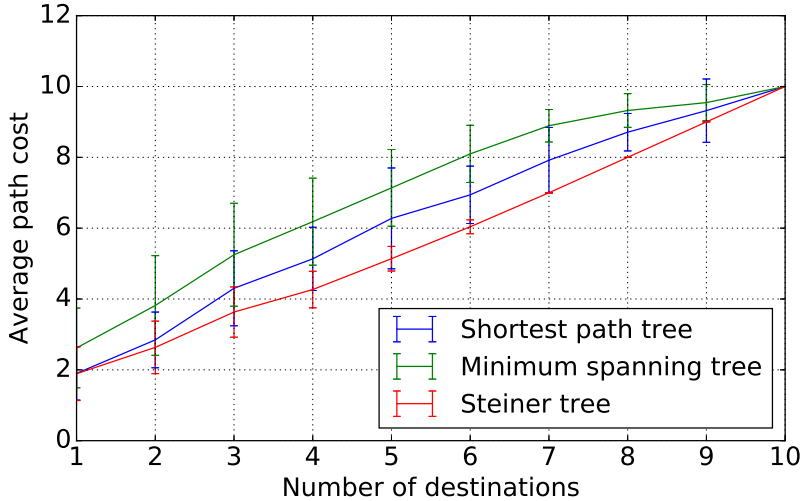


Figure 4.3: Path cost of network graph example

The graphs used to present our results are generated using a consistent colour coding scheme. *Blue* data belongs to the shortest path tree, *red* data belongs to the Steiner heuristic and *green* data represents the minimum spanning tree.

### Path Cost

We define path cost as the sum of the cost of all links used in a tree. As we assume all our links have a cost of 1, this is simply the amount of links used on the forwarding tree to reach all destinations.

To get an accurate representation of the amount of links used by a forwarding tree we use the average path cost over a network. To calculate this cost we use all possible destination combinations for multicast and every possible geographically-clustered destinations for geocast, as explained in Section 4.3.4.

As an example with destination size 1: There are 11 nodes in the network shown in Figure 4.1. These 11 nodes each have 10 destinations giving us 110 (source, destination) sets. We take the average cost of these 110 routing trees for each of the three routing tree approaches.

We will start presenting our results as graphs that show the average cost for a number of destination per routing tree type. In Figure 4.3 the results for network in Figure 4.1 are shown. The error bars represent the standard deviation. For this specific network we can see that the shortest path tree cost is close to that of the Steiner heuristic when the destination set is small. We

can also observe that when the destination set includes all nodes the routing costs of all trees converge.

Later in the chapter we show an average normalized path cost per graph. This cost has been normalized by the number of edges in a graph to allow comparison between graphs of different sizes.

### Edge Usage and Fairness

To determine how ‘fair’ the link utilization is, we evaluate it for different networks. The link utilization metric describes how evenly the load is distributed in the network. If a few links are used for almost every combination of source and destination nodes it could get overloaded. Overloading a few links and leaving others completely unused is not likely to be a desirable property, and should be something to take into account.

We define the edge usage of an edge as the fraction of unique (source, destination) combinations where it was used as part of the tree. For example, if we do 10 runs with different (source, destination) pairs and a certain edge was used in 6 of those runs, its edge usage would be 0.6.

We believe the fairness of edge usage to be an important factor as it describes the load distribution within the network. With fairness we refer to how well the edge usage is distributed in the network, maximum fairness would imply that all edges have equal edge usage. A situation where few links carry almost all traffic might not be desirable from a cost and load distribution standpoint.

Using Figure 4.4 we will explain how our stacked bar charts for edge usage are constructed. Figures 4.4a, 4.4b, and 4.4c show the edge usage fraction per edge for the network in Figure 4.1. Each of the bars represents an edge, with the height representing the fraction of runs this edge was included in the tree. These edges were sorted with decreasing edge usage for viewing convenience. We can see that the shortest path and Steiner heuristic trees use all edges and the minimum spanning tree only uses 10 out of a total of 18. Figure 4.4d combines these graphs into a single stacked bar chart per routing tree. We can clearly see that the usage is more evenly distributed in the shortest path and Steiner heuristic methods and that for the minimum spanning tree a considerable fraction of edges is never used and another significant fraction is almost always in use.

## 4.4 Evaluation Results

In this section we will present the results over all the graphs we have evaluated. We start with the general results, where we will show the path cost and edge usage of all three trees. Following the general results we will discuss the effect

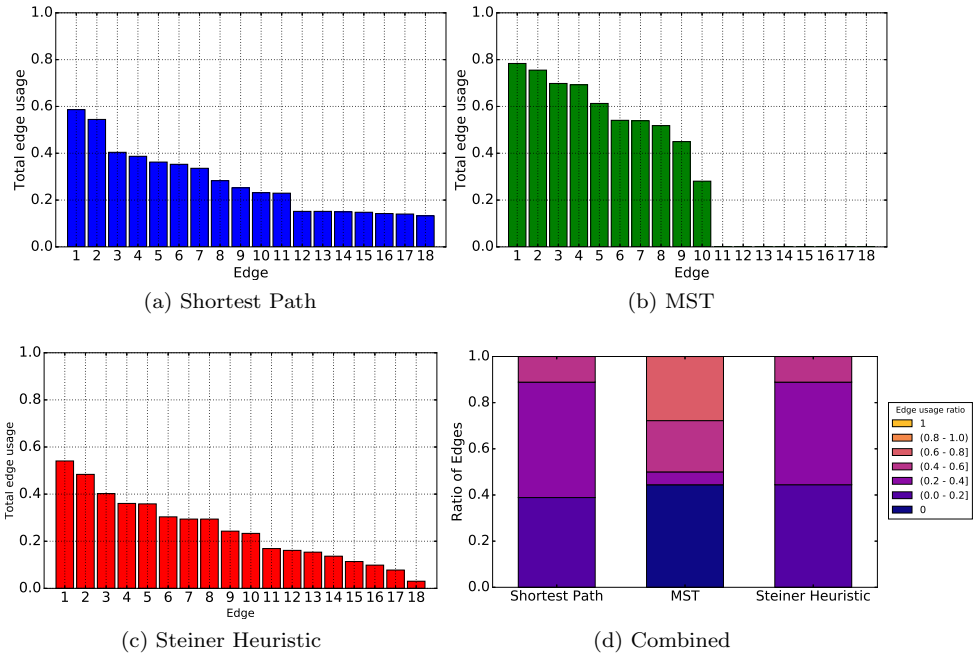


Figure 4.4: Edge usage of the network in Figure 4.1

of certain network characteristics on the efficiency of the different forwarding trees. Finally, we will show results for several special network types.

#### 4.4.1 General Results

##### Average Path Cost

We start by looking at the average path cost for our evaluation set. Figure 4.5 shows the average path cost for the 85 networks taken from the Topology Zoo [39] that have fewer than 20 nodes with all edges having weight 1. It was not feasible to compute the multicast performance for the larger networks due to the large number of destination combinations. The graphs show the number of destinations on the x-axis (starting with 1) and the average cost on the y-axis. Each line in the graph represents one network. As shown in Figure 4.5, on average, almost all networks we evaluated show similar results. There are a few outliers visible in the results that we will discuss later.

In Figures 4.5a and 4.5b we show the average cost of routing a packet in a multicast and geocast situation respectively, using shortest path forwarding on



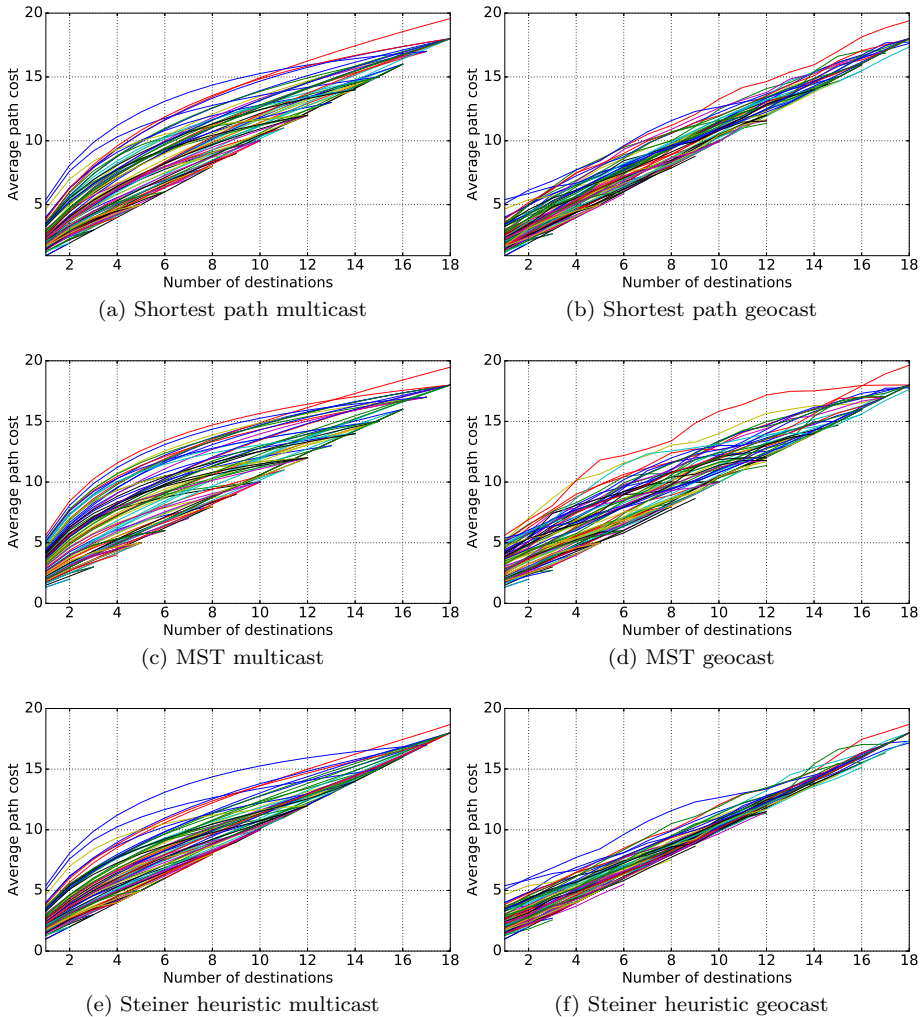


Figure 4.5: Results over 85 real networks smaller than 20 nodes

networks where all edges have cost 1. The case for using a minimum spanning tree and a Steiner heuristic can be seen in Figures 4.5c, 4.5d and Figures 4.5e, 4.5f respectively.

In general, we can see that the geocast scenario is more efficient in terms of forwarding cost than multicast in situations where the number of destinations is around a third of the total number of nodes in the network. We can also

observe that the Steiner heuristic is the most efficient forwarding method as expected. The shortest path method is however not that much less efficient while using significantly less computational resources. We can see that the minimum spanning tree approach shows less than optimal results but is not necessarily much less efficient depending on the network. It also has the benefit of being precomputed so forwarding costs would be extremely low.

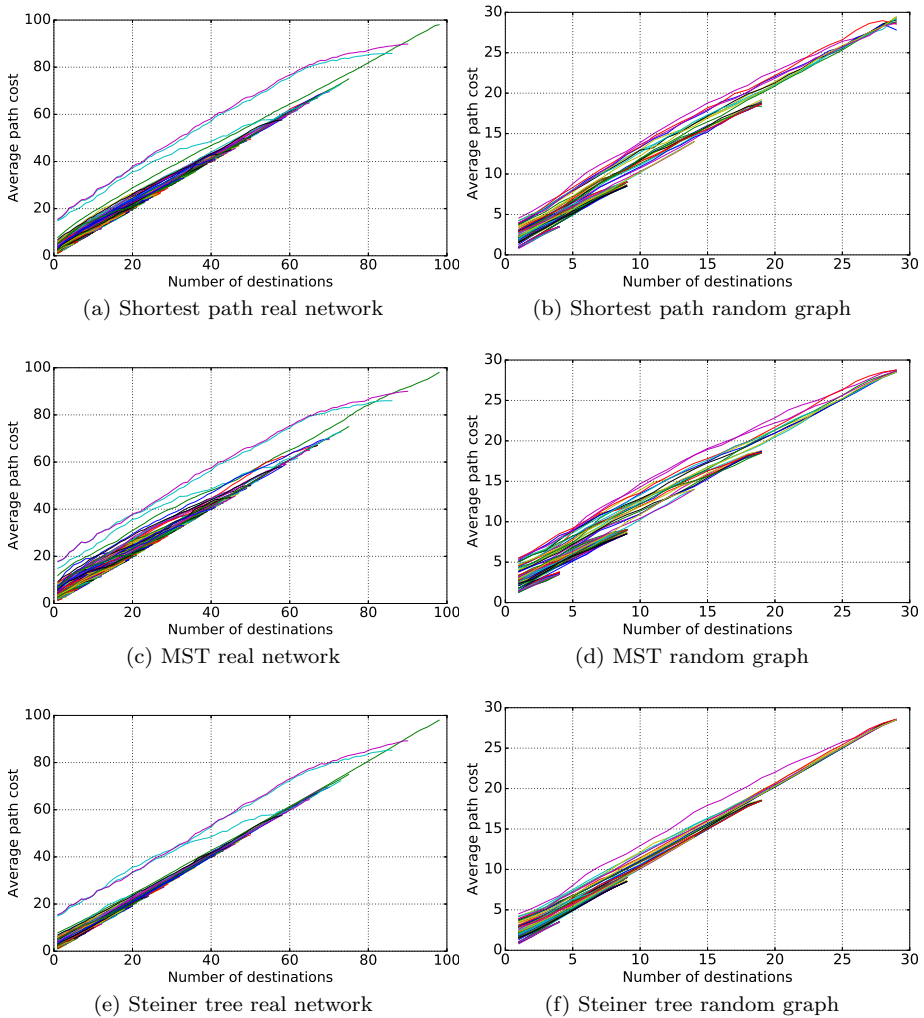


Figure 4.6: Geocast results over 225 real networks and 98 random graphs

We evaluated geocast results for all networks in the Topology Zoo [39]. These graphs are an extension of the graphs in Figure 4.5, also including the networks with more than 20 nodes found in the Topology Zoo. These results can be seen in Figures 4.6a, 4.6c and 4.6e for the shortest path tree, minimum spanning tree and Steiner tree respectively. In these graphs we can see the linear relation between the number of destination nodes and the average cost more clearly. The average cost (or links used) is almost identical to the number of destinations for the shortest path (Figure 4.6a) and Steiner heuristic (Figure 4.6e). The minimum spanning tree has a higher cost when compared to the others. There are three obvious outliers for the shortest path and Steiner tree. These are networks that consist of several rings, each consisting of a large amounts of nodes. This leads to high overall cost to reach these destinations unless a significant portion of the network is used as destination, as can be seen by their eventual return to the linear relation between cost and destinations.

Figures 4.6b, 4.6d and 4.6f show the geocast cost for the random geometric graphs we evaluated. These results are comparable to the results for the actual networks. We only observe a small difference in the lower maximum costs found, likely caused by the stronger correlation between geographic distance and network distance in the random geometric graphs.

### Edge Usage and Fairness

In an ideal environment we would like to distribute the distribution tree in the network in such a way that every edge is used equally. This is under the assumption that (source, destination) pairs are also evenly distributed throughout the network.

In Figure 4.7 we show the fraction of runs that a certain fraction of edges has been used. Each graph shows the results for shortest path, minimum spanning tree, and Steiner heuristic. In Figure 4.7a and 4.7b we compare the results for

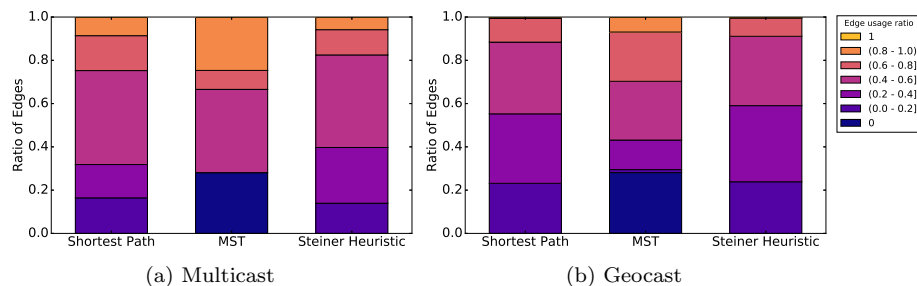


Figure 4.7: Edge usage

all multicast runs with geocast runs over the same set of networks with less than 20 nodes. We can clearly see the difference between the multicast and geocast scenario. With multicast there is a number of edges that are almost always used, while this effect is diminished when destinations are geographically clustered. We observed the same results for geocast on the full set of real networks.

In general we can conclude that the fairness of the minimum spanning tree approach is the lowest as a significant number of edges is never used. Of course this result was to be expected as the same tree is used for every (source, destination) set.

Based on our general observations on multiple different networks, we observe that the more connected a network is the more equal the load is distributed. This makes sense as there are more possible paths in the network to reach all destination. On average multicast forwarding seems to use more edges compared to geocast. This result can be explained by the geographic clustering of the destinations, making the path from source to destinations share more edges.

#### 4.4.2 Correlation with Network Characteristics

Some network characteristics have influence on the performance of forwarding trees. In other words, the way some networks are designed lead to a certain forwarding performance and give them specific values for these characteristics. The characteristics of particular interest are the average node degree of the network and the betweenness. We calculate the average normalized path cost per graph for the following results. The path costs are normalized by dividing them by the number of edges present in the graph.

Node degree is the number of edges a node has. In the case of a fully connected network this is equal to  $N_{deg} = |G| - 1$ . The minimum node degree is 1, as can be found in a node that is only connected to a single other node (for example in a star topology). The average node degree of a network is simply the average of all node degrees in that network. This average gives an indication of how well connected a network is, in general networks with a higher average node degree have shorter paths between two nodes.

Figure 4.8 shows the normalized average path cost of a network for the different routing trees plotted against the average node degree of the network. Every dot in these graphs represents a the average value for a network given a certain routing tree, with the line being fitted to these results of the dots for the same forwarding tree. As expected we see a strong correlation between the two values. We can conclude that the different routing trees show similar performance when the node degree is below 2. This makes sense as there are only a few possible paths to choose between with such low node degrees. When

the node degree is higher, more efficient forwarding trees (that establish shorter paths) are more beneficial to use as there is more choice. Note that our fitted lines do converge for node degrees close to 5, but this is due to the lack of networks with such a high average node degree.

The betweenness centrality of a node is the fraction of shortest paths the node is on in the network. Figure 4.9 shows the results for this metric in a similar manner to the node degree graphs. We see that when the average betweenness centrality of a network is high, the average normalized path cost is also higher. The correlation for multicast seems stronger than that for geocast with this metric.

### 4.4.3 Special Networks

As mentioned before, the general topology of a network has a large effect on how efficient geocasting, or any other forwarding method, is in the network. A few of the real world networks showed interesting results due to their topology. The shape of these networks might affect the choice of routing method that should

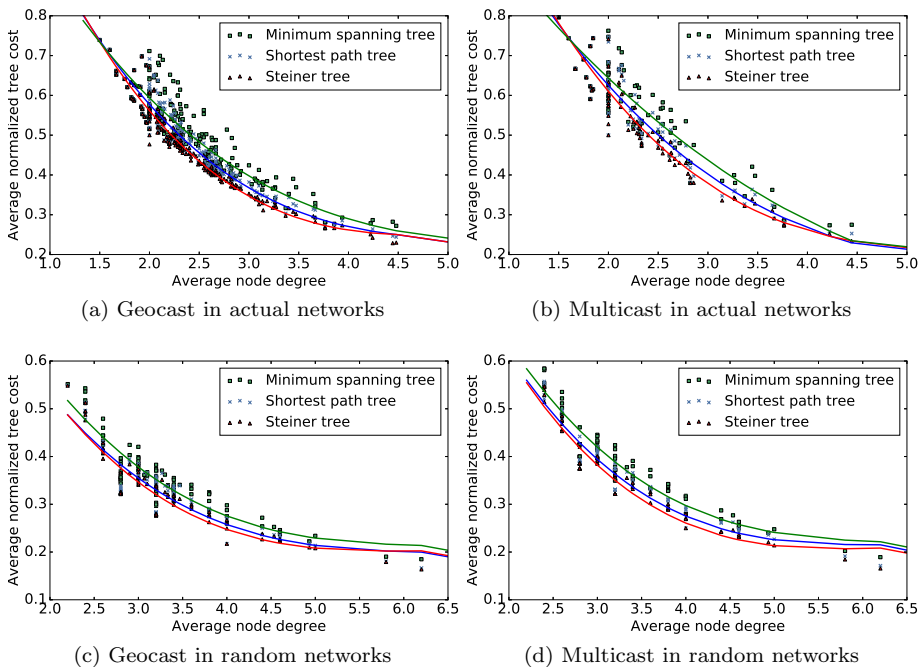


Figure 4.8: Node degree against cost

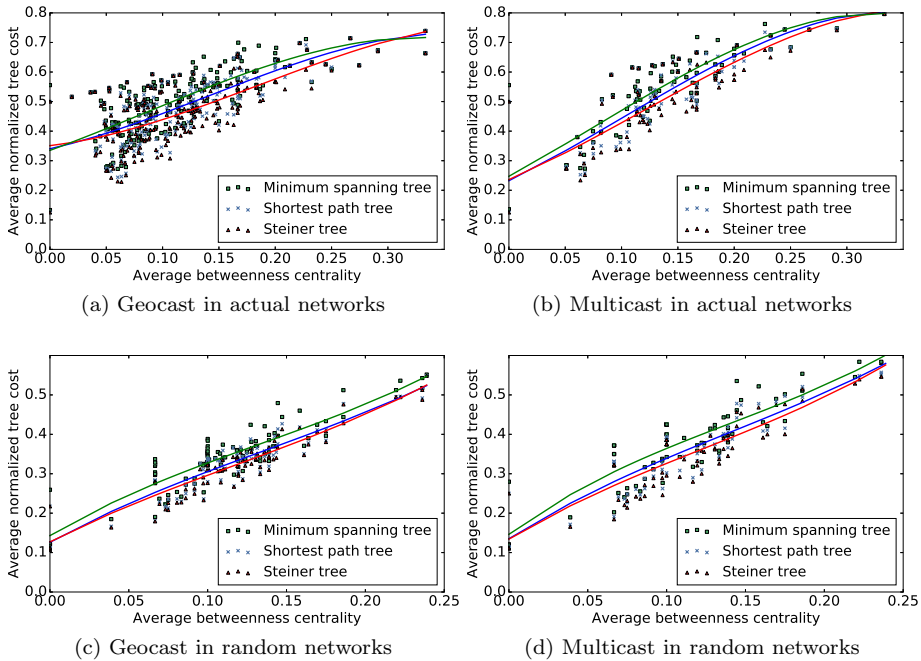


Figure 4.9: Average betweenness against cost

be used in those networks. In this section we evaluate some ideal networks of the following types: line, ring, star and fully connected networks. We evaluate these networks with geographically clustered destinations as described in Section 4.3.4.

We show the results of some these networks in Table 4.1. In this table we present the average link cost as fraction of the Steiner tree cost, as this is the lowest link usage possible. We have generated networks off the types ‘Line’, ‘Ring’, ‘Star’ and ‘Fully connected’ with 5, 10 and 20 nodes. In Table 4.1 these are shown (with ‘Fully connected’ shortened to ‘Full’) followed by the number of nodes.

### ‘Line’ Networks

These networks simply look like strings with routers on them (Figure 4.10a), every router is connected to two other routers with the exception of the routers on the edge of the line. Due to every router only having one link towards the geocast region in most cases, the shortest path approach is very efficient here.

In a ‘true’ line network the minimum spanning tree is identical to the network and performs the same as shortest path and the Steiner heuristic as can be seen in Table 4.1.

A line network’s node degree is given by the function  $D_{line}(N)$ , where  $N$  is the number of nodes. The betweenness centrality  $B_{line}$  is independent of the number of nodes, this is a fixed property of a line network.

$$D_{line}(N) = \frac{2(N - 2) + 2}{N} \quad (4.4)$$

$$B_{line} = \frac{1}{3} \quad (4.5)$$

### Ring Networks

In networks that are designed as a ring, every node is connected to two other nodes as shown in Figure 4.10b, the shortest path method is less efficient. This is likely caused by using both sides of the ring to reach a geocast area if the source is located on the opposite side of the destination in the ring. The Steiner heuristic always produces an optimal tree in such a network. The minimum spanning tree can be extremely suboptimal depending on the location of the source and destination nodes, in the worst case making what should be a one link path use the entire rest of the ring to deliver a packet.

The average node degree of a ring network  $D_{ring}$  is logically always 2, as each node is always connected to two other nodes. The average betweenness  $B_{ring}$  can be described by Eq. 4.7. This equation approximates the average betweenness of a ring network.

$$D_{ring} = 2 \quad (4.6)$$

$$B_{ring} \approx \frac{\frac{N-2}{4}}{N-1} \quad (4.7)$$

### Star Networks

These networks, also called hub and spoke networks, generally have one or more hubs that have the majority of other routers connected to them in a star pattern. A single star network can be seen in Figure 4.10c. The effect of this type network of layout on forwarding is that all traffic has to flow through a central node. When multiple star networks are connected this leads to a few heavily used links between the hubs. If we consider a network that has only one hub we see that there is no difference in the performance between multicast and geocast routing. This makes sense as all routers (excluding the hub router)

are two hops away from every other router (again excluding the hub). There is no possibility to optimize the distribution tree in this situation, every tree performs identical as seen in Table 4.1.

A star network has an average node degree described by  $D_{star}$ . This logically correlates with the fact that all nodes, except the centre node, are only connected to that centre node, thus having node degree 1. The center node has node degree  $N - 1$  as it is connected to all other nodes. The average betweenness centrality  $B_{star}$  decreases with the number of nodes, as the centre node is always on the shortest path between all other nodes. The centre node thus has a betweenness centrality of 1, and all other have a value of 0.

$$D_{star} = \frac{2(N - 1)}{N} \quad (4.8)$$

$$B_{star} = \frac{1}{N} \quad (4.9)$$

### Fully Connected Networks

An unlikely network to occur in reality, but an interesting theoretical situation to evaluate is the fully connected network. Here every router has a direct link to every other router (Figure 4.10d). The result is a network in which every node can reach every other node in one hop. The shortest path and Steiner tree are always optimal (and identical) in this situation, as both simply use the direct connection between all nodes to reach the destinations. The minimum spanning tree will lead to two hops between most node pairs as it creates a star network. This result logically corresponds to the node degree graph, the higher the node degree (equal to  $|N| - 1$  in this case) the lower the average cost.

The average node degree  $D_{fully\_connected}$  for a fully connected network is logically one less than the number of nodes in the network, as all nodes are connected to all others. The average betweenness centrality  $B_{fully\_connected}$  is 0 as no node is on a shortest path between two other nodes, they are all directly connected.

$$D_{fully\_connected} = N - 1 \quad (4.10)$$

$$B_{fully\_connected} = 0 \quad (4.11)$$

## 4.5 Summary & Conclusion

In this chapter we analysed the efficiency and fairness of a shortest path tree, Steiner tree and minimum spanning tree for geocast and multicast forwarding. Our goal was to find the most efficient forwarding tree for geocast traffic.



Graph	SPT	ST	MST
Line 5	1.0	1.0	1.0
Line 10	1.0	1.0	1.0
Line 20	1.0	1.0	1.0
Ring 5	1.081	1.0	1.205
Ring 10	1.155	1.0	1.253
Ring 20	1.191	1.0	1.291
Star 5	1.0	1.0	1.0
Star 10	1.0	1.0	1.0
Star 20	1.0	1.0	1.0
Full 5	1.0	1.0	1.215
Full 10	1.0	1.0	1.170
Full 20	1.0	1.0	1.132

Table 4.1: Relative tree cost for special networks

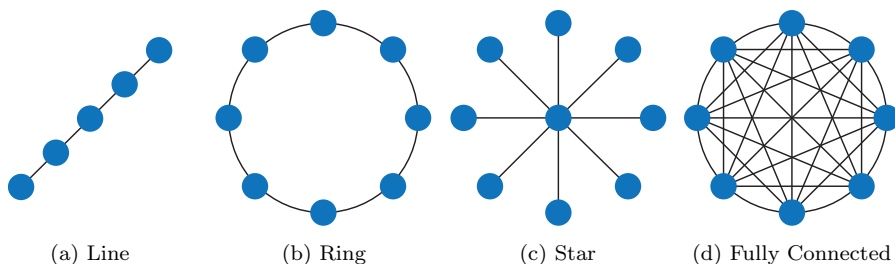


Figure 4.10: Special networks

We have shown that the average cost of a routing tree towards a geographically scoped destination is lower than that of a randomly distributed destination set which would be comparable to a general multicast scenario. This result can be explained by the relation between geographical distance and network distance. We have shown that there is a large amount of path overlap in most networks when the destinations are geographically clustered. The effect is most visible when the number of destinations is close to a third of the number of nodes in a given network.

We have also compared all three forwarding tree on how equal their edge usage distribution is. This metric is important when it is not desirable to forward traffic over a single link, but rather redundancy is required. We have

shown that a Steiner tree shows the most equal distribution of edge usage, closely followed by the shortest path tree. As expected the minimum spanning tree does not perform favourably on the edge usage metric due to the fixed distribution tree used. We do note that this behaviour might be desired in certain situations.

From the evaluation results it seems that networks with a high average node degree and low average betweenness centrality have the lowest forwarding costs. These characteristics can be used when deciding on a routing tree to use in a specific network.

Based on our results we can conclude that for a relatively small number of destination nodes the minimum spanning tree approach is the least efficient, using more edges and having, on average, a larger total cost. The differences between the shortest path tree and Steiner tree is visible for small numbers of destinations but it is not that great.

Overall we conclude that using a shortest Path tree is likely the best choice for a geocast routing algorithm. The path cost and link fairness are close to that of the Steiner tree while likely requiring less computational resources. A minimal spanning tree seems unsuited as it generally has a higher link usage than both other options.

We can use the outcomes of the evaluation in this chapter to help in the design of a routing algorithm for geocast based on the addressing scheme we presented in Chapter 3. We will develop a shortest path geocast routing algorithm for geocast in the next chapter.

---

## Geographic Routing Algorithm Design

*To facilitate large scale geocast, a fixed-network geographic routing algorithm is needed that can route packets efficiently towards a destination area. Our goal is to design an algorithm that can deliver shortest-path-tree-like geographic forwarding while relying purely on distributed data without central knowledge.*

*In this chapter we present and implement two algorithms for geographic routing. One algorithm is based purely on distance-vector data. Another, more complicated algorithm is based on path data. We show that our purely distance-vector-based algorithm can come close to the number of links used by a shortest path tree when a small number of routers are present in the destination area. We also show that our path-based algorithm can come close to the link usage of a shortest path tree in almost all geocast situations. The contents of this chapter are based on the work presented in "Design & analysis of a distributed routing algorithm towards Internet-wide geocast" [56].*

1. Introduction	
2. Background & Related Work	
3. Geographic Addressing	
4 Geographic Forwarding Trees	
5. Geographic Routing Algorithm Design	6. Geographic Routing Implementation and Evaluation
7. Infrastructure Assisted Contention-Based Forwarding for Geocast	
8. Conclusions and Future Work	

## 5.1 Introduction

In the previous chapter we evaluated different types of forwarding trees for geographic and multicast routing. We have shown that for geocast traffic (in which destination nodes are geographically clustered) the most efficient forwarding tree, taking the forwarding complexity into account, is the shortest path tree. Now that we have an addressing system and know which forwarding tree is most efficient for geocast we can design a routing algorithm. In this chapter we will present the design of two routing algorithms for network-layer geocast using the shortest path tree.

While geocast routing might seem similar to multicast routing in some ways, and by extension is a mostly solved problem, multicast-like routing will not be sufficient in a geocast environment for multiple reasons. Multicast routing algorithms are mostly designed to route packets towards predefined multicast groups that are relatively static. Receivers also need to subscribe to specific groups, which can prove problematic in situations where either the sender(s) or receivers have a high rate of change as it causes a high amount of signalling overhead. Geocast packets on the other hand will have to be routed to a set of routers based on a sender-specified destination area that could contain several or even zero routers. The destination area can take any shape or size which in turn makes predefined areas problematic.

Previous proposals for geographic routing in fixed networks are mostly overlay networks, such as the original proposal by Navas and Imielinski[5], or application layer based, an example being extended DNS[14]. While such solutions have the obvious benefit of being deployable without wide-scale support, there are also multiple downsides to them. The overlay network approach has extra transmission overhead and a build in hierarchy due to the way routers are structured. Application layer solutions suffer similar problems, the main problem with the DNS-based approach being that the DNS server needs to know the locations of all devices. Both solutions are less resilient to change. We propose an alternative approach to the problem: Implementing geocast on the network layer. A network-layer implementation would allow us to use information already available due to unicast routing. The system would also be more resilient due to not relying on the availability of certain servers. Embedding geocast in the network itself will also allow it to route around problems in the network. It would also enable such a system to possibly scale to the entire Internet. Enabling Internet-wide geocast could potentially allow fine grained geographically scoped message transmission for everyone. The main benefit would be that sending hosts on the network do not require any sort of geographical information, they can just send a geocast packet to the router serving them.

In this chapter we answer our third research question: *How can packets*

*be efficiently routed towards a geographical area?* To provide such an efficient geocast solution, the underlying routing protocol will need to take geographic information into account. Traditional routing methods such as unicast or multicast routing have drawbacks in the geocast scenario in that additional signalling or even over the top solutions are needed to enable geographic routing. Unicast routing has the obvious drawback of sending one packet per destination. This would lead to communications overhead when a large number of devices is present in the destination area. On the other hand, the per-packet processing overhead is minimal as unicast routing is well understood and optimized. Multicast routing seems like a better fit as it already supports one-to-many communications. The main drawback for geographically scoped communication is that most multicast routing solutions depend on subscription messages. In a geocast solution, routers will need to report their coverage area and evaluate which routers cover the destination area of a packet. Another drawback would be the requirement of predefined destination areas, as it would have to be known which multicast group covers which area.

The main problem for geocast routing is that routers need to know where they sit on the forwarding tree from the source to all destination. A router can not simply forward a packet on all links that have a shortest path to one of the destinations, as they will likely send packets to routers that have already received such a packet through another link. A solution would be to give all routers full network knowledge, such as in link-state routing, but this approach might not be scalable in very large networks or between networks. For this reason we set out to design a distributed algorithm. Routers should not depend on some central authority or need full network knowledge to make their forwarding decisions. This also prevents us from constructing a least cost (Steiner) tree, as we have described in Chapter 4. In that chapter we also noted that establishing a shortest path tree requires full network knowledge. Without full network knowledge a router can not know its place on the forwarding tree and in turn it is not possible for the router to make a correct forwarding decision. One of the main challenges we face in this chapter is establishing such a tree while routers do not have full network knowledge.

For efficient geographic routing we need a routing algorithm in which geographical areas are central to packet routing. A geographic routing algorithm will need to efficiently forward packets that have a geographic destination to all routers that (partially) cover the destination area. We specifically refer to coverage instead 'being in the area', as the important thing is that devices connected to the router are in the destination area. The most important aspect is the ability to route a packet to multiple destinations using the lowest number of hops possible, without sending duplicate packets over the same link.

We use two area definitions in our geocast system: Coverage area and destination area. *Coverage Area* defines the geographic area that is covered

by a router, devices in this area can be reached through this router. Coverage areas of routers may overlap or even be identical, for example multiple providers servicing the same area. *Destination area* refers to the geographic area to which a packet is sent. This area does not need to be identical to the coverage area of a router, instead routers should calculate if the destination overlaps with their coverage area.

In this chapter we specifically focus on routing between routers (as enclosed by the green rectangle in Figure 5.1), we do not consider end-hosts. We consider the router that initially receives a packet as the source. The actual source might be an end-host connected to that router or some other network. We consider a router a destination if the intersection of one or more of its coverage areas and the destination area is not empty. A router's coverage area (as enclosed by the blue rectangles in Figure 5.1) is defined by one or more rectangles that enclose all end-hosts connected to the router and the area covered by all wireless access devices that are connected to the router.

The main research question we answer in this chapter is: How can we efficiently route geocast packets within a network without full network knowledge? The main contribution of our work is threefold:

- We design a geographical routing algorithm using purely distance-vector-based information,
- We design an efficient geographical routing algorithm based on path information,
- We validate and evaluate the proposed algorithms in terms of link costs

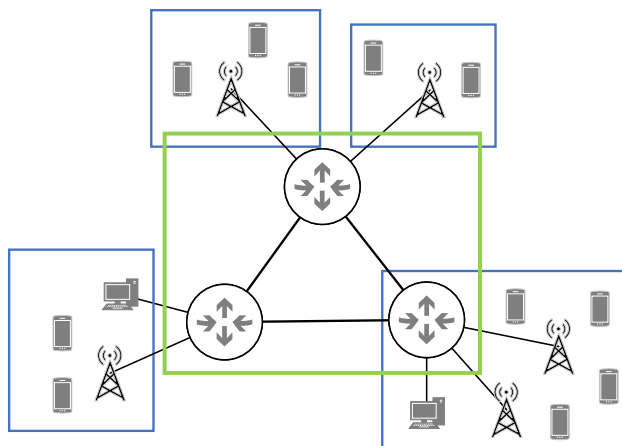


Figure 5.1: Area of interest

on a large set of real world network topologies.

Whereas the second routing algorithm mentioned above is computationally very simple, the first algorithm is very efficient with respect to link cost, at the cost of a somewhat higher link cost for the resulting distribution tree and can be used in cases where network capacity is less important than router resources.

The remainder of this chapter is structured as follows: In the next section we will describe previous work in the area, including our own work on geographic addressing. Section 5.3 will describe the algorithms we have designed to perform geographic routing. We evaluate our algorithms in Section 5.4. Finally we draw our conclusions in Section 5.5.

## 5.2 Related Work

In this section, we will describe previous work done on geocast and geographic routing. We start with describing related work in the wireless domain, moving towards work on geocast in a wired setting.

As mentioned in Chapter 2, geocast was initially introduced by Navas and Imielinski for wired networks [5]. Their approach relied on special routers that know their location and forward packets based on the destination point, circle or polygon. Routers are connected hierarchically: A router that covers a certain area connects to ‘lower’ routers that cover smaller areas within that coverage area. Routers can calculate the intersect of destination and their coverage using the GPS coordinates. Downsides of this approach are the hierarchical router requirement, the need for routers to perform area intersection calculations and the variable length of the addressing (points, circles, or polygons).

In later work from the same authors they studied improved routing cost [22] by approximating the destination / coverage area intersection. They have also studied alternate approaches based on addressing predefined locations [8].

Most work on the topic of geocast has been done in the wireless ad-hoc network context, and especially the VANET case. Overviews of such routing protocols and underlying mechanism can be found in [25], [26] and [27]. In most of these protocols the location of forwarding nodes is tightly coupled with the destination of a packet, a next hop node will generally be in the direction of the target geocast area. The correlation between the position of the next hop node and the location of the destination area does not necessarily exist in a fixed wired network situation. Especially in situations where a network serves several access networks, there might be very little correlation between the forwarding routers and the actual destination area. On the other hand, the fixed wired environment is usually mostly static, this enables route distribution to be effective over long distances.

There are off-course algorithms for multicast routing such as Protocol Independent Multicast (PIM). These could be used in some capacity for geocast routing but they do have some drawbacks. PIM Dense Mode (PIM-DM) relies on an initial flooding stage where routers that are not subscribers send a prune message back to their forwarding neighbor [34]. We would ideally like to not have this behaviour in our geocast system as we believe the number of destination areas that might be addressed in a short time could be very large. Alternatively, PIM Sparse Mode (PIM-SM) relies on an initial Rendezvous Router that routes packets before a shortest path tree is established [33]. Due to the large number of varying geocast destinations and the overhead caused by the Rendezvous Router we believe this approach would not be feasible.

Another, application layer, approach to geocast is to use DNS to resolve geographical areas to an IP addresses by extending the DNS [14]. When the eDNS server is queried for a certain area, it returns the IP addresses of all entries in that region. The eDNS was designed for vehicular networking scenarios, so it would only have to return a list of RSUs in the target area. Scaling the system to track the movements of all vehicles to also allow geocasting from multiple networks at the same time was later found to be somewhat feasible [23]. For a truly Internet-wide deployment such a system would need to scale significantly. DNS delegation would also be complicated if updates are to be distributed through the network in a relatively short time.

As we can see, there is a large number of existing geocast system. However, none of them are optimized to function in large wired networks. The assumption that router location is related to forwarding direction simply does not hold in wired networks. Overlay networks have their own problems in that they rely on centralized knowledge or introduce significant overhead. We attempt to solve these shortcomings by basing our routing system on the addressing scheme proposed in Chapter 3, and developing network level routing algorithms.

### 5.3 Algorithm Design

In this section, we will describe the process we have followed to design our geographic routing algorithms. We will start by describing the path notation we will use in the rest of the chapter. We will then discuss the simplest algorithm possible that will achieve our stated goal: flooding. In the following subsections we will add conditions to build increasingly complex forwarding rules, resulting in our distance-vector-based algorithm. Following that, we will briefly analyse the performance of this algorithm. We continue by describing our path-based forwarding algorithm, followed by short sections on possible link state approaches and hierarchical routing.

We define the primary goal for our geographic routing algorithm as follows:



to deliver a message addressed to a certain area to all routers that cover that area with minimal cost. We will use the hop count (which for a tree we define as the total number of transmission used per packet to reach all destinations) as our cost metric, with a lower number of hops being better. To achieve our goals we choose to target a shortest path tree from the source to all routers that cover (advertise) the destination area. We also have the secondary design goals of limiting the processing overhead and using a system where no per destination signalling is needed. Our algorithms are designed around the assumptions that all links in the network are symmetrical in both connectivity and cost.

Our addressing system allows the aggregation of coverage areas into a single area. This allows a network to advertise its coverage to the outside world as a single address. Our path-based proposal might be suitable for inter-network routing like BGP using the aggregated address for a network. However, in this chapter we mostly focus on a single network scenario to show that our system can function.

### 5.3.1 Path Notation

We will use paths in the network to better explain and eventually build our routing algorithm on. We denote a (shortest) path  $p_{n,m}$  through a network  $G(V, E)$  from a node  $n \in V$  to another node  $m \in V$ .

$$p_{n,m} = n \rightarrow \dots \rightarrow m$$

We define the length of a path  $l = |p_{n,m}| - 1$  as the number of nodes it contains minus one. A path has a minimum length of 1 as  $|p_{n,m}| \geq 2$  (assuming  $n \neq m$ ), and can have an arbitrary number of nodes ( $x_a, x_b, \dots$  where  $x_a \neq x_b$ ,  $x \neq n$  and  $x \neq m$ ) between  $n$  and  $m$ . We denote the  $k^{\text{th}}$  node in such a path as  $p_{n,m}(k)$ . For example,  $p_{n,m}(1) = n$  in the path above. Note that we will use paths in the description of our distance-vector approaches, even though this approach only uses cost information. The path information used in the algorithm is always limited to the next and previous hop, information that would also be available to a purely distance-vector-based algorithm as it can be inferred from the advertised cost information.

### 5.3.2 Flooding

To better explain our approach we will start with the simplest solution that requires no network knowledge: flooding. Using a flooding approach will lead to every packet traversing each link in the network at least once.

Flooding would guarantee that packets are delivered to all addressed destinations. The downside is that there would be significant overhead, especially in larger networks with few routers in the addressed destination area. The total

per packet transmission cost would be constant, equal to the number of links in the network assuming routers would ignore duplicate packets coming in on different links. It should be noted that both ends of a link can send the same packet at the same time, in which case there can be multiple transmission of the same packet on one link.

The transmission overhead is large for such an approach, each link in the network would transmit the packet at least once. The processing overhead is limited to checking if a certain packet has already been processed before, or checking if an incoming packet has been received on the shortest path link to the source, provided that unicast routing info is present.

### 5.3.3 Distance Vector

Using every link in the network is not very efficient; we would like to construct a perfect shortest path tree through the network. We can improve on the flooding algorithm by introducing shortest path knowledge to the routers using a distance-vector approach. A distance-vector algorithm (like those used for unicast) would give all routers knowledge of the shortest-path next hop to all other routers. Coupled with geographic coverage information for these routers, this could enable geocast in a network.

For the following algorithms we assume the routers have the following knowledge:

- Coverage area for every router in the network.
- The cost and next hop for reaching every other router in the network.

A Router receives a packet that has a geocast address in its destination field, as described in Chapter 3. The router then checks this address against the coverage area of the route advertisements it has received. Packets are forwarded to the routers that have overlapping coverage with the destination area. We denote the set of routers with coverage overlap of the destination area as  $D$ . We will now describe 4 distance-vector algorithms in order of increasing complexity that use only the cost to other routers to forward packets to their destinations.

#### DV Algorithm 1

For our first attempt we simply try to limit the flooding in the network to the ‘direction’ of the destinations. We use the term direction loosely here, as the actual geographical location of links and routers does not necessarily correspond to the area they cover. In this simple approach, each router will forward packets it receives on its shortest path link to each of the destinations, except for the link the packet was received on.

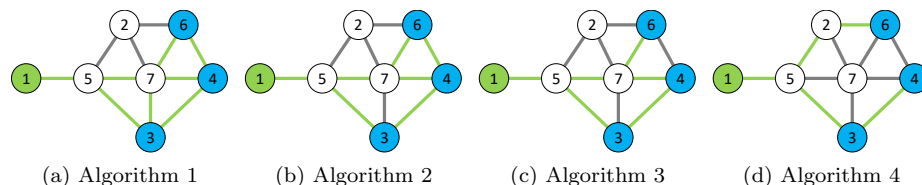


Figure 5.2: Example routing tree of different distance vector algorithms

We define our forwarding function as  $fn(n, D)$ , which returns for a router,  $n$ , the set of neighbours to forward the packet to based on the destination set,  $D$ .

$$fn(n, D) = \forall m \in neighbors(n) : \exists dst \in D : m = p_{n, dst}(2) \quad (5.1)$$

In this function, the path  $p_{n, dst}$  is the shortest path from the current router  $n$  to a single destination  $dst$ . By definition this path passes through a next hop  $m$  in the position  $p_{n, dst}(2)$ , that may be the same as the destination  $dst$  (in which case the path would have a length of 1). With distance-vector information each router is aware of at least two routers on such a path, itself, the destination and the next hop (which could be the same as the destination). The next hop router is simply the router that advertises the destination with the lowest cost (number of hops).

For each router  $n$  that receives the packet we choose neighbours  $m$  to forward to based on if they are the second entry on the known shortest path to a destination  $dst$  in the destination set  $D$  of the packet.

While this simple distance-vector approach leads to a shortest path in the case of a single destination, with multiple destinations the performance is worse. As routers cannot know how they fit on a shortest path tree from the source to each destination, forwarding on the best next hop to all destinations would act like a form of limited flooding. This is caused by each router forwarding the packet on its shortest path links to all destinations. While the algorithm floods the packet in the general direction of the destinations, there is still a large overhead in terms of links used compared to a shortest path tree.

In Figure 5.2a we can see an example network consisting of 7 routers with a source router 1, and destination routers 3, 4 and 6. The links used by our simple algorithm are coloured green. We can clearly see the ‘limited flooding’ effect here, especially in router 7. This router is also forwarding to router 3 as it is the shortest path from the point of view of router 7. It is, however, not on the actual shortest path from the source to that destination and router 3 has already received the packet from another router.

**DV Algorithm 2**

It is obvious that the algorithm we described previously is not very efficient; it uses more links than necessary to reach all destinations. We can improve the performance of the algorithm by ignoring packets that do not arrive on the reverse path interface to the source. The shortest path  $p_{n,src}$  should have the previous hop,  $ph$ , as the second entry. This reverse path check already slightly reduces the average link usage due to routers not forwarding packets for which they are not on the reverse path, but the ‘limited flooding’ problem remains. Routers that are on the reverse path to the source and destination routers will still forward the packet to the other destinations in most cases.

To solve this forwarding problem we add a check for the cost towards the destination. We only forward packets if the current routers cost towards the destination is smaller than the cost reported by the previous hop. This check confirms that the current router  $n$  is actually closer to the destination  $dst$  than the previous router  $ph$ .

We extend our forwarding function  $fn$  with these extra checks. This gives us the function  $fn(n, D, src, ph)$ , where we add the source  $src$  and previous hop  $ph$  of the packet as extra inputs. The output is the set of forwarding next hops as before.

$$fn(\bullet) = \begin{cases} \forall m \in neighbors(n) : \\ \exists dst \in D : m = p_{n,dst}(2) \wedge & \text{if } ph = p_{n,src}(2) \\ |p_{n,dst}| < |p_{ph,dst}| & \\ \emptyset & \text{if } ph \neq p_{n,src}(2) \end{cases} \quad (5.2)$$

In Figure 5.2b we can see the effect. Router 7 sees that the cost for the previous hop (router 5) to reach router 3 is 1. The cost for router 7 to reach router 3 is also 1, thus the packet is not forwarded on that link. Note that in this case we might actually prevent two transmission, as the packet might pass each other on that link as router 3 might send a packet for router 6 through 7. We still see router 4 sending a packet it receives from 3 to 6 as the cost 3 reports is 2 while router 4’s own cost to 6 is 1.

**DV Algorithm 3**

We can further improve the algorithm by checking the cost to reach the source. A router should check if the cost to the source as reported by a candidate next hop router is higher than its own cost to reach the source. Logically, if this was not the case, the candidate next hop should have already received this packet via another path. This prevents the packet from propagating ‘backwards’ in

certain situations. This check improves performance due to the fact that a router receives a packet because it is on the shortest path tree for at least one destination, but evaluates its forwarding for all destinations. The ‘directed flooding’ effect is reduced, but unneeded transmissions are not completely eliminated.

We once again extend our formula  $fn$  by adding the next hop  $nh$  as an input and checking the path length from the candidate next hop  $m$  to the source.

$$fn(\bullet) = \begin{cases} \forall m \in neighbors(n) : \\ \exists dst \in D : m = p_{n,dst}(2) \wedge \\ |p_{n,dst}| < |p_{ph,dst}| \wedge & \text{if } ph = p_{n,src}(2) \\ |p_{n,src}| > |p_{m,src}| & \\ \emptyset & \text{if } ph \neq p_{n,src}(2) \end{cases} \quad (5.3)$$

We can see the improvement of this addition in Figure 5.2c, where the packets router 4 receives from router 3 and 7 are not forwarded to router 6 as the cost for router 4 to reach router 1 is identical to the cost of router 6 to reach router 1.

#### DV Algorithm 4

Our final improvement to this algorithm is to prevent random selection of the next hop when there are two equal length paths for a destination router. The path is now selected through a deterministic method. A router will select the next hop router based on its router ID. We chose (arbitrarily) to use the lowest next hop ID for this. This choice forced packets down the same links when there is a choice. This change will force packets with similar destinations over the same link, leading to less overall link usage. Our path-based algorithm will exploit this deterministic behaviour for its forwarding as we will explain in Section 5.3.5.

The results of this modification are visible in Figure 5.2d. Note that had we chosen to use the highest next hop id, the forwarding tree would be equal to that shown in Figure 5.2c, so this modification does not guarantee a shorter forwarding tree. It does however force packets over somewhat similar paths in cases where shortest paths to multiple destinations share a similar lowest next hop id.

This final distance-vector algorithm can be implemented by looping over all destinations, finding the candidate next hop and evaluating this next hop with the rules given before. This gives us a worst case complexity of  $O(|D| \log(|V|))$

per forwarding operation, where  $|D|$  is the number of destinations and  $|V|$  the number of vertices in the network.

### 5.3.4 Intermezzo: DV Performance

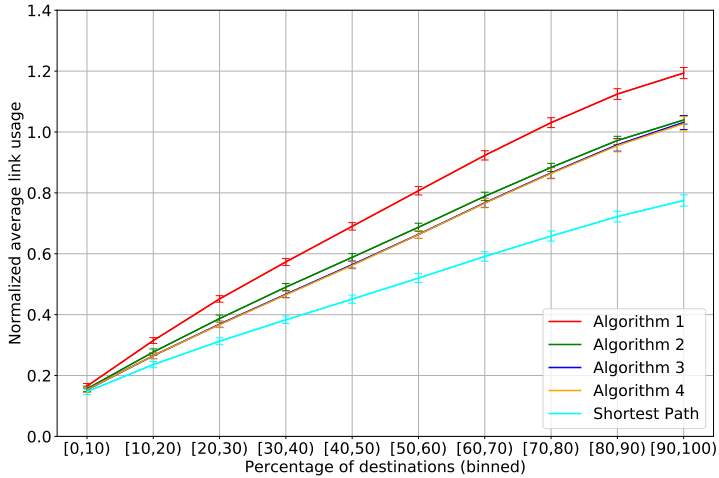
Before we describe our path-based solution we will evaluate the link usage of our DV-based algorithms. We do this to evaluate if our proposed solution satisfies our initial goal of link usage similar to a shortest path tree.

We perform our evaluation over a large set of real world network graphs taken from the Topology Zoo [39]. Our exact evaluation method will be explained in Section 5.4. For now, we will present the performance of our algorithms as a normalized link usage metric. This is calculated by taking the average link usage for a certain number of destinations in a network and dividing it by the number of links in the network. A value of 1 represents all links being used, a value above 1 shows that one or more links are used multiple times. We then bin the number of destinations per 10% and average these numbers over all networks with more than 10 nodes. We compare the link usage of the algorithms described above with the link usage a shortest path tree (cyan line/box) would have.

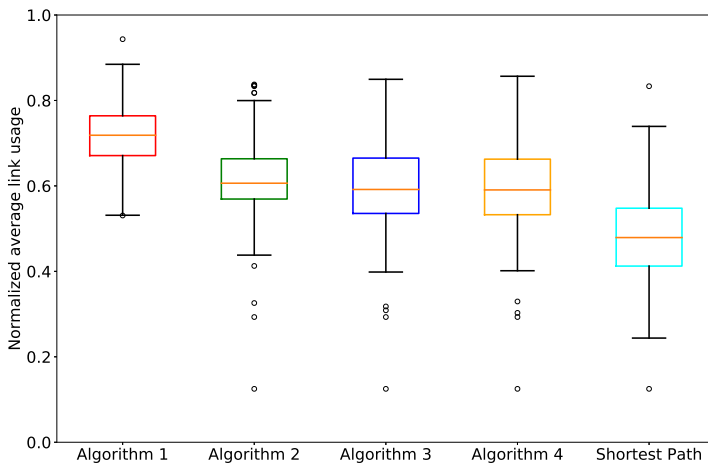
The overall results of our evaluation are shown in Figure 5.3a. We can see that the improvement between the first and second distance-vector algorithm is relatively large, while further improvements provide only minor benefits. Overall, the link usage of our 4th distance-vector algorithm is around 12% larger than the shortest path tree link usage for a small number of destination to around 32% when (almost) all routers are inside the destination area. On average the link usage is 24% larger than our shortest path tree target. This result implies that for situations where only a small number of routers in the network would be addressed, the simple solution might be viable, but for large destination sets the overhead is relatively large.

Figure 5.3b shows box-plots for all algorithms averaged over all numbers of destinations. Using this plot we can compare the overall performance of the different algorithms. We can mainly see that the largest improvement was made with relatively simple additions to our forwarding rules, and that later additions only marginally improve the link cost of the forwarding tree.

After evaluating the performance of the different distance-vector algorithms we conclude that not one comes close to the tree cost of the shortest path tree. In some cases there are even two identical packets traversing the same link when routers forward at the same time, resulting in packets ‘crossing’ each other on the link. Due to these results we will develop a path-based algorithm that will have link usage closer to the shortest path tree.



(a) Binned performance



(b) Overall performance

Figure 5.3: DV algorithm performance

### 5.3.5 Path-Based Distance Vector

While the link usage of the 4th distance-vector algorithm presented is identical to the shortest path tree in the case of the network in Figure 5.2, this does not hold for larger networks. We can clearly see this by looking at the link usage in Figures 5.3a and 5.3b. The main problem with the distance-vector

approach is that routers have no information on their place in the complete forwarding tree in the network. Considering the limited knowledge that is used to calculate forwarding decisions in the DV algorithms we can certainly do better with more information about other paths. As stated before, our aim is to establish a forwarding tree which is as close as possible to the shortest path tree. To prevent the limited flooding effect (unnecessary links are used to reach the destinations) and also keep the amount of information that needs to be distributed in the network relatively low, we have investigated an approach where routers not only know the next hop to each destination, but also know the complete path to other routers. This will function somewhat like the Border Gateway Protocol (BGP) [30]. This information will allow routers to make decisions that can lead to a forwarding tree that is almost identical to a shortest path tree, at the cost of computational overhead and larger route advertisements.

Our proposed path-based algorithm evaluates forwarding decisions on a per destination router basis. The destination address of a packet arriving at a router is mapped to a set of routers advertising (partial) coverage of this destination. A router can now use its shortest path information for each of these destinations to evaluate its forwarding options, similar to the purely cost based algorithms presented earlier.

We develop our path-based algorithm on the basis of the 4th distance-vector algorithm we presented earlier. The forwarding rules from this previous algorithm are extended to no longer use only the number of hops but the entire path to base the evaluation on.

The main problem we try to fix with this path-based approach is that routers have no knowledge of alternate paths through the network while making their forwarding decisions. This can lead to extra transmissions in some cases where for example destination routers think they need to forward the packet to other destinations, while in reality these have already been reached. We attempt to solve this problem by keeping track of two distinct paths towards each other node in the graph when possible. For now we will assume each router knows the one or two (when such an alternate path exists) shortest paths towards all other routers.

We will start by explaining our path distribution method. The algorithm calculating the next hop(s) will be described following this, followed by an explanation of the lowest next hop ID rule. We conclude this subsection by describing situations in which the algorithm fails to establish a forwarding tree that resembles a shortest path tree.



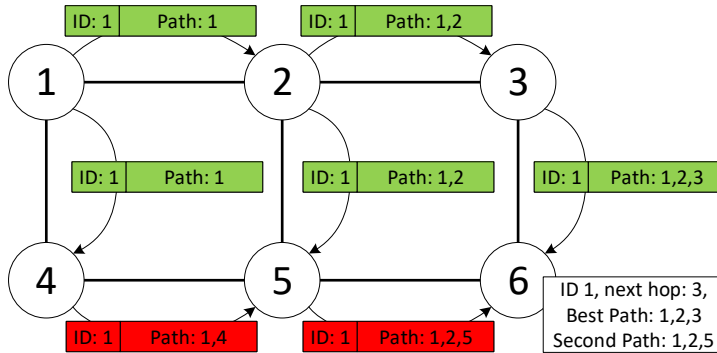


Figure 5.4: Example of route distribution

### Path Distribution

Each router or node in the network advertises its coverage area(s) on each of its links. This advertisement contains the path, initially only the advertising router. A router must append its own id to the path it is propagating.

An advertisement packet contains the *Area*, *Cost* and *Path* to reach it. The *Path* is a set of router IDs:  $Path = ID_0 \rightarrow ID_1 \rightarrow \dots \rightarrow ID_n$ , where  $ID_0$  is the advertising router and  $ID_n$  the previous hop as seen from the receiving router.

The advertising router id should be unique in the network. Using this method, different routers with identical or overlapping coverage areas can be uniquely identified. This allows a router to know which of its links lead to routers that cover the geographic area in the destination field.

A router will transmit the best path towards each router it is aware of to all neighbouring routers, except if this neighbouring router is contained in the path. In that case the router will transmit an alternative, possibly longer, path that does not contain the other router. If an alternate route is not available, for example because the router has no other links, the path containing the neighbour is returned. A neighbour can detect such a loop due to the path information in the advertisement. This system with two distinct paths ensures that routers have knowledge about the existence or non-existence of alternate routes.

Figure 5.4 shows an example of the path for the router with id 1 being distributed through a network with 6 nodes. The paths marked in green are chosen by the receiving routers due to the lowest next hop id rule. The paths marked in red are kept as second best path by the receiving routers.

A router will need to keep track of the advertisements it receives on all its links. Assuming each router has one area it will cover (this could also be zero

or multiple areas), there will be an entry per destination per link resulting in  $|V| \times deg(n)$  entries for a router where  $|V|$  is the number of routers in the network and  $deg(n)$  the node degree of the router itself (the number of links this router has). The average node degree for a network is given by  $2 \cdot |E|/|V|$ , where  $|E|$  is the number of links in the network, giving us a average space complexity of  $O(|E|)$  for a single router.

The resulting routing table has a number of entries that is at least equal to the number of distinct (router, coverage area) pairs in the network. The worst-case scenario is that the number of entries is twice the number of pairs due to the existence of alternate routes. An example would be router 3 in Figure 5.4, which will receive an alternate route from router 6 to router 1 with the path  $1 \rightarrow 2 \rightarrow 5 \rightarrow 6$ , as the best route known to router 6 passes through router 3 itself.

In the event a router detects a link as no longer available, by no longer receiving advertisements on that link, it will stop advertising paths that contain this link to its neighbours. As routers do not propagate paths that contain themselves in the path, eventually all nodes will have updated path information.

### Choosing Forwarding Next Hops

Once a router receives a packet, it will evaluate the packet's source and destination. The destination geocast address is translated to all known routers in the network that (partially) cover that area using the method described in Chapter 3. After the router generates this list of the covering router ids it can move on to the forwarding step.

Initially a router will perform three simple checks for every neighbour it has and every destination router in the destination area. i) The router will start by checking if it is not the destination itself, ii) the neighbour is the next hop on the shortest path towards the destination and iii) the neighbour is not the previous hop. If these checks pass the router can move on to a more complicated check to make the final forwarding decision. These initial checks are described with function  $fn(n, D, src, ph)$ , which takes the current router  $n$ , destination set  $D$ , source  $src$  and previous hop  $ph$  as input values.

$$\begin{aligned}
 fn(n, D, src, ph) = & \forall m \in neighbors(n) : \\
 & \exists dst \in D : dst \neq n \wedge \\
 & m = p_{n, dst}(2) \wedge m \neq ph \wedge \\
 & fn^*(n, src, dst, ph, m)
 \end{aligned} \tag{5.4}$$

A given destination  $dst$  from the set of destinations  $D$  and the candidate next hop  $m$  to that destination are evaluated for forwarding if it passes the checks described before: i) the current router  $n$  is not the destination, ii) the next hop

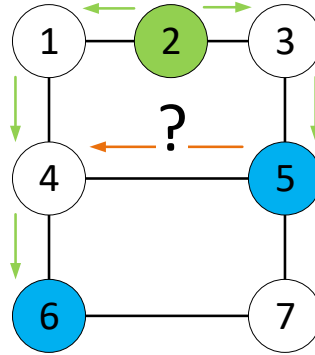


Figure 5.5: Forward lookup from node 5 to 4

$m$  for the destination  $dst$  is not the previous hop  $ph$  and iii) we do not already have the next hop in the forwarding hop list because of another destination in the set. This initial step can be seen in Algorithm 3. If a candidate next hop passes these initial steps we compare three possible paths to each other in  $fn^*(n, src, dst, ph, m)$ .

We will use Figure 5.5 to illustrate our path comparison rules. In this figure, router 2 is the source for a packet that needs to be delivered to routers 5 and 6. We will focus on router 5, which has to decide if it will forward the packet it received towards router 6. There are two shortest paths from 5 to 6, namely

---

**Algorithm 3:** Next hop(s) algorithm
 

---

**Input** : Destination routers list  $D$   
 Previous hop  $ph$   
 Source router  $src$

**Output**: List  $result$  with next hops for the packet

```

1 List  $result$ ; // Initialize list
2 if  $prev\_hop == -1$  then // is entry router
3   foreach  $dst \in D$  do
4     if  $self.nextHop(dst) \notin result$  then
5       result.add( $self.nextHop(dst)$ )
6 else
7   foreach  $dst \in D$  do
8      $nh \leftarrow self.nextHop(dst)$ ;
9     if  $dst \neq self$  and  $nh \neq ph$  and  $nh \notin result$  and
10     $find\_diff(dst, src, ph, nh)$  then // Algorithm 4
11      result.add( $nh$ )
  
```

---

$5 \rightarrow 4 \rightarrow 6$  and  $5 \rightarrow 7 \rightarrow 6$ . Both candidate next hops pass the initial check described in Algorithm 3. We will evaluate the path through router 4 for this example as it has a lower id, the exact reasoning behind this choice will be explained in the next section.

We start by constructing the path from the source to the destination as seen from the next hop router  $p_{src,dst}^{nh}$ . This is the shortest path from the source to the destination that the next hop can know of, given the path information it has.

$$p_{src,dst}^{nh} = p_{src,nh} \triangle p_{nh,dst} \quad (5.5)$$

Here  $p_{src,nh}$  is the shortest path from the *source* to the *next hop* node and  $p_{nh,dst}$  the shortest path from the *next hop* to the *destination*. We define the  $\triangle$  operation as the concatenation of the two paths excluding duplicate routers except the one that connects the paths. This is done in Algorithm 4 on lines 15 to 24.

In our example  $p_{src,nh} = 2 \rightarrow 1 \rightarrow 4$  and  $p_{nh,dst} = 4 \rightarrow 6$  which gives us  $p_{src,dst}^{nh} = 2 \rightarrow 1 \rightarrow 4 \rightarrow 6$ . In some situations both paths could share multiple routers at the start, leading to the exclusion of all but the last of these shared routers in the constructed path. In our example there is only one shared router so it stays on the path as it is also the last.

We now construct a similar path as seen from the previous hop router  $p_{src,dst}^{ph}$ .

$$p_{src,dst}^{ph} = p_{src,ph} \triangle p_{ph,dst} \quad (5.6)$$

Where the path is constructed from the shortest path from the *previous hop* to the *source*  $p_{src,ph}$  and the shortest path from the *previous hop* to the *destination*  $p_{ph,dst}$ . This is done in Algorithm 4 on lines 5 to 14.

In our example  $p_{src,ph} = 2 \rightarrow 3$  and  $p_{ph,dst} = 3 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 6$  which gives us  $p_{src,dst}^{ph} = 2 \rightarrow 1 \rightarrow 4 \rightarrow 6$ . Note that the previous node reports a longer path towards node 6 to node 4 as the shorter path passes through node 4 itself.

Finally we construct the path that the packet will take if we forward it,  $p_{src,dst}^n$ , which is the path as seen from the current router  $n$ .

$$p_{src,dst}^n = p_{src,ph} \triangle n \triangle p_{nh,dst} \quad (5.7)$$

The path is constructed from the path from the *previous hop* to the *source*, the path from the *next hop* to the *destination* and the router  $n$  itself.

Using the example in Figure 5.5, this results in  $p_{ph,src} = 3 \rightarrow 2$  and  $p_{nh,dst} = 4 \rightarrow 6$  which gives us  $p_{src,dst}^n = 2 \rightarrow 3 \rightarrow 5 \rightarrow 4 \rightarrow 6$ . Note that we include the current node in this path! Using all relevant paths, we can compare their lengths in  $fn^*(n, src, dst, ph, m)$ .

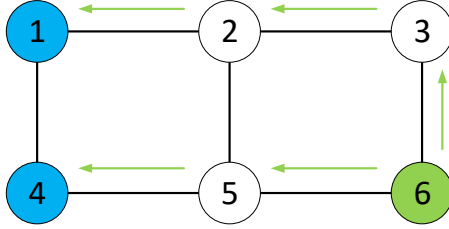


Figure 5.6: Forwarding from node 6 to nodes 1 and 4

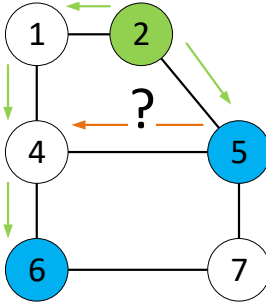


Figure 5.7: Forwarding lookup from node 5 to 4

$$fn^*(n, src, dst, ph, m) = |p_{src, dst}^n| \leq |p_{src, dst}^{ph}| \wedge |p_{src, dst}^n| \leq |p_{src, dst}^m| \tag{5.8}$$

Where  $fn^*$  takes the current router  $n$ , source  $src$ , destination  $dst$ , previous hop  $ph$  and the candidate next hop  $m$ .  $fn^*$  returns true or false.

In our example from Figure 5.5,  $fn^*$  would check if  $|2 \rightarrow 3 \rightarrow 5 \rightarrow 4 \rightarrow 6| \leq |2 \rightarrow 1 \rightarrow 4 \rightarrow 6|$ . This gives us  $(5 \leq 4)$ , which is false. The result is that router 5 will not forward the packet to router 4. If the statement would have been true, the second evaluation would have also been False (in this case it is even the same path).

Pseudo code for the entire forwarding operation is given in Algorithms 3 and 4. In Algorithm 3 we show the initial checks. The router will always forward on the shortest path to the destinations if it is the entry router for the packet in the network (if  $ph == -1$ ). If the packet is received from another router in the same network the router finds a candidate next hop ( $nextHop$ ), checks if it is not the destination, the packet is not returned on the previous hop and that the next hop is not already included in the forwarding next hops

list. If all these checks pass the router will call Algorithm 4 which performs the path comparison checks.

The number of times these path differences have to be calculated per packet is limited by the node degree of the router. Our path difference algorithm combines 4 paths into 2 by looping over their entries. When these 2 resulting paths are equal length another loop over a path is needed leading to a worst case of 5 loops over a path. The worst case for this path length is the diameter of the network giving a worst case time complexity of  $O((2|E|/|V|) * |V|) = O(|E|)$  per forwarding operation.

### Lowest Next Hop ID

When a router has the option of two or more paths of identical length to forward on for a certain destination it has a choice. We let routers base this choice on the lowest router id of the next hop. This makes the choice between paths deterministic and by extension allows routers further in the forwarding tree to know for which destination they are part of the forwarding tree.

Consider the network from Figure 5.6, here a tree is constructed from node 6 to nodes 1 and 4 based on the rules described previously. We choose router 3 as the forwarding hop to router 1 over router 5 because it has a lower id. In Algorithm 4 we can see the lowest id rule implemented in lines 43 to 52.

A similar choice also needs to be made if a router needs to choose between forwarding or not based on path knowledge of its candidate next hop. The router can compare the two paths and check which of the paths has the lowest ID next hop from where they diverge from each other. Using this method a router can determine where it sits in the forwarding tree and for which destinations it should forward.

To illustrate this, we will use a modified version of the network shown in Figure 5.5 with node 2 removed, shown in Figure 5.7. The source is router 2 and the destination are routers 5 and 6 as before. Router 5 has to decide if it forwards the packet it has received to router 6. There are two possible paths available to the router  $5 \rightarrow 4 \rightarrow 6$  and  $5 \rightarrow 7 \rightarrow 6$ , where the first path has the lower next hop ID. This path through router 5,  $2 \rightarrow 5 \rightarrow 4 \rightarrow 6$ , is now compared to the path as seen from the candidate next hop router 4:  $2 \rightarrow 1 \rightarrow 4 \rightarrow 6$ . These paths diverge from each other after router 2, where the best path as seen from router 4 has the lowest next hop ID. Router 5 can now deduce that router 4 has already received this packet and it does not have to forward the packet.

This method of choosing one path over the other allows our system to only use one path towards each destination. In some cases this leads to a forwarding tree that has a slightly higher cost than the ideal shortest path tree, but multiple paths are not used to reach the same destinations.

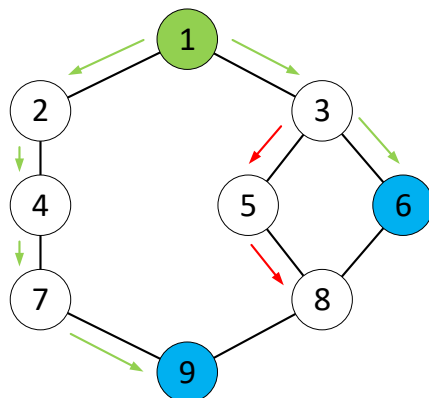


Figure 5.8: Shortest example of forwarding error loop

### Deviation From the Shortest Path Tree

The algorithm using limited path knowledge described before constructs a forwarding tree that is close to the shortest path tree in terms of link usage. However, the algorithm fails to construct such a tree in some specific situations where the network contains loops within loops. A minimal example of one such network can be seen in Figure 5.8. In this figure node 1 represents the source, nodes 6 and 9 are the destinations.

Routers in the small loop will receive advertisements for the source and destination from both their neighbours as they keep track of the two best paths (when multiple paths exist) to the source. Both paths will use the small loop to reach those destinations as the distance is shorter compared to the large loop. The result is that the routers inside the small loop have no knowledge of the alternate path through the larger loop and mistakenly believe they should forward the message for node 9. The router that connects the small loop to the larger loop (router 8 in Figure 5.8) does have this knowledge and will correctly not forward the packet.

The maximum cost of the extra link usage for this topology is half the number of links on the small loop. This occurs in the case where the destination in the small loop is on the side of the higher next hop id from the source (like node 6 in Figure 5.8). In practice, as our evaluations will show in Section 5.4, this problem almost never occurs and when it does the extra overhead caused is minimal.

### 5.3.6 Link State

For completeness we have to mention the option of using a link state algorithm for geographic routing. Using a link state algorithm, which can provide full network knowledge, each node can determine if it is on the shortest path between a given source and destination and base its forwarding decisions on this information.

The benefit of this approach would be the perfect shortest path tree for any given (source, destinations) combination. The main downside would be computational, storage and communication overhead of the full network graph as compared to the other alternatives presented. Each node would have to compute the shortest paths from the source to all destinations for every incoming geocast packet. The worst case complexity of this operation would be  $O(|V|^2 \log(|V|))$ . It is possible to pre-compute the shortest path between each node in the network (given there are no network changes in the meantime). In that case the router would just need to lookup the next hop for each (source, destination) pair, but this would require storing this information.

Link state algorithms would also allow each router to compute a Steiner tree for a given source and destination set, in theory allowing the network to forward using the lowest cost tree possible.

However, the amount of data that would need to be transmitted and the computational overhead for such an approach would be large and scale with the size of the network. Considering these drawbacks we think a link state approach is not feasible and will not consider it further in the remainder of this chapter. We will also show in Section 5.4 that the path-based distance-vector approach already comes close to a shortest path tree in terms of link usage.

### 5.3.7 Hierarchical Routing

Due to the ability of our addressing scheme to aggregate the geographic addresses, as described in Chapter 3, it is possible to advertise an entire network as a single coverage area. This enables geographic routing on a large scale, as each network would not be represented by a single or even multiple coverage areas per router, but by a single unified area.

As an example we take a network covering an entire city. This network could aggregate the coverage area of the routers in a single address. While this single address might not be completely identical to the coverage area of the individual routers (it will be slightly larger in reality), it allows the network to advertise its area in a single advertisement. The same holds for any autonomous system, we believe our system could work for inter-network geocast using this method. Within a network the distance vector or path-based approach could be used based on what the exact requirements are. Between networks the path-based approach would be most appropriate.



Let us consider the previous example of a city covered by multiple networks. A fairly common situation, where we likely have multiple networks in most places given competing wired and wireless Internet providers. In this situation any geocast packet with a destination in this city would need to be routed towards all the networks that cover this city. Within each network the packet needs to be routed towards all routers that have coverage inside the destination area, which in turn should deliver the packet to all connected devices as mentioned in Chapter 3. We will present one such system in Chapter 7.

## 5.4 Evaluation

In this section, we will describe the evaluation of our proposed path-based routing algorithm. Our main goal is to measure the link usage of our algorithms and compare them to the shortest path tree and Steiner tree as used in Chapter 4. We will start with briefly mentioning the tools we used to perform the evaluation followed by the method for destination selection. We then describe how we measure the path-based algorithm's link usage and describe the method we have used to evaluate our algorithms. We will evaluate how close the forwarding tree constructed by the algorithm is to the shortest path and Steiner tree. Our main metric will be the number of links used to construct the tree.

### 5.4.1 Tools

All our evaluations are run over a set of real-world networks taken from the Topology Zoo [39] unless otherwise noted. Using these networks, we hope to more accurately evaluate performance in real-world scenarios as compared to randomly generated ones.

To analyse our algorithms on these networks we use the network library NetworkX [40] for the python programming language. We use this tool to import the Topology Zoo network graphs. We remove all nodes that are not connected to other nodes from these graphs. If a graph is disconnected we use the largest connected part. This set of graphs is identical to the set used in Chapter 4.

On the network graphs we use the route distribution system described in Section 5.3.5 and the forwarding algorithms described in Sections 5.3.3 and 5.3.5. We will describe the exact method later in this section.

### 5.4.2 Destination Distributions

For our evaluation we define two categories of destination sets: Geographically scoped and randomly distributed destinations. We believe these two sets cover most realistic use cases.

### Geographically Scoped Destinations

In most networks that we have evaluated we observe that the geographical distance between two routers and the network distance (number of hop between them) is closely linked. This observation has led us to believe that within a network most geocast traffic will be geographically scoped in its destination router set.

For our geographically scoped destination set we use each node in the network as a source for every possible group of geographically scoped destinations. The destinations are selected based on their location, for  $n$  destinations we select a node in the network and add the  $n - 1$  geographically closest nodes to the set. Each node is selected once, duplicate sets are filtered out as they would represent the same destination area. The source is never included in the destination set. The result is that each possible geographically scoped (source, destination) combination in the network is evaluated exactly once.

### Randomly Distributed Destination

Because in practice it seems unlikely that all geocast destination will be geographically clustered in a network, we also evaluate randomly selected destinations. We believe such a situation can occur when a network ( $A$ ) serves multiple other networks (for example  $B, C, E$ ). Let's assume networks  $B, C$  and  $E$  all cover our destination area. It is unlikely that the connections of these networks to  $A$  are geographically clustered, thus the randomly distributed destination scenario is also important.

As with the geographically scoped destinations we select each node once as the source. For the destinations we select every possible combination of destinations that do not include the source exactly once.

### 5.4.3 Evaluation Method

We evaluate our routing algorithm by running it on a collection of real world networks taken from the Topology Zoo project [39]. We initialize every network by performing the route distribution step until the routing table of each router is stable. This is done by letting the routers exchange information in steps, in each step all routers transmit their path information to all their neighbours.

We evaluate each possible (source, destination(s)) combination, generated in the way described earlier in this section, in the network by inserting a packet with the given destination(s) at the source router and forwarding it until no router has any operations left to perform. Forwarding is performed by the path-based algorithm described in Section 5.3.5 and the distance-vector algorithm described in Section 5.3.3. Each packet is forwarded on all the link(s) this algorithm returns based on the source, destinations and information from the

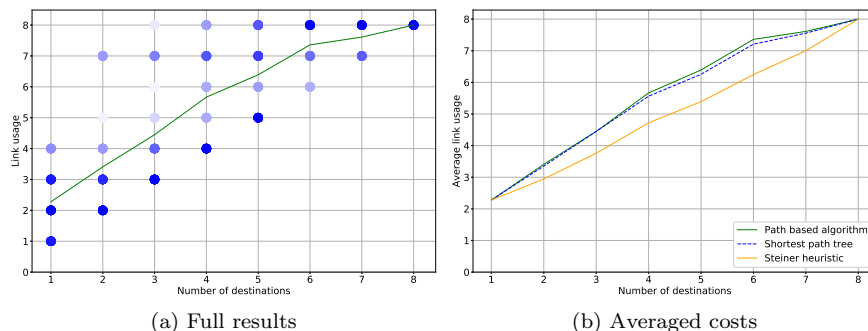


Figure 5.9: Normalized link cost for the network from Figure 5.8

previous and candidate next hop routers. Similar to the route distribution, the forwarding is also performed in steps. Each forwarding step allows all routers to forward a packet if they have any. Effectively, this makes it a time-slot based evaluation of our algorithms.

This simulation is run for a subset of destinations that are geographically scoped and for randomly distributed destinations as described before. We then compare the average number of links used for a given number of destinations to the number of links used by a shortest path tree and a Steiner tree for the same (source, destination) combinations. The values for the shortest path and Steiner tree are taken from the results presented in Chapter 4. The shortest path tree is calculated by finding the shortest paths from the source to all destinations and taking the union of these paths. The Steiner tree is calculated using a well known heuristic [54], which is the same approach we used in Chapter 4.

#### 5.4.4 Evaluation Metrics

To evaluate the routing performance we look at the resulting link usage in our evaluation graphs. This metric will give us an indication of how efficient the routing algorithm performs its goal of establishing a shortest path forwarding tree. We compare the average number of links used per number of destinations.

We define link usage as the number of links that are used to forward a message from the source to all destinations. If a link was used twice (i.e. in both directions) this counts as two link uses. For example the forwarding tree shown in Figure 5.2a uses 7 links while the more efficient algorithm in Figure 5.2d only uses 5 links.

To better illustrate the method we use to present our results we will demonstrate this method for a single network. In Figure 5.9a, we show the

routing cost in links used over all geographically scoped destination locations in the example network from Figure 5.8. The green line represents the average link usage, with the colour intensity for the blue dots showing the relative occurrence of the link usage for a certain number of destinations. The effect of the loop inside a loop (explained before in Section 5.3.5) can be clearly seen here with the cost of 3 and 4 destinations mainly around 4 and 7, but not 5 and 6.

Figure 5.9b plots the performance in link cost of the path-based routing algorithm (green line) against the performance of a shortest path tree (dashed blue line) and a Steiner tree heuristics algorithm (orange line) for the same network used in Figure 5.9a. The green line corresponds to the solid green line in Figure 5.9a. We can see that the performance of the routing algorithm in terms of links used is close to that of the shortest path tree.

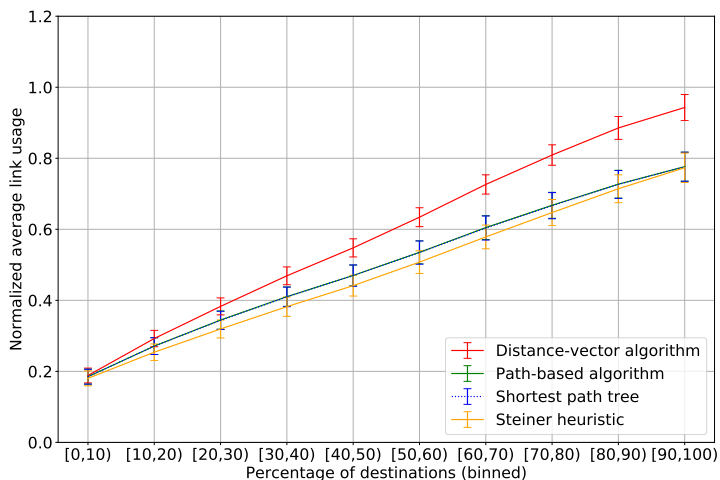
As different networks have different numbers of routers and links, the results for them are not directly comparable. We normalize the link usage to allow us to make this comparison. The normalization of link usage is done by dividing the link usage with the number of links in the network, resulting in a number between 0 and 1. Values above 1 are possible if there are multiple transmission on the same link.

### 5.4.5 Evaluation Results

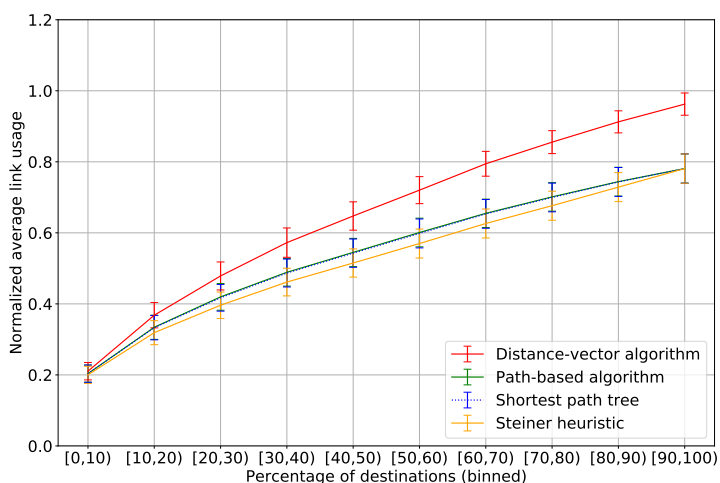
Now that we have described how we evaluate our algorithms, we will show the average normalized link usage of our algorithms compared to a shortest path and Steiner tree in this section.

In Figure 5.10, we show the normalized link usage on the y-axis. The x-axis represents the normalized number of destinations. This normalization is done by binning the number of destinations for every 10%. We show the distance-vector algorithm (algorithm 4), the path-based algorithm, shortest path tree and Steiner heuristic normalized over a subset of real world networks. This figure contains results for a subset of graphs containing the 86 graphs over which we also have a complete set of shortest path and Steiner heuristic results for randomly distributed locations. This set is limited to 86 graphs due to the time needed to evaluate all random combinations in larger networks and contains Topology Zoo networks with 20 nodes or less.

Figure 5.10a shows the geographically scoped results over the 86 graphs while Figure 5.10b shows the same values but for randomly distributed destinations. We believe such a scenario could occur in transit networks were the points networks connect to each other do not necessarily correlate with their geographic coverage, especially if these networks cover the same area. In both scenarios the line for our path-based algorithm and the shortest path tree almost completely overlap. As expected based on our previous work, the average extra link usage



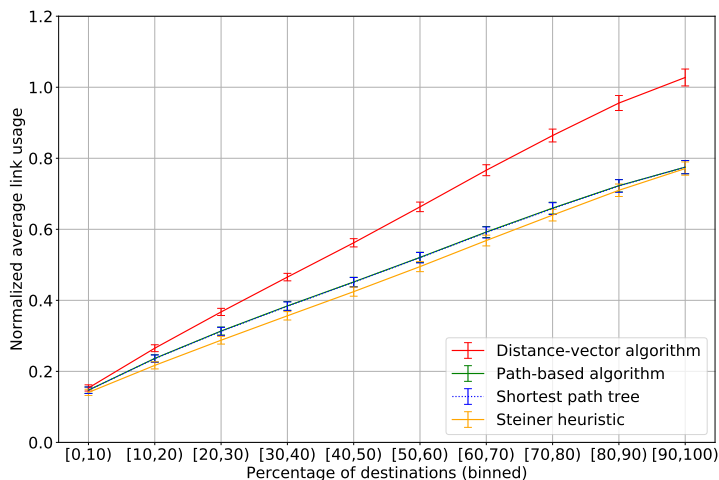
(a) Geographically scoped



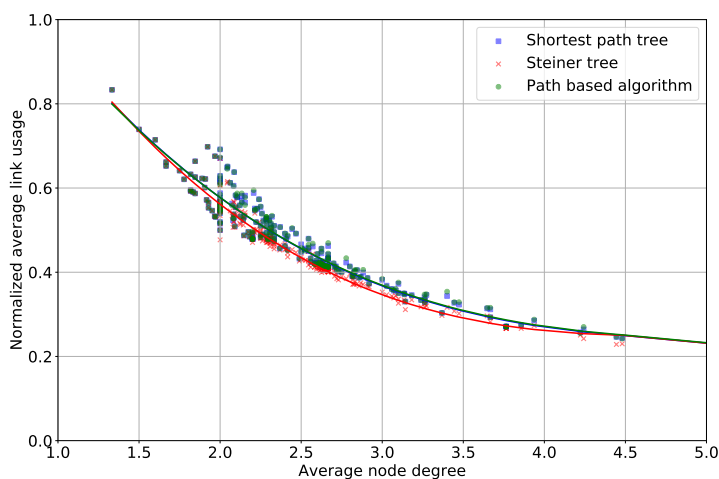
(b) Randomly distributed

Figure 5.10: Normalized link cost for real world graphs comparing geocast and multicast

compared to the Steiner tree is relatively small. We also note that our best purely DV-based algorithm performs reasonably well when the number of destination in a network is small. The DV-based algorithm performs worse when the number of destinations is high, as in larger networks the forwarding



(a) Link usage set against percentage of destinations addressed



(b) Link usage set against the average node degree of the network

Figure 5.11: Geographically scoped normalized link usage for all real world graphs

decision can not be made correctly with the limited information available.

Figure 5.11 contains geographically scoped results over the complete set of 227 real-world network graphs we used for the evaluation. Figure 5.11a shows results over this set using the same method used for Figure 5.10a. We can see

that over this larger set of graphs, which also contains larger networks, the average cost for our path-based algorithms is similar, but the link usage of DV-algorithm 4 is slightly higher implying that its performance might degrade depending on the network size.

In Figure 5.11b we show the normalized average link cost of an entire graph (the average number of links used over all geographic destination combinations divided by the total number of links in a graph) plotted against the average node degree of the network. The average node degree is the average number of links a router has, this is an indication of how well connected a network is. We can see that the average link cost of the path-based routing algorithm is close or equal to that of the shortest path tree. In general, the cost for geographically scoped destinations is close to that of the ideal Steiner tree. We also observe that the more well connected a network is, the lower the average cost to reach a certain destination area.

### 5.4.6 Path Knowledge vs. Hop Knowledge

Our distance-vector-based algorithm described in Section 5.3.3 has a higher link usage compared to the path-based algorithm described later. It does however have some benefits over the better performing algorithm:

- Lower communications overhead due to DV like cost exchange
- Lower forwarding complexity

The communication overhead depends on the size of the network. The larger the network is, the longer paths are that our path-based algorithm has to communicate through the network. The distance-vector algorithm only has to exchange a single cost metric per router in the network. This also has the benefit of reduced storage requirements for each router.

We have shown that the forwarding complexity for the distance-vector algorithm is  $O(|D|\log(|V|))$ , where  $|D|$  represents the number of destination routers and  $|V|$  the number of routers in the network. The path-based algorithm has a lookup complexity of  $O(|E|)$ , where  $|E|$  is the number of link in the network. The forwarding complexity for the distance-vector algorithm is significantly less compared to the path-based approach, at the cost of a higher forwarding tree cost.

In the 227 real networks, on average the distance-vector-based algorithm has 28% worse link usage compared to the more complex algorithm with a standard deviation of 0.26. The best case was identical link usage with the worst case 112% extra links used. We can conclude that in some networks the extra transmission overhead could be an acceptable trade-off for the lower computational burden put on the routers themselves. There is no single perfect

choice here, the algorithm will have to be selected based on the network. We do observe that the overhead of the distance-vector algorithm is lower in smaller networks, and is almost always high in larger networks of more than 15 nodes.

## 5.5 Summary and Conclusions

In this chapter, we have presented a purely distance vector and a path-based algorithm for geographic routing. We have also evaluated the link usage of these algorithms on a set of real world networks.

Our best distance-vector-based algorithm performs relatively well, and in the worst case has an only 32% higher link cost compared to the shortest path tree. In a situation where the entire network is not addressed this overhead is even lower.

We have shown that our path-based algorithm can construct forwarding trees to multiple destinations that are close in link cost to the shortest path tree from the source to the destinations. Our proposed algorithm establishes forwarding trees that are almost equal to the shortest path tree and close to the optimal Steiner tree in link cost. The algorithm can improve on the distance-vector-based algorithm, especially in situations where a large number of routers in the network ( $> 25\%$ ) are part of the addressed area.

Depending on the network type the distance vector algorithm might actually be preferable in certain situations where the extra computational overhead in the router does not outweigh the extra transmission overhead in the network. We expect that this routing approach combined with a hierarchical approach in which autonomous systems advertise one or more areas will eventually allow Internet-wide geocast to become a reality.

The evaluation performed in this chapter has one major drawback: Steps happen in sync with each other, we will not see any effects related to timing and other dynamic behaviour in this evaluation. To evaluate our algorithms in a more realistic scenario we will need to evaluate them in either a network simulator or implement the algorithms and evaluate those in a real or emulated network. In the next chapter we will present and evaluate implementations of the algorithms presented in this chapter.



**Algorithm 4:** find\_diff (Find path difference)

---

```

Input  : Candidate next hop next_hop; The previous hop prev_hop;
           Destination dst; Source src
Output : Boolean result
1  nh_src = self.pathTo(next_hop, src);
2  nh_dst = self.pathTo(next_hop, dst);
3  ph_src = self.pathTo(prev_hop, src);
4  ph_dst = self.pathTo(prev_hop, dst);
5  ph_dst_removed = ph_dst;
6  foreach n ∈ ph_src do
7  |   if n ∈ ph_dst_removed then
8  |   |   ph_dst_removed.remove(n)
9  |   ph_src_removed = ph_src;           // Copy list
10 foreach n ∈ ph_dst do
11 |   if n ∈ ph_src_removed then
12 |   |   ph_src_removed.remove(n)
13 |   ph_src_dst = ph_src_removed;       // Copy list
14 |   ph_src_dst += ph_dst_removed;      // Add lists
15 |   nh_src_removed = nh_src;         // Copy listp
16 foreach n ∈ nh_dst do
17 |   if n ∈ nh_src_removed then
18 |   |   nh_src_removed.remove(n)
19 |   nh_dst_removed = nh_dst;         // Copy list
20 foreach n ∈ nh_src do
21 |   if n ∈ nh_dst_removed then
22 |   |   nh_dst_removed.remove(n)
23 |   nh_src_dst = nh_src_removed;      // Copy list
24 |   nh_src_dst += nh_dst_removed;     // Add lists
25 |   p_src_dst = ph_src;
26 |   p_src_dst += nh_dst;             // Path through this router
27 result = false;
28 if length(ph_src_dst) ≥ length(p_src_dst) then
29 |   if self ∈ nh_src then
30 |   |   result = true;                 // on nh to src
31 |   else if length(nh_src_dst) > length(p_src_dst) then
32 |   |   result = true;                 // Other path is worse
33 |   else if length(nh_src_dst) == length(p_src_dst) then
34 |   |   temp = [self] + ph_src;
35 |   |   for i = 1; i < length(nh_src) + 1; i ++ do
36 |   |   |   if nh_src[length(nh_src) - i] != temp[length(temp) - i]
37 |   |   |   |   then
38 |   |   |   |   |   if nh_src[length(nh_src) - i] > temp[length(temp) - 1]
39 |   |   |   |   |   |   then
40 |   |   |   |   |   |   |   result = true;
41 |   |   |   |   |   |   |   break;
42 return result;

```

---



---

## Geographic Routing Implementation and Evaluation

*In this chapter we implement and further evaluate the path-based and distance-vector-based forwarding algorithms presented in Chapter 5. We describe the structure of our implementation and the information distribution format. We evaluate our implementations in a emulated network environment based on the same network graphs used in the previous chapters. We confirm the link usage results of Chapter 5 and show that our algorithms converge relatively quickly following link loss and restoration in the network. The contents of this chapter are based on the work presented in "Implementation and Evaluation of Distributed Geographical Routing" [57] and "Design & analysis of a distributed routing algorithm towards Internet-wide geocast" [56].*

1. Introduction	
2. Background & Related Work	
3. Geographic Addressing	
4 Geographic Forwarding Trees	
5. Geographic Routing Algorithm Design	6. Geographic Routing Implementation and Evaluation
7. Infrastructure Assisted Contention-Based Forwarding for Geocast	
8. Conclusions and Future Work	

## 6.1 Introduction

In the previous chapter we have presented two routing algorithms for geocast: One distance vector and one path-based algorithm. In this chapter we will present a prototype implementation of these algorithms. The evaluation of both algorithms we have performed for the previous chapter only evaluated the algorithms in a static environment and did not simulate any timing effects. In this chapter we will evaluate both algorithms in a changing network topology.

Evaluating the distance vector and path-based algorithms on a set of graphs gave us information on their efficiency in creating forwarding trees for geographically scoped broadcast. This method has the downside of not taking into account timing based events and a dynamic topology caused by link drops. To resolve these shortcoming the main research question for this chapter is: *How do our algorithms perform in real network scenario?*

To answer this question and better validate the proposed algorithms, we will need to evaluate both our algorithms in a more realistic scenario. This can be done by either running both algorithms in a simulator, or evaluating it on an actual or emulated network. As both options require (partially) implementing the algorithms, we choose to evaluate them in an emulated network. Evaluating an implementation of the forwarding systems allows us to better evaluate properties such as convergence behaviour compared to the more ‘static’ evaluation we used in the previous chapter. In the purely graph based evaluation of the previous chapter we do not evaluate effects caused by timing differences, as every action effectively takes place within a time-slot. By using an implementation of the algorithms and evaluating them in a network emulator we can gather more accurate data and also validate the results of the previous chapter.

This chapter has four main contributions:

- We will present implementations of the distance-vector and path-based algorithms described in Chapter 5.
- We evaluate the link usage of these implementations and confirm the results of the previous chapter.
- We evaluate the convergence time of the algorithms after initialization, link loss and link restoration.
- We evaluate the packet loss and duplicate packets during convergence caused by link loss and link restoration.

This chapter is structured as follows: We will first present the details of our implementation in Section 6.2. In Section 6.3 we will describe the evaluation method we have used to evaluate our implementation. We show the results

of our evaluation in Section 6.4. In Section 6.5 we discuss open issues of our prototype. Finally, we draw our conclusions in Section 6.6.

## 6.2 Implementation

In order to better validate the proposed algorithms we have implemented both the distance-vector and path-based geographic routing algorithms. Both are backed by a protocol for advertising paths, as presented in Chapter 5. Routers will periodically send path information to their neighbours. These packets contain the best paths a router has to all other routers it is aware of. We also use this path distribution method for our DV implementation to eliminate timing differences caused by the speed of the underlying information distribution system. We simply only use the associated cost information for the distance-vector implementation. We will start by presenting the general structure of the software, followed by our information distribution, information tracking and forwarding approach.

### 6.2.1 Software Structure

Our software consists of 8 main components, which are shared by both implementations: The packet receiver, advertisement parser, packet forwarding, link path table, general path table, coverage table, route advertisement (RA) generator, and packet sender. These components and their mutual relationships can be seen in Figure 6.1. The general structure of the software is mostly identical for both algorithms, only the “packet forwarding” section has algorithm specific behaviour.

The packet receiver handles all incoming packets. Packets that have a geocast destination address are passed to the packet forwarding system while advertisement packets are passed to the advertisement parser.

The route advertisement parser parses the advertisement packet into path data and coverage data. Path data consists of a path for each router included in the advertisement as described in Chapter 5. This data is given to the per link path table and the general path table. Coverage information consists of a geographic address of the coverage area and the id of the covering router. This information is passed to the coverage table.

The per link path table is a table of all path advertisements received on a link. It is essentially the best path table of the router on the other side of the link, with the possible exception of paths that include the current router (unless no other path was available).

The general path table contains the best and second best known path towards all other routers in the network. It also includes the best next hop for each destination. Due to our lowest id next hop choice this is not necessarily

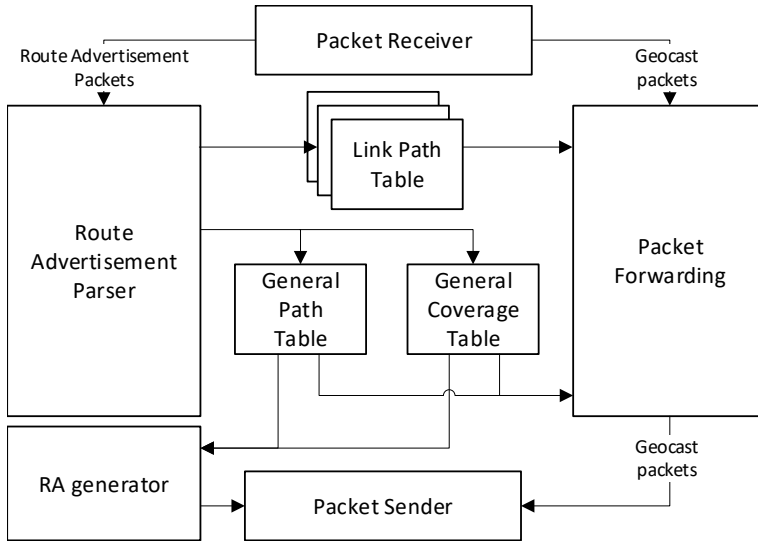


Figure 6.1: Global structure of the routing software

the same as the best known incoming path as they might not following the exact same path.

The coverage table is simply a mapping of coverage areas to the router id of the routers covering those areas using the addressing method described in chapter 3. The Packet forwarding system receives geocast packets and actually runs the forwarding algorithm based on the information it receives from the coverage, path and per link tables. It gives the packet including a list of links to forward it on to the Packet sender.

The route advertisement generator uses the information from the path and coverage table to generate a route advertisement for each link the router has. When for a link, the neighbouring router is included in the path the second best known path is sent when available.

Finally the packet sender, as its name implies, sends out all packets on the correct interfaces.

### 6.2.2 Route Distribution

Routers periodically send route advertisements consisting of all shortest paths and coverage areas known to the router. They take the following form: Advertising router ID (1 byte), Number of advertisements in packet (1 byte) followed by the actual advertisements. A single route advertisement consists of: the coverage area (16 bytes: an IPv6 address), the path length (1 byte),

the covering router id (1 byte) and the advertised path (of path length bytes, with a minimum of 1).

The best route table is used to generate these route advertisement packets. As described in Section 5.3.5 of Chapter 5, a neighbour specific modification is made if a path contains the neighbour it is transmitted to. If this happens the second best route is used if it exists, letting the neighbour know that there is another path. If there is no alternate path, the available path is used. This lets the receiving router know there is no other known path, except the return path to a certain other router.

Assuming perfectly synchronized distribution messages between routers (every router immediately tells its neighbours about new information), it follows that the theoretical time needed for every node to have at least one path to each other node is given by  $t_c = t_{ra} \cdot d$ . Where  $t_c$  is the time needed to converge (in seconds),  $t_{ra}$  the time between route advertisements and  $d$  the diameter of the network (the maximum distance, or hops, between any two nodes). This is the theoretical lower bound for convergence time, in practice the time needed for convergence will always be higher.

### 6.2.3 Distance-Vector-Based Forwarding

To provide a comparison to the path-based algorithm we have also implemented our distance-vector-based forwarding method. To accurately compare the distance-vector to the path-based algorithm we have kept most of the systems identical. We use the same distribution and information tracking systems and only modify the forwarding decision making system. We simply use the length of the best path in the router's global table as the cost metric. This approach allows us to use the distance-vector algorithms without changing the underlying distribution system. The benefit is that the results from running these algorithms become comparable, there are no timing differences as a result of different distribution systems.

This implementation follows the rules we describe in Section 5.3.3 of Chapter 5 for the 4th DV algorithm, which includes the lowest next hop id rule. Overall this approach results in a relatively simple forwarding algorithm at the cost of a non-perfect forwarding tree. In some situations this can lead to links being used that are not needed and can even cause packets to be delivered twice at the same router as we have observed in Chapter 5.

### 6.2.4 Path-Based Forwarding

The path-based forwarding procedure is based on the known paths to the source and destination(s) as described in Section 5.3.5 of Chapter 5. To briefly recap: For each destination  $d$  in the set of destination routers  $D$  we evaluate

the forwarding path through the current router. We combine our best paths to the source and  $d$  into a candidate forwarding path. We compare this path to a similar combination of paths reported by the candidate next hop and the previous hop (this is the main reason we keep a path table for each interface). If our candidate path is better than the other two options the packet is forwarded on this path. This approach ensures that our path is indeed the shortest path from source to destination based on our limited knowledge. Using this method we can construct forwarding trees that are close to the shortest path tree in terms of link usage, as we have shown in the evaluation of the previous chapter.

## 6.3 Evaluation

To validate our algorithms and their implementation, we have tested them on a large selection of networks during different convergence scenarios. In this section we will explain the general evaluation approach which will explain which metrics we are interested in, followed by the different scenarios we are evaluating and how these are selected for each evaluation run.

### 6.3.1 Evaluation Metrics

We want to learn how our algorithms will function when a network topology changes. To achieve this goal we focus our evaluation on the convergence times and the behaviour of both protocols during convergence following a link drop and subsequent link restoration. We evaluate our system on the following timing values related to convergence:

- Initial convergence time: the time in seconds it takes the algorithm to converge after initialization.
- Convergence time following a link drop: the time in seconds it takes the algorithm to converge after a link is dropped from the network.
- Convergence time following a restored link: the time in seconds it takes the algorithm to converge after a previously dropped link is restored in the network.
- Packet delivery restoration time following a link drop: the time in seconds it takes for all destinations to receive packets again after a link has been dropped.

We assume an algorithm to be converged at the time the last instance of the algorithm has made an update to its forwarding table. This results of the fourth metric should differ between the distance-vector and path-based



algorithms, the first three should be the same as they use the same underlying route distribution system.

We further evaluate the number of packets that are lost during protocol convergence. For this last metric we take into account the number of destinations that did not receive a packet. We take the sum of missed packets at all destination routers. For example, if 2 out of 3 destinations did not receive a certain packet then these two packets are considered lost. Using this method we hope to accurately represent the number of deliveries that were missed during the convergence time.

We will also look at the possible multi-path issues during a convergence phase, in which packets are delivered to a router over multiple paths. For this metric we take the sum of duplicate deliveries at all destination routers. Ideally these duplicate deliveries should never occur, but it is likely unavoidable during protocol convergence due to incomplete or outdated network knowledge in different routers. In the case of the distance-vector based protocol, we expect larger numbers of duplicate packets as the information to build a perfect shortest path tree is just not available.

As a base-line for how efficient the protocols construct their forwarding path we will look at the link usage. We use a normalized link usage metric to more easily compare efficiency between different networks. This normalization is done by describing the link usage as a fraction of the total number of links in a network. This metric is identical to the one used in the evaluation in Chapter 5, and should show comparable results.

### 6.3.2 Evaluation Run Overview

Before each simulation run we create the network using Mininet [41]. This is a tool that can create a virtual network and virtual hosts on a single computer. We will describe the exact procedure in the next subsection.

After the network has been created we start the routing processes on each virtual host and give the routing algorithm time to converge. We have set this waiting time to 25 seconds as we have not observed the convergence taking longer than this time. Routers are configured to exchange route advertisements every 0.5 seconds. Route entries expire after two seconds, leading to a relatively fast update time.

After this initial time, we randomly select a source node and a destination area. We then start transmitting packets from this source. 10 seconds after the transmissions start, we drop a randomly chosen link on the forwarding tree for those packets. We measure how long it takes the algorithm to converge again and the effect this has on the packets that are in transit during this time. 50 seconds after the link drop, we restore the link and we measure the convergence duration again. We also measure the number of packets lost during the entire

run.

The distance vector and path-based algorithm are run on the same set of networks with the same sources and destinations. The link that is dropped from the network is also kept identical between the two different algorithms.

### Network Creation

As with the evaluation in Chapter 5, we use real world network graphs taken from the Topology Zoo project [39] to evaluate our protocol on a set of realistic networks. We use a total of 162 networks ranging in size from 6 to 51 routers, with an average of 26 routers. This is the same set of networks used for the evaluations in Chapters 4 and 5 limited to those with 51 or less nodes. This size limitation was chosen as emulating larger networks proved problematic due to resource limitations. In case a network is not connected, we use the largest connected part. We import these networks into a Mininet virtual network [41] by generating a new node running our routing implementations for each vertex in the graph and establishing a link (1 Gbit/s Ethernet) between nodes if there is a corresponding edge in the graph.

### Node Locations

The latitude and longitude location (as given in the Topology Zoo) of the node is used to generate a coverage area of a size corresponding to depth 10 of our addressing scheme (Chapter 3) to ensure routers cover reasonable portions of a network. For each run within a network, a destination area is randomly generated. This area can cover between 10% and 95% of the bounding box containing all nodes in the network, ensuring we have destination that cover different amounts of destination areas. As mentioned before, we randomly select a source node for each run. The source can be inside or outside the destination area.

### Link Drop Procedure

To simulate a link drop we set both ends of a link to have a 100% packet loss rate using the Linux network emulator [58] functionality. The link to drop is chosen from the set of links on the shortest path tree from the source to the destinations. With this link selection we try to guarantee that the dropped link will at least have some effect on the forwarding situation. The dropped link is chosen in such a way that it does not lead to a disconnected network, so the algorithm can actually converge to the new situation and still reach all destinations. We choose this link by constructing the shortest path tree from the source to all nodes in the destination area and randomly choosing a link on this tree to drop. Later in the run we restore this link by returning the loss

rate on both ends to 0%. Due to the way a dropped link is chosen and the way our algorithms construct their forwarding trees, it is not guaranteed this link is on the forwarding tree all the time. For both algorithms the dropped link was on the forwarding tree in 92% of the runs

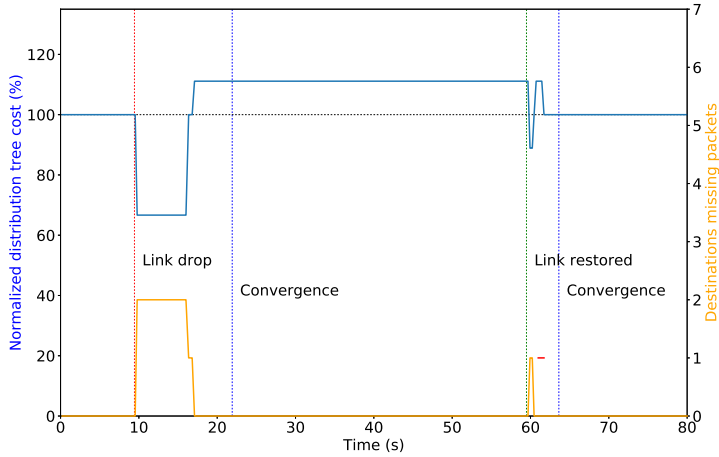
## 6.4 Evaluation Results

In this section we will present the results of the evaluation scenarios described in the previous section. We start with a few examples runs to better explain the method we use to collect the experimental results. For the overall results we start with an overview of our algorithm's performance in a converged state and continue with the convergence time of the protocol in different network states. We end the section with an analysis of packet loss and duplicate packet delivery during protocol convergence.

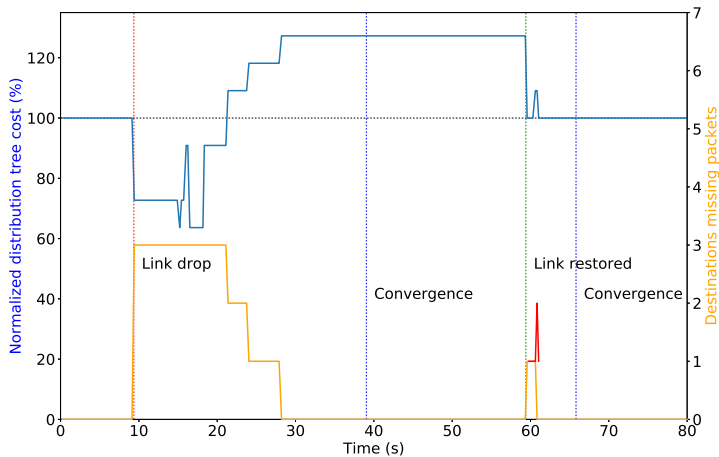
### 6.4.1 Example Runs

To better explain our experimental results, we show the convergence behaviour of runs in two different networks in Figure 6.2. On the left y-axis (blue line) we show the normalized distribution tree cost where 100% corresponds to the link usage of the initial distribution tree. On the right y-axis (yellow line) we show packet loss as the number of destination routers that did not receive a packet at a certain time. The x-axis shows the elapsed time from the start of the packet transmissions in seconds. The time of the link drop is marked as a dotted red vertical line and the link restore time with a dotted green vertical line. The dotted blue vertical lines mark the time at which the protocol has converged to the new situation.

We can see that immediately following the loss of a link there is a period of packet loss, but packet delivery is restored before the network has fully converged. This can be explained by the fact that the link loss (almost always) occurs on a path that is used, resulting in local convergence before the protocol in the entire network has had time to converge. A similar pattern can be seen following the link restore, but the packet loss is minimal here. This is likely caused by the fact that information about new links propagates faster compared to information of lost links due to the way that time-outs function. The small red lines that can be seen following the link restoration represent packets that arrived multiple times at a router in the network. Packets are sometimes routed over multiple paths during convergence, temporarily increasing link usage as can be seen in the graphs. The normalized distribution tree costs decreases after the link loss as the forwarding tree does not reach the destination nodes any more. Eventually the protocol converges and the normalized cost exceeds 100%, due to the larger tree that is needed to route around the missing link.



(a) Example run 1



(b) Example run 2

Figure 6.2: Examples of convergence behaviour on different networks

Following the restoration of the dropped link we can see the normalized cost return to 100% as the protocol returns to using the initial distribution tree.

### 6.4.2 General Link Usage

The link usage of an algorithm shows how many links are used to deliver a packet to all destinations, less links used being considered better. We normalize

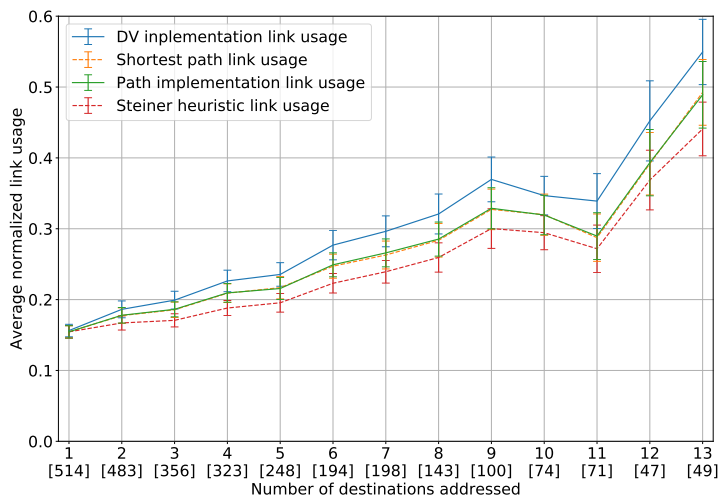


Figure 6.3: Normalized link usage in a converged network

this metric like we did in Chapter 5, by dividing the cost by the total number of links in a network. We compare the link usage of both our algorithms against the shortest path tree and Steiner heuristic. Both shortest path tree and Steiner heuristic link usage are obtained in the same way as in the evaluations in the previous chapters.

In Figure 6.3 we plot the link usage of both our implementations against the shortest path tree and a Steiner tree for the same (source, destination) tuples for all 162 networks. The shortest path and Steiner tree link usage are a subset of the data used in Chapters 4 and 5, limited to the (source, destination) tuples used here. On the y-axis we show the average of the normalized value of the link usage, meaning the number of links used to reach all destinations divided by the number of links in the network. The x-axis shows the number of destinations addressed and below that, the number of runs with this number of destinations in brackets. The error bars show the 95% confidence interval for that value. There are more runs with a lower number of destinations as there are simply more possible (source, destination) tuples. Runs with more than 10 destination were not numerous enough to provide statistically significant results, as can be seen by the growing error bars.

In general our path-based protocol establishes forwarding trees that use a comparable number of links compared to a shortest path tree. In our 2916 runs there were only 392 runs (13%) where the path-based algorithm established a path with a different number of links than the shortest path tree. In most of these cases (238) this path was one link longer, and in some rare cases (154) one

shorter. The former can in some cases be explained by minor inefficiencies (as described in Chapter 5), and the latter difference can be explained by situations where there are multiple shortest paths from the source to a single destination. If the chosen path overlaps with that of another destination the resulting link usage can be lower compared to the situation where the other path was taken.

The distance-vector algorithm has a higher link usage compared to the path-based one. The main cause of this is that multiple links are sometimes used to reach the same destination. The higher transmission cost is offset by a simpler algorithm. For this algorithm, 1130 runs (39%) had a different link usage compared to the shortest path tree. A total of 1037 runs had a higher link cost while 93 runs had a lower cost. As with the path-based algorithm, the lower link costs were caused by situations where multiple shortest paths were available, and some overlapped. This result corresponds with the results from Chapter 5.

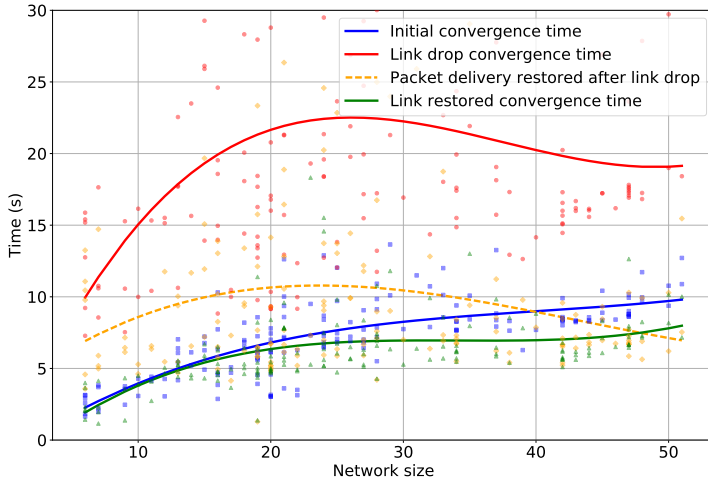
We can conclude that our path-based protocol performs mostly as expected with respect to the number of links used to construct a forwarding tree. Our distance-vector based protocol has significantly higher link usage. There are no surprising results here, these results are all in line with the graph-based analysis done in Chapter 5.

### 6.4.3 Convergence Time

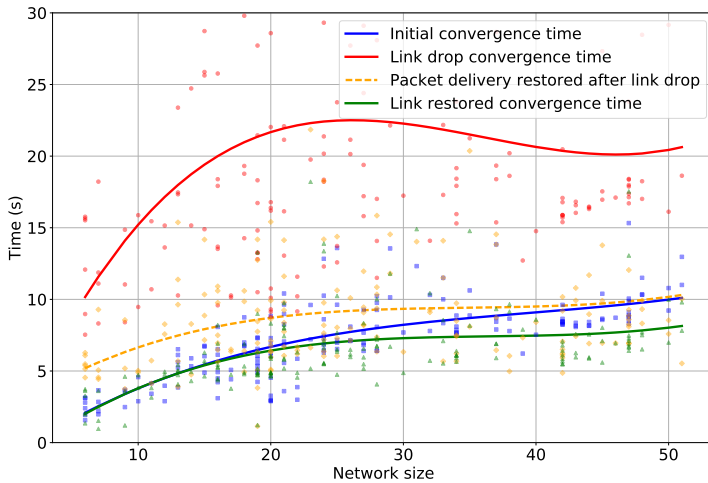
One of the more important aspects of any routing protocol is the time it takes to converge following a change in the network. The correlation between the network size (the number of routers in the network) and the time convergence takes during the evaluation is shown in Figure 6.4 for path-based and DV-based routing. In these figures we show per network averages as points, the lines represent the average over all runs. We can see that the initial convergence time (blue line) of the network shows a strong correlation to the network size. As a larger network mostly corresponds to more possible paths it correlates with the time it takes for the algorithm to converge.

Both the path-based and distance-vector graphs show comparable results for the convergence times. This is the result of both using the same underlying route distribution method. Small differences are caused by differences in the exact time nodes started in relation to each other combined with the time-out of 2 seconds.

On average the convergence time after a link drop (red line) is significantly larger than that of adding or restoring a link (green line) to the network. As noted before this can be explained by the time-out of 2 seconds that needs to occur before link loss is propagated while new links are advertised with at most a 0.5 second delay. Due to the identical underlying route distribution method these results are identical for both the path-based and distance-vector-based



(a) Path-based results



(b) Distance-vector-based results

Figure 6.4: Convergence times of all runs in the evaluation

algorithms.

One of the interesting things to note in this graph, is that the time to convergence after a link restoration (green line) is lower than the initial convergence time (blue line) for larger networks. We expect this is caused by the locality of the change. In larger networks a single link change is not likely to

affect path entries further away in the network, although this does strongly depend on how well connected the network is.

For the path-based protocol, the time it takes for full packet delivery to be restored after a link drop (orange striped line) follows a similar pattern as the link drop convergence time, but takes less time overall (Figure 6.4a). The distance-vector protocol shows behaviour more like the link restore convergence time (Figure 6.4b). This is likely caused by the locality of the destination area, where especially in larger networks the area contains a small number of routers relative to the number of routers in the entire network. The main difference between the packet delivery restoration time in both protocols is that the path-based method seems to perform worse in networks with 15 to 35 nodes, but better in larger networks. This effect is likely caused by the location of the dropped link in relation to the destination nodes. In relatively small networks it is more likely that a destination is close to a dropped link leading to a lower chance that the path-based method has an obvious route around the problem. The distance-vector approach has no knowledge of these paths and simply attempts (an inefficient) alternative path once it knows about the missing link.

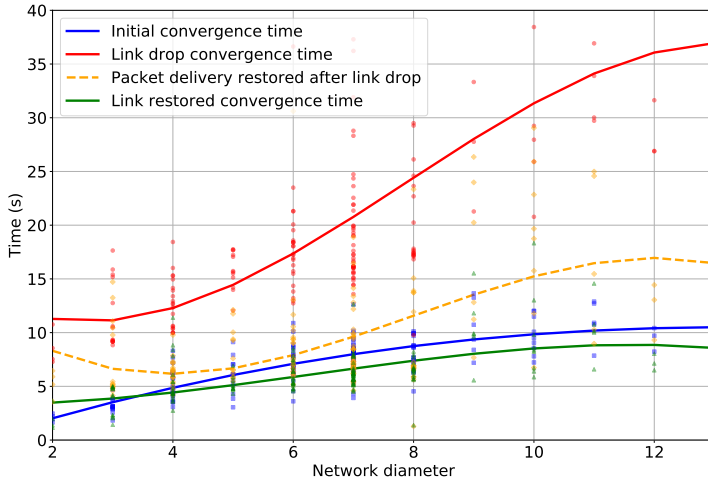
In Figure 6.5 we plot the convergence time against the diameter of the network. The diameter of a network is the length of the longest shortest path in that network. As can be expected there is a strong correlation between this value and the time it takes for the algorithms to converge, as worst case a message needs to travel over the number of links equal to the diameter to reach all routers.

We see that for both path-based (Figure 6.5a) and distance-vector (Figure 6.5b) algorithms there is a strong correlation between network diameter and convergence time, especially for the time it takes to converge following a link drop. The correlation does seem to hold longer for the distance-vector algorithm compared to the path-based algorithm. The convergence time for the path-based algorithm seems to increase less for higher network diameter. This can be explained by the path-based algorithms knowledge of alternate paths, shortening the time needed for the new topology information to become available slightly.

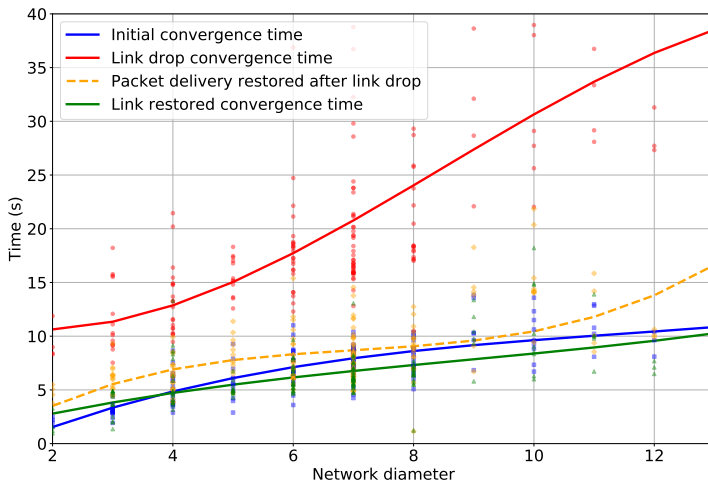
#### 6.4.4 Behaviour During Convergence

As can be expected, during the convergence directly following a link drop, a number of packets is not delivered to the destination routers. We plot these losses for the path-based algorithm in Figure 6.6a and for the distance-vector based results in Figure 6.6b. In these figures, the red dots represent the number of routers that did not receive a packet in a certain simulation run at that time. The intensity of the red colour corresponds to the number of runs in which





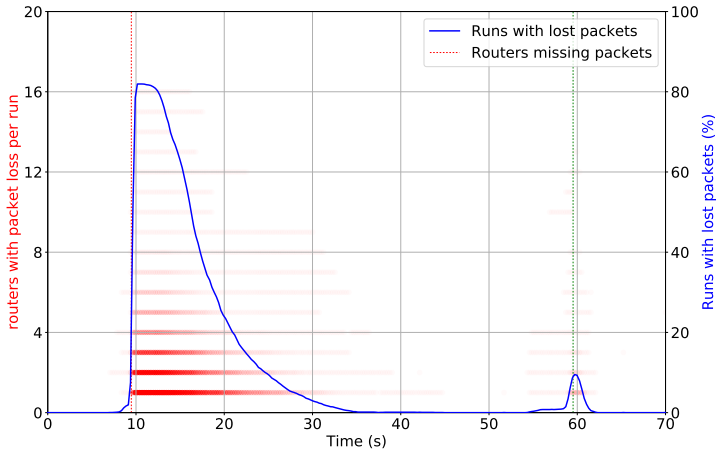
(a) Path-based results



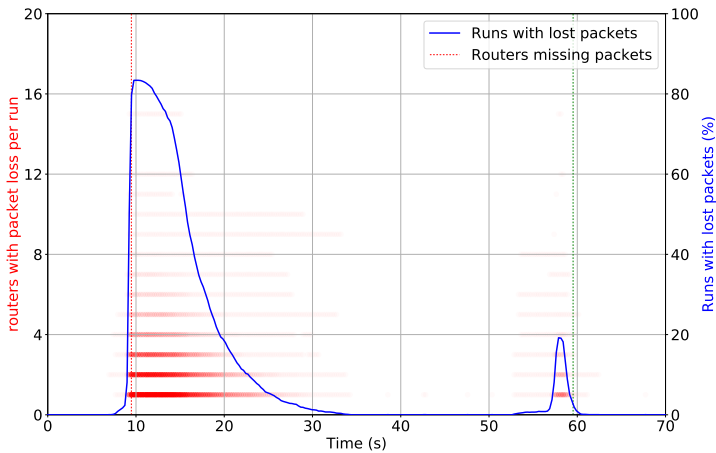
(b) Distance-vector-based results

Figure 6.5: Convergence times of all runs in the evaluation

this number of packets was lost at that particular time. Barely visible dots represent a single occurrence. The blue line represents the percentage of runs in which packet loss occurred at that time. Note that while we plot the red and green dotted line to represent the link drop and restoration point, this does not correspond to the exact drop time in all simulation runs. This can be seen



(a) Path-based results



(b) Distance-vector-based results

Figure 6.6: Packet loss data

by dropped packets that occur earlier than expected. There are two causes for this effect: the varying start-up time in the emulation, mostly depending on network size, and the fact that packets losses are plotted based on the time their packet entered the network. We believe this is the best approach to keep networks of different sizes comparable as this would otherwise have an effect on the time they are lost.

Packet loss mainly occurs after a link drop, we can see that both algorithms

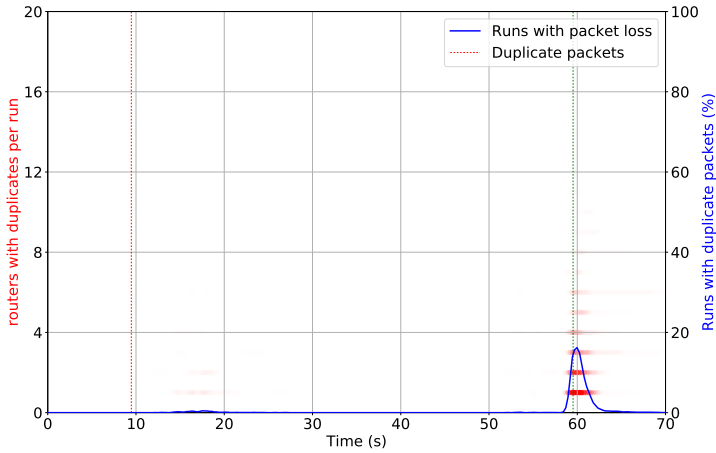
have packet loss in slightly over 80% of the runs following a link drop. This result implies that a small fraction of runs managed to restore from a link loss without losing any packets. This can occur when the link lost is close to a destination and the algorithm manages to converge almost instantly. We can observe that packet loss is resolved slightly faster with the distance-vector algorithm compared to the path-based algorithm, in line with what we could see in Figure 6.4.

Looking specifically at loss caused by convergence after a link restoration we see that this barely leads to any lost packets. The distance-vector-based routing algorithm actually seems to restore packet delivery slightly faster compared to the path-based algorithm, although the packet loss during link restoration for the distance-vector protocol is twice as high compared to the path-based protocol.

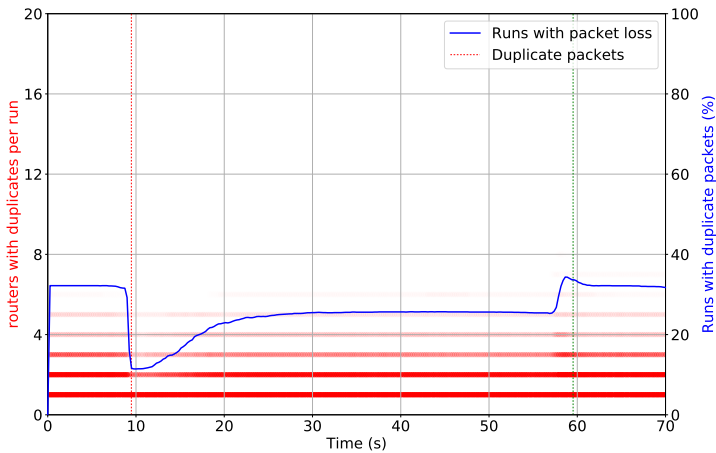
While our routing system should not route packets in a loop it does in some specific conditions route packets to a single router through multiple paths. This effect can occur during convergence when the network is temporarily in a configuration that allows a multi-path situation to occur. This effect can be observed in Figures 6.2a and 6.2b as a solid red line. We can see this line corresponds to the small peak in path cost directly following the restoration of a link. We show the overall duplicate packet delivery rate for both algorithms in Figure 6.7. In these figures the red dots represent the number of routers in a certain run that received a certain packet twice, or more times. As with the loss graphs, the x-axis represents the time at which the packet was sent from the source router measured from the start of a run. The blue line represents the percentage of runs in which a packet was received multiple times by any router. As we can see this effect mainly occurs when a link is restored (or added) to the network.

The path-based approach has a small amount of duplicate deliveries during link restoration, caused by the algorithm converging. These duplicate deliveries seem to occur between 3 and 6 seconds after the link dropped. Link restoration does cause a significant number of duplicate deliveries (present in 17% of the runs). This is caused by the algorithm in different routers temporarily having a different view of the network and in turn their place on the shortest path tree.

The distance-vector approach appears to have duplicate deliveries all the time. These consistent duplicate packets are caused by the algorithm delivering packets over multiple routes to a single destination in some cases. This is the result of the algorithm's limited view of the network making it impossible for the algorithm to accurately determine its place on the shortest path tree, as explained in Sections 5.3.3 and 5.3.4 of Chapter 5. Interestingly the amount of runs with duplicate deliveries is on average lower when a link is missing from the network. This is likely caused by the reduced complexity of the network, leading to less options for the shortest path tree.



(a) Path-based results



(b) Distance-vector-based results

Figure 6.7: Duplicate packet delivery data

Compared to unicast routing we avoid certain issues in our path-based approach, like the count to infinity problem, by relying only on path information. Packets are simply not forwarded any more if a router finds itself on anything but the shortest path from its point of view due to the way the algorithm makes the forwarding decision. On the topic of network change we can conclude that our protocol correctly re-establishes a forwarding tree with only temporary problems, such as a router receiving packets over multiple links, in some cases

following a link restoration.

## 6.5 Open Issues

In this section we will discuss some of the drawbacks our current implementation has and how these should ideally be addressed to make geographic routing feasible on larger networks and between networks.

### Lookup Inefficiencies

Our implementation currently checks for an address match in a large table. While the address comparison itself is very fast, a simple binary AND operation on the destination and coverage entry as described in Chapter 3, the process of checking all entries is not. The current implementation simply loops over all coverage entries to find the interfaces it needs to forward a packet on. This works fine because our tests only include a relatively small number of coverage areas (equal to the number of routers in the network). For a large scale deployment an efficient data structure would need to be developed such that coverage lookups can be significantly faster.

As coverage areas are two-dimensional and can overlap, a simple tree structure to optimize lookups will not work. There are however data-structures optimized for the lookup of two-dimensional structures, such as the  $r^*$  tree [59].

### Route Distribution Improvements

In the current implementation the path and coverage information is combined in a single advertisement. While path and coverage information need to be known on a per router (or network) basis, there is no need to always combine them on the wire.

In a real deployment the coverage area of a router or network would likely not change very often. On the other hand, path information is likely to change much more often and is critical for the system as a whole to function. As a result, the path information should have a short distribution interval while the coverage information should be distributed less often. This will also reduce the network load as the coverage information takes up a significant amount of space.

The most straightforward implementation of this solution would completely decouple path and coverage advertisements. Coverage information can be advertised as (router id, coverage area) tuples, and exchanged between routers on a large time scale. In networks with little change such an advertising interval could be in the range of hours or even days. Path information should still be advertised every few seconds as this is needed to handle topology changes in

the network. Path information can be advertised as simply the collection of best paths known to a router, as the original router id is already included in the path.

If we were to improve the algorithm to deal with variable path costs more change would be needed. Paths would need to include a cost metric for every link in the path, effectively doubling the space requirement for the advertisements and path tables. Such a change would also require changes to the forwarding logic which is out of scope for this thesis.

### **Inter-Domain Routing**

We have only evaluated our solution in single networks as an intra-domain routing protocol. Deploying this system for inter-domain routing could prove challenging due to the complexities of such system, especially with regards to the distinction between internal and external routes.

Routers with a connection to other domains will ideally aggregate internal coverage areas into one or more areas depending on how these areas do or do not overlap. This process will have to be consistent between all externally connected routers in a network.

Aggregating external routes will likely cause problems as they are coupled to a unique router or network id. Aggregation could still be done for internal usage only, not for re-advertisement to other networks. The one exception could be if a network is the only connection point for another network to the greater Internet, but in general this is likely not a good idea.

### **Denial of Service Potential**

Geocast would allow packets to be sent to all devices inside a geographic area. Allowing such broadcasts carries the risk of making denial of service attacks possible. One or more hosts could send a continuous stream of packets to a certain area to overload the router capacity, end-hosts or possibly the wireless medium. This issue is not only applicable to our current implementation, but also to the concept of geocast, or more accurately geographically scoped broadcast in general. While it is not within scope of this thesis to go in depth into Denial of Service prevention, some measures would have to be taken to prevent the abuse of geocast.

To prevent such abuse from occurring the capability to sent geocast messages should be restricted. Sending packets could for example be restricted to certain hosts or certain networks. Ingress routers should be able to filter incoming traffic to prevent unauthorized geocast packets from entering the network. In a system related to vehicular networks, geocast could be restricted to a traffic management entity that is allowed to sent alerts. This would in turn prevent other entities, such as random vehicles, from geocasting messages to

any arbitrary area. In reality restrictions would likely have to be more dynamic, vehicles should be able to transmit certain warnings (such as a collision warning) to neighbouring areas, but not a large region.

## 6.6 Summary & Conclusion

In this chapter we have presented an implementation of a our distance vector and path-based geographic routing algorithms introduced in Chapter 5. We have confirmed the findings of Chapter 5, that both protocols forward packets along a close to optimal shortest path tree in situations where the network is stable. We have also shown that during periods where the network changes both our algorithms can recover in a reasonable time. The time it takes our algorithms to recover depends on factors like network size and the network diameter.

During network convergence due to link failure we lose packets as can be expected, both algorithms do however recover from this state in a reasonable time. Following a link restoration there is only minimal packet loss. The protocol does deliver packets over multiple routes to a destination in this situation. While this effect is not desirable it seems limited to certain network topologies and does to a some degree prevent packet loss from occurring.

Possible improvements for our implementation are mostly related to the information distribution method. In the current implementation coverage area and the path to the covering router are tightly linked. We would like to completely decouple these things to increase scalability. Coverage area information does not need to be advertised as often as path information, and this change could thus decrease overhead. In general our routing protocol, especially with these improvements, can provide another step in enabling Internet-wide geocast in the future.





---

## Infrastructure Assisted Contention-Based Forwarding for Geocast

*In the previous chapters we have designed a geographic routing system. In this chapter we will use this system in a vehicular networking scenario. One standard of vehicular communication is ETSI ITS-G5 GeoNetworking. One of the forwarding methods for geocast in this standard is Contention Based Forwarding (CBF). CBF is dependent on a favourable vehicle distribution to forward messages over multiple hops. A method to extend the effective range of vehicles is to use road side infrastructure to help forward messages. We propose a slightly modified CBF algorithm for road side infrastructure to enable infrastructure assisted forwarding for geocast messages, without modifying the CBF algorithm in the vehicles. We show that with a relatively small modification we can significantly increase delivery rates while also reducing wireless load and delivery delays. The contents of this chapter are based on the work presented in "Infrastructure Support for Contention-Based Forwarding" [60].*

1. Introduction	
2. Background & Related Work	
3. Geographic Addressing	
4 Geographic Forwarding Trees	
5. Geographic Routing Algorithm Design	6. Geographic Routing Implementation and Evaluation
7. Infrastructure Assisted Contention-Based Forwarding for Geocast	
8. Conclusions and Future Work	

## 7.1 Introduction

In the previous chapters we have shown that our geographic routing algorithm for fixed networks can efficiently route packets to their destination. Now that we have functioning geographic routing, we can apply this to solve forwarding problems for vehicular networking.

Without infrastructure vehicles have to rely on close-range communication. The types of communication can range from transferring a vehicle's current speed and information regarding acceleration to a neighbouring vehicle, to warning oncoming traffic of accidents. All these types of communication allow traffic to flow more efficient. This kind of close-ranged data can be transferred through wireless communication to vehicles within range or relayed through a small number of vehicles to others that are relatively near.

Data from farther away can also be useful to increase traffic efficiency, consider information on accidents up the road or local weather warnings delivered to vehicles on specific streets. There are many reasons a geocast packet cannot be delivered to all vehicles in its destination area. Among these reasons are interference, congestion and the main focus of this chapter: the distance between vehicles.

Without existing infrastructure, a message sent to an area needs to be forwarded via multiple hops to reach a destination. Depending on the distance and traffic density there might be gaps between vehicles that are larger than the transmission range of a vehicle. We show such a situation in Figure 7.1, where the first car in the destination area (red rectangle) cannot be reached due to the distance between cars.

Besides this forwarding problem, the current ad-hoc forwarding mechanisms used in ETSI GeoNetworking [28] can also suffer from high end-to-end delays as shown in [61]. Using infrastructure, so called Road Side Units (RSUs), to route messages through a fixed network could help reduce this delay by reducing the amount of wireless steps needed. Ideally this reduction in wireless hops can also help reduce the overall load on the wireless medium. In this chapter we assume these RSUs are connected to the same network, which has a functioning geographic routing protocol such as presented in Chapter 5.

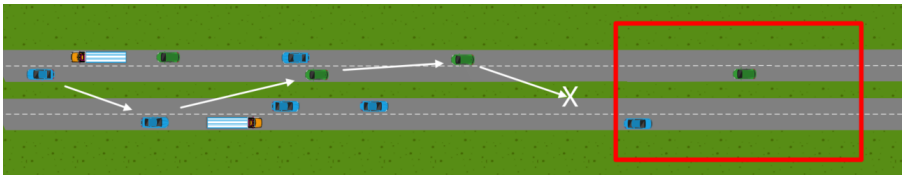


Figure 7.1: Message cannot be delivered due to inter-car distance.

Using infrastructure to help vehicular networks forward geocast messages is not new. However, most past proposals for infrastructure assisted geographic forwarding, such as [62, 63] and [64], make the vehicle an active participant in the routing protocol. We propose to approach the problem from a different angle: We assume an existing VANET protocol (ETSI GeoNetworking [28]), and propose to add infrastructure assisted forwarding without modifying the forwarding algorithm used by the vehicles.

ETSI GeoNetworking [28] defines two possible algorithms for forwarding packets towards a geographical area: The Greedy Forwarding (GF) algorithm and the Contention Based Forwarding (CBF) algorithm. The GF algorithm actively selects the neighbour that is closest to the destination as a forwarding next hop. The CBF algorithm works by simply broadcasting the message and assuming a neighbour closer to the destination will forward it. While this might not seem very efficient there is some redundancy build into the CBF algorithm as we will explain in Section 7.2.

In this chapter we will answer our fourth research question: *How can fixed network geographic routing be applied to the vehicular networking domain?* We do this by presenting an algorithm that modifies an existing ETSI GeoNetworking forwarding method to use RSUs in the forwarding process. We base the requirements for this modification on those presented in Chapter 1, but applied to a vehicular network situation:

- Minimal modification to the existing forwarding algorithm.
- Lower the the number of wireless transmissions needed to deliver a message.
- Faster message delivery compared to the current forwarding algorithm.

We present and evaluate an algorithm that can efficiently forward messages through infrastructure that meets all these requirements defined above. Because not modifying the algorithm in the vehicle is one of our main requirements, we choose to focus on the CBF algorithm as it does not actively select a next hop, but rather passively lets the best next hop forward packets.

The main contribution of this chapter is the design and evaluation of a modification to the CBF algorithm for RSUs. This modification allows RSUs to increase the delivery ratio and reduce the wireless load of the normal CBF method used in ETSI GeoNetworking. Our proposal also significantly reduces the delivery delay of packets.

This chapter is structured as follows: In Section 7.2 we will describe the CBF algorithm used by ETSI GeoNetworking in greater detail. We will present the problem of forwarding gaps using CBF in Section 7.3. In Section 7.4 we present our algorithm that will allow infrastructure to help with the forwarding of CBF packets towards a destination. We evaluate our algorithm in a simulation in

Section 7.5. In Section 7.6 we discuss some open issues regarding our proposal. Finally we summarise and draw conclusion in Section 7.7.

## 7.2 Contention Based Forwarding

CBF is a forwarding technique for ad-hoc networks in which packets are forwarded based on which node transmits first [65]. In the context of ETSI GeoNetworking this is done via time-outs based on the distance between the sender and the receiver [28]. From now on, when we refer to CBF we refer to the specific algorithm used in ETSI GeoNetworking.

The general concept for CBF is that when a vehicle (or other device) sends a message to a destination area, it simply transmits this message and starts a timer to schedule a retransmission. If the original transmitter later overhears another node transmit the packet, it will know the message was forwarded and cancel the timer so as not to rebroadcast the message. When another node receives a message it starts a timer based on the packet's progress. The progress is defined as the distance the packet got closer to the destination area compared to the previous transmitter. The more progress a packet has made, the lower the timer is. When a timer expires and the node has not received a duplicate of the packet it transmits the packet. This system ensures that the receiving node closest to the destination area retransmits a packet, and other nodes do not. Equation 7.1 shows the method used by ETSI GeoNetworking to calculate the time-out  $T$ .

$$T = \begin{cases} T_{max} + \frac{T_{min}-T_{max}}{Dist_{max}} \times Dist & \text{for } Dist \leq Dist_{max} \\ T_{min} & \text{for } Dist > Dist_{max} \end{cases} \quad (7.1)$$

In this equation  $T_{max}$  is the maximum time-out (100 ms),  $T_{min}$  the minimum time-out (1 ms).  $Dist$  is the distance between the transmitter and receiver. The maximum communication distance  $Dist_{max}$  defaults to 1000 meters in the standard [28].

When a node transmits a message it also starts a timer of  $T_{max}$ . When this timer expires and the node has not received the same message from another node, it will rebroadcast the packet. This mechanism prevents single packet losses from leading to an undelivered packet. When all nodes inside a destination area are reached there are almost certainly nodes left with a broadcast timer running. This will lead to at least one, and depending on the relative position of the nodes possibly multiple, broadcasts of the message that are not strictly needed. This can be considered overhead of the protocol.

There is a small difference between CBF inside and outside the destination area. Outside the destination area the timer is based on the progress towards

the destination area, nodes that do not make progress towards the destination area drop the packet. Inside the area the timer is based purely on the distance towards the transmitting node. This ensures a packet is spread throughout the destination area, independent from where it first entered the area.

We show an example of a CBF scenario in Figure 7.2, with the transmission range of the red and blue car represented by the curved lines of the same colour. In this figure vehicle 1 transmits a packet to an area which covers vehicle 5. Vehicle 1 starts a timer for retransmission with the maximum time. The transmission range of vehicle 1 includes vehicles 0, 2 and 3. Vehicle 0 will decide to do nothing because it is not between the sender and the destination area. The other receiving vehicles (2 and 3) now start a timer based on their distance to the sender (vehicle 1). The timer of vehicle 3 will expire first and this vehicle will transmit the packet. This transmission will be received by vehicles 1, 2, 4 and 5. In vehicles 1 and 2 the timer will be cancelled and they will take no further action on this packet. Vehicles 4 and 5 will start their timer when they receive the packet. Vehicle 5's timer will expire first, leading to a transmission which will cancel the timer of vehicle 4. If there are no further vehicles in the destination area, vehicle 5 will do one more transmission when its retransmission timer expires.

Like any ad-hoc forwarding mechanism CBF suffers from delivery problems in situation with a low node count, or low traffic density in the vehicular networking context.

### 7.3 Multi-Hop Transmission Range

To be able to forward a message by V2V communication, vehicles need to be in range of each other. In a situation with a high traffic density, like rush hour, this is almost never a problem. When traffic is less dense there might be gaps in the traffic larger than a vehicle's transmission range. These gaps will prevent

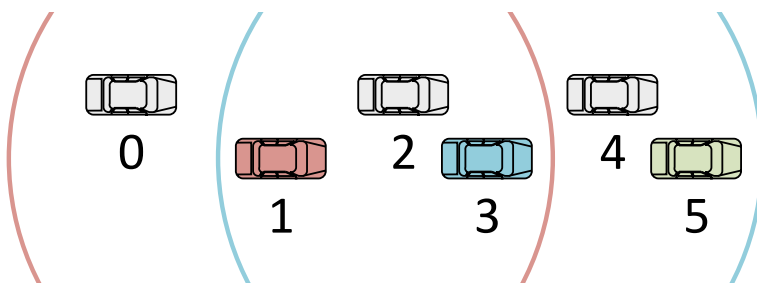


Figure 7.2: Example CBF scenario

hop-by-hop forwarding.

In this section we explore the relation between traffic density and the distance a message can theoretically reach. More specific: we look at the probability of a message reaching a certain distance given a certain traffic density. We simplify traffic to a one-dimensional model where vehicles have a position on a line.

We make several assumptions in this section:

- Vehicles exist as points in a one dimensional system.
- The transmission range is a fixed value, there is nothing influencing this range.
- Traffic is distributed based on a specific inter-vehicle distance.

### 7.3.1 Situational Variables

Several variables have influence on the maximum distance that can be reached using CBF in ETSI GeoNetworking. The most important of these are the traffic density and the transmission range. We will show the impact of both on the theoretical range of vehicles.

The transmission range  $d_{trans}$  is the maximum distance at which another wireless device can receive a transmission. For our simple model we assume that the transmission range of vehicles is a fixed value and doesn't change. We also do not take into account possible packet loss, we assume the channel is perfect.

We base our traffic density values on real world traffic data from Rijkswaterstaat, the Dutch road maintainer. Our data is based on the INWEVA 2017 report [66], specifically the working day hourly data which gives the average number of vehicles on a road segment per hour on an average working day. We use the 'Hengelo-Noord' road segment of the A1 highway in the Netherlands. We chose a highway with medium traffic specifically as traffic densities close to the road capacity will not have any forwarding gaps. We calculate the inter-vehicle distance based on an average speed of 120 km/h. For the evaluation of reachable distances in this section we use the values shown in Table 7.1. In this table we show the average traffic from 6 different hours and the corresponding inter-vehicle distance given a speed of 120 km/h. The different hours are chosen to get results that are applicable to a range of traffic situations, ranging from rush-hour to night time with almost no traffic.

### 7.3.2 Multi-hop Probability

To calculate the distance we can reach using multi-hop communication we first need to know the number of hops we can reach given a transmission range

Time of day	Vehicles/h	Inter-vehicle distance $\mu$ (m)	$\lambda$
08:00 - 09:00	2679	44.79	0.0223
09:00 - 10:00	1765	57.14	0.0147
13:00 - 14:00	2100	67.98	0.0175
20:00 - 21:00	1177	101.95	0.0098
23:00 - 24:00	493	243.41	0.0041
02:00 - 03:00	98	1224.49	0.0008

Table 7.1: Inter-vehicle distances assuming a speed of 120 km/h used in the simulations, based on traffic on the A1 near Hengelo

$d_{trans}$  and a certain inter-vehicle distance  $\mu$ .

As the inter-vehicle distance can be assumed to be given by an exponential distribution, the probability of having at least one vehicle within a certain distance of another vehicle is given by the cdf of the exponential distribution:

$$P(X \leq x) = 1 - e^{-x\lambda} \quad (7.2)$$

In this equation  $x$  is the distance and  $\lambda = \frac{1}{\mu}$ , where  $\mu$  is the inter-arrival distance of vehicles.

To calculate the probability of reaching exactly  $k$  hops, we multiply the probability of  $k$  successive occurrences of a vehicle within  $d_{trans}$  meters with the probability of the next vehicle not being within  $d_{trans}$  meters. This equation is given in Equation 7.3.

$$P(K = k) = (1 - e^{-d_{trans}\lambda})^k e^{-d_{trans}\lambda} \quad (7.3)$$

In this equation  $k$  is the number of hops,  $d_{trans}$  the transmission range in meters and  $\lambda = \frac{1}{\mu}$  with  $\mu$  the the inter-vehicle distance in meters.

The probability to reach at least  $k$  hops is given by Equation 7.4.

$$P(K \geq k) = \sum_k^{\infty} (1 - e^{-d_{trans}\lambda})^k e^{-d_{trans}\lambda} \quad (7.4)$$

In this equation we sum Equation 7.3 onwards from  $k$  to obtain the probability reach at least  $k$  hops.

We plot these last two equations in Figure 7.3. We use a transmission range of 200 meters and  $\lambda$  of 0.0098, which corresponds to the traffic density between 20:00 and 21:00 on the road segment mentioned in Section 7.3.1. We plot Equation 7.3 in Figure 7.3a and Equation 7.4 in Figure 7.3b. We can see that given this  $d_{trans}$  and  $\lambda$  we have around a 50% probability to reach 5 hops.

### 7.3.3 Probability of Reaching a Certain Distance

Now that we know the probability of reaching a specific number of hops given a certain inter-vehicle distance and transmission range we can calculate the probability to reach a certain distance. The probability of reaching a certain distance is logically dependent on the number of hops that will be reached. Each hop can have a maximum distance of  $d_{trans}$  with a minimum of 0 (assuming vehicles can drive next to each other).

An exact expression for the number of hops is known but transforming this to a reachable distance is non-trivial as this depends on the exponentially distributed inter-vehicle distance. In other words, the number of hops and the distance per hop are not independent. Due to the complexity of calculating an exact expression for the probability of reaching a certain distance we have generated a large number of inter-vehicle distances and performed an evaluation on that list.

We assume there is a initial vehicle at the 0 meters position which will act as the transmitter. We generate a list of inter-vehicle distances based on the inter-arrival rate data from INWEVA 2017 [66] as show in Table 7.1. We find the farthest vehicle that can be reached given a certain transmission distance, and repeat this for the found vehicle until no further vehicles can be reached (the distance is greater than  $d_{trans}$ ).

We show the results of 1.000.000 of these simulation runs in Figure 7.4a. Note that the line for a single hop ends abruptly at 200 meters, which was the transmission range we chose. All other lines also end at their respective multiples of the transmission range. We can combine these per hop results into a pdf for these results.

In Figure 7.4b we show the probability of reaching a certain distance in meters with a multi-hop transmission given a transmission range of  $d_{trans} = 200$  and a maximum of 5 hops (the combination of those shown in Figure 7.4a). Note that the probability of reaching any distance within  $d_{trans}$  is equal, but

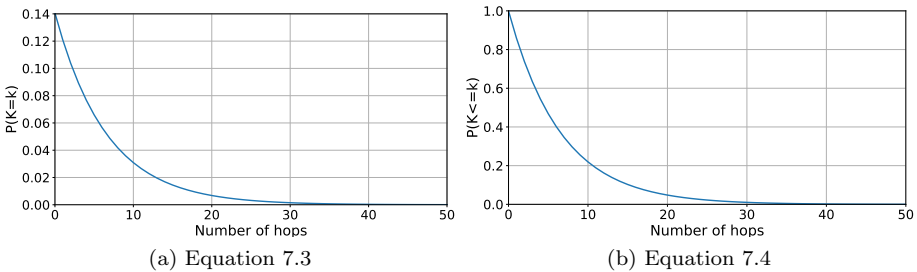


Figure 7.3: Theoretical number of hops given  $d_{trans} = 200m$  and  $\lambda = 0.0098$



this probability drops and steadily declines when passing the initial transmission distance.

We show an overview of the probability to reach a certain distance given three different transmission ranges and an unlimited number of hops using CBF for different traffic densities in Figure 7.5. We refer to specific traffic densities by the hour in which they occur, as given in Table 7.1. Specifically, Figure 7.5b shows this probability given a transmission range of 200 meters, Figure 7.5a given a transmission range of 300 meters, and Figure 7.5c given a transmission range of 400 meters.

From these figures we can see that the probability of reaching distances over 500 meters declines for lower traffic densities. Even with moderate to high traffic densities the probability of reaching distances beyond 1 km are less than 60% given a transmission range of 200 or 300 meters, which can not be considered reliable at all. The situations with low traffic are even worse, we cannot expect a message to be reliably forwarded over more than a few hundred meters in these situations.

With higher transmission ranges (such as 400 meters shown in Figure 7.5c), we see a significantly higher reachability probability. The higher traffic densities show that the probability of being disconnected is very low. We have to take into account that high traffic densities can negatively impact transmission range due to the interference caused by a higher number of devices communicating, as a result these results might not be representative of reality.

We see a linear decrease over the distance from 0 to  $d_{trans}$ . This is caused by the probability of reaching a vehicle within this range not being dependent on the reachability of previous vehicles. After the initial drop in reachability the probability to reach a certain distance decreases exponentially.

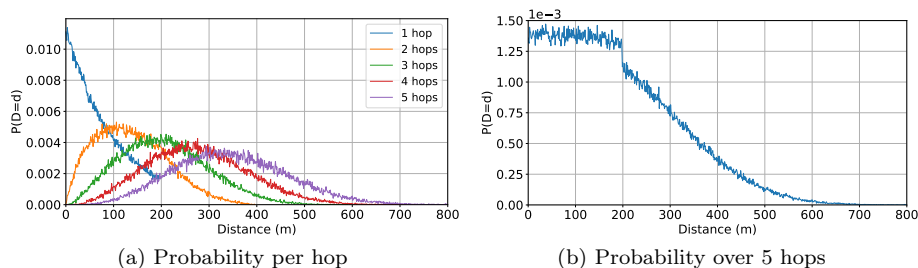


Figure 7.4: Probabilities of reaching an exact distance,  $\lambda = 0.00980$  and  $d_{trans} = 200\text{m}$ .

## 7.4 Infrastructure-Assisted Geographic Broadcast

As we have shown in Section 7.3, geographic broadcasting that relies purely on ad-hoc forwarding between vehicles has reliability issues in less than very dense traffic. We believe this issue can be solved while improving efficiency and increasing reliability in all traffic densities using available infrastructure (RSUs). RSUs can assist in bridging the ‘gaps’ between cars on the road described earlier and decrease overall wireless traffic by routing messages between themselves.

One of the challenges for such a system is preventing the vehicles from rebroadcasting messages with minimal, or even zero, modification of the existing protocols. Another is routing the message between the RSUs so that only RSUs that should receive the message do so. We choose to modify CBF as this allows

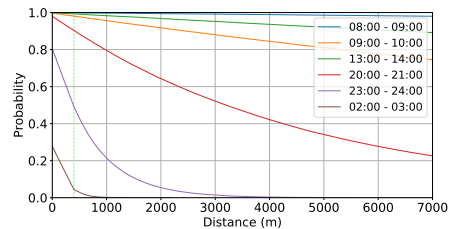
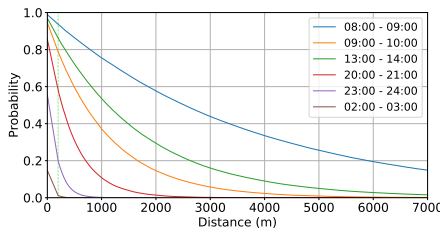
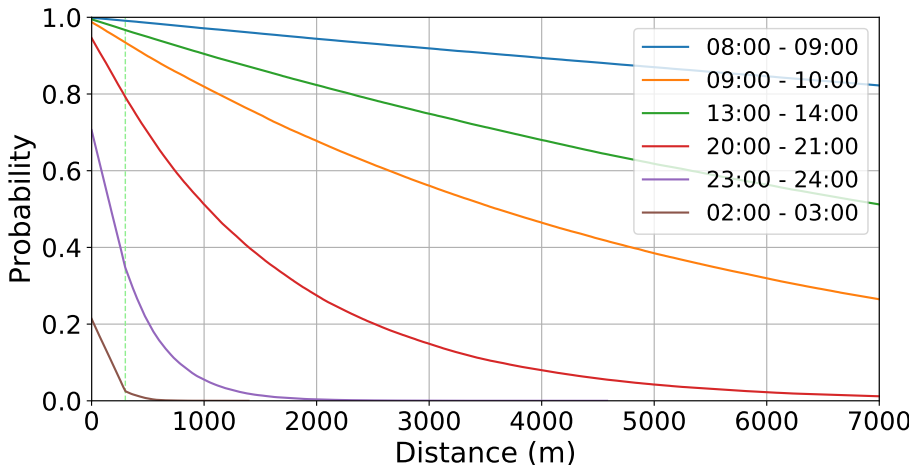


Figure 7.5: Probability to reach a distance per transmission range

us to prevent vehicle-to-vehicle forwarding without modifying the forwarding algorithm within the vehicle. As mentioned before this modification is possible with CBF due to its ‘cancellation’ property. Other available forwarding methods in ITS-G5, such as greedy forwarding, depend on selecting a forwarder by the sending node. These algorithms would require modification in the algorithm for vehicles to enable infrastructure assisted forwarding. With CBF, packets that are received a second time cancel scheduled transmissions of the same packet received earlier. This property allows RSUs to change the forwarding behaviour of vehicles by selectively transmitting packets to cancel these timers.

Our proposal is based on several assumptions: We assume that at least some RSUs are placed along a road. We simplify this road to a one-dimensional line on which vehicles travel. Our RSUs do not participate in the ‘normal’ CBF procedure followed by the vehicles, but instead use our algorithm. All RSUs are connected through a network that supports geocast to distribute messages between them. RSUs are also aware of the coverage of other RSUs, which can be achieved by the routing protocol described in Chapter 6.

### 7.4.1 Proposed Algorithm

Our infrastructure assisted CBF proposal makes no modification to the forwarding logic of the vehicles, they follow the Contention Based Forwarding algorithm as defined in the ETSI Geonetworking standard [28] and briefly explained in Section 7.2. For the most part, RSUs act as normal geo-routers, but they make CBF decisions based on their location and proximity to other RSUs and vehicles. As part of their normal operation RSUs also keep track of all vehicles in their range by listening to periodically sent frames containing, among other things, the vehicles position and speed.

When a vehicle transmits a geocast packet it can reach a RSU in two ways:

- Directly received by a RSU (single hop),
- Forwarded by one or more vehicles before reaching the RSU.

For our algorithm, and by extension the receiving RSU, both situations are identical. In the text we assume the source of a message is a vehicle, but this could also be any other device with geocast functionality.

The simplest case is that of full RSU coverage. In this situation the RSU that receives the initial geocast packet from a vehicle will forward it to all RSUs that cover (parts of) the destination area. These RSUs can then broadcast the packet. If the initial receiving RSU itself is outside the destination area, it will also send the packet to the next RSU in the direction of the destination area. This RSU can then transmit the packet with a remaining hop limit (rhl) of 0 to cancel vehicular contention based forwarding, but it will only do this

if a vehicle is present that could have overheard the initial transmission from the source vehicle. This cancelling step prevents the message from still being forwarded through CBF, which would lead to an increase in wireless channel load.

In the case of incomplete coverage of the destination area the process is more complicated. The RSU that receives the initial broadcast will have to check if the edges of the destination area are covered. If this is the case it will simply transmit the packet to all RSUs that cover the destination area, which in turn will set the correct *rhl*, with a procedure we will explain later, before broadcasting the packet. In case there are no RSUs covering the edges of the destination area, the RSUs closest to those edges outside of the area will also have to transmit the packet. This method ensures that if there is a forwarding path using CBF to the area it will be used. The downside of this method is that the overall load on the wireless channel is higher compared to a single forwarding path to the destination area. In this case the *rhl* will also be calculated according to the procedure explained below.

When no RSUs are present there is no difference compared to normal contention based forwarding as defined in the standard, as we make no changes to the CBF procedure in vehicles. Our algorithm works exclusively on RSUs and only interacts with the CBF functionality of vehicles by sending geocast packets.

### Choosing the Remaining Hop Limit

When there is full coverage by RSUs in the destination geocast area, we can make them broadcast a message with a *rhl* of 0. A receiving vehicle that had a timer for that packet will cancel the timer and thus prevents vehicles from rebroadcasting the message. Vehicles that did not receive the message before will not forward it due to the value of the *rhl*. However, when there are gaps in the RSU coverage, we need vehicles to forward the message to also reach these areas. We ensure this happens by setting the *rhl* to a non-zero value that is based on the distance from the transmitting RSU to the next RSU. We calculate the *rhl* as shown in Equation 7.5:

$$rhl = \left\lceil 2 \times \frac{d_{rsu} - d_{trans}}{d_{trans}} \right\rceil \quad (7.5)$$

In this equation  $d_{rsu}$  is the distance to the next RSU in the direction of the destination area. If the RSU is inside the destination area this is the distance to the furthest RSU.  $d_{trans}$  is the transmission range of a vehicle. In reality this value would be highly dependent on external factors, such as interference. Using this *rhl* the message should always be able to at least reach the edge of the next RSU's coverage area.

This remaining hop limit is based on a worst case scenario, in which vehicles are distributed in such a way that two hops are needed to traverse a distance of  $d_{trans}$  as shown in Figure 7.6. In this situation a vehicle has another vehicle just in front of it and a third vehicle is just out of range of the first, but not the second, and so on. The result of this vehicle distribution is that two transmissions are required to cover a little more than one transmission distance. This rhl calculation ensures that it is always possible to reach all vehicles between two RSUs given a certain transmission range. It is likely best to underestimate the transmission range ( $d_{trans}$ ) so message delivery can be guaranteed in less than ideal situations.

### Initial Receiving RSU Procedure

We show the algorithm for the initial receiving RSU in Algorithm 5. The RSU first checks if the received packet has already been received before. If this is the case the packet is not processed further. If it was not received before we add it to the packet cache and forward the packet to all RSUs that cover the destination area and the first RSU in the direction of the destination area.

---

**Algorithm 5:** RSU receiving geocast packet on wireless interface (V2I)

---

```

Input : Geocast packet gbm
          Source src
          Transmitter src
          Destination area dst
1 Boolean transmit = false;
2 if packet ∈ gbm_cache then
3   | return; // Duplicate packet, drop
4 gbm_cache.add(gbm); // Add packet to cache
5 send_to_RSUs(gbm); // Send the packet to all relevant RSUs
6 if gbm.rhl ≥ 0 then
7   | if !covers(dst) & !closest_rsu(dst) then
8     | transmit = true; // Not covering area and not closest
9   | else if covers(dst) & calcRHL == 0 then // Equation 7.5
10  | | transmit = true; // Covers area and (local) full
    | | coverage
11 if transmit then
12  | sleepRandom(0.001, 0.003); // Wait between 0.001 and 0.003s
13  | gbm.rhl = 0; // Cancel CBF forwarding
14  | Broadcast(gbm);
15 return;

```

---

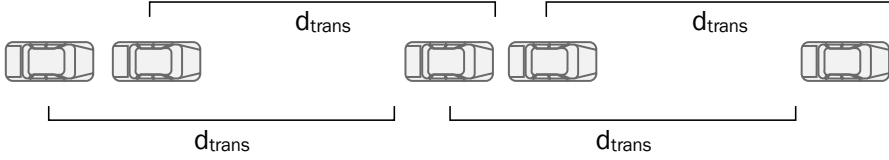


Figure 7.6: Most un-optimal CBF scenario

The next step is to figure out if we need to cancel CBF packet forwarding by vehicles. The RSU checks if the remaining hop limit is larger than 0. If it is we check if the RSU is not in the destination area or the closest RSU to the edge of the destination area (line 7). If this check passes it means we can cancel further CBF of the packet as other RSUs will deliver it. Another reason to cancel CBF is if we are inside the destination area and there is full RSU

---

**Algorithm 6:** RSU receiving packet on fixed interface (RSU-to-RSU)
 

---

```

Input : Geocast packet gbm
          Source src
          Forwarder fwd
          Destination area dst
1 if packet ∈ gbm_cache then
2 | return; // Duplicate packet, drop
3 gbm_cache.add(gbm); // Add packet to cache
4 Boolean transmit = false;
5 if covers(dst) & vehiclesInArea(dst) then
6 | gbm.rhl = calcRHL(); // Equation 7.5
7 | transmit = true;
8 else if !covers(dst) & !insideArea(src, dst) then
9 | if isClosestToEdge(dst) then
10 | | gbm.rhl = calcRHL(); // Equation 7.5
11 | | transmit = true;
12 | else if containsOverhearers(src) then
13 | | gbm.rhl = 0;
14 | | transmit = true;
15 | else if isClosestToSrcRSU(fwd) then
16 | | gbm.rhl = 0;
17 | | transmit = true;
18 if transmit then
19 | Broadcast(gbm);
20 return;

```

---

coverage in our surroundings (lines 9-10).

If one of the previous checks passed the RSU broadcasts the message after a small delay (line 12) with the remaining hop limit set to 0 (line 13) to prevent further vehicular CBF forwarding.

### Fixed Network Receiving RSU Procedure

The procedure followed by an RSU that receives a geocast message on its fixed interface from another RSU is shown in Algorithm 6.

As with the previous algorithm the RSU first checks if the packet was seen before and if it was stops further processing (lines 1-2). If the packet is new it is added to the packet cache (line 3). The RSU then performs its transmitting checks. If the RSU covers the destination area and vehicles are inside this area (line 5) the remaining hop limit is calculated according to Equation 7.5 (line 6).

If the previous check failed and the RSU does not cover the destination area and the original source is also outside the destination area (line 8), we perform some extra checks. There are three conditions under which this RSU will still broadcast the message: i) This RSU is closest to the edge of the destination area (line 9) and will calculate the remaining hop limit if so. ii) This RSU covers vehicles that could have overheard the original source (line 12) and should cancel forwarding with the remaining hop limit set to 0 (line 13). iii) This RSU is the first RSU between the RSU that received the message on its wireless interface and the destination area (line 15). If this is the case the RSU should also set the remaining hop limit to 0 to prevent possible vehicle-to-vehicle forwarding (line 16).

If any of these checks passed the RSU will transmit the message on its wireless interface (lines 18-19).

### Routing

Communication between RSUs is handled by the geographic routing protocol described in Chapter 5, using the addressing scheme from Chapter 3. This system ensures that packets arrive at the RSUs that cover the destination area. For our algorithm to function we require that all RSUs are on the same network and can communicate with each other.

In some cases the RSU that received the packet on its wireless interface should also address other regions. This is needed in the situation where contention based forwarding needs to be cancelled to prevent V2V forwarding.

### Examples

To help illustrate how our algorithm works we will present two examples, shown in Figure 7.7. In these figures we indicate transmissions of vehicles and RSUs

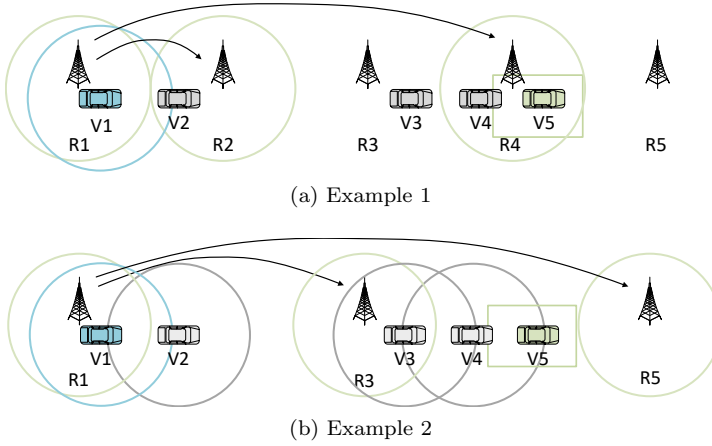


Figure 7.7: Infrastructure assisted CBF examples

by coloured circles around them. These circles also represent the transmission range.

Example 1: We show a full coverage situation with a RSU every 800 meters, assuming a 400 meter transmission range in Figure 7.7a. We have a car (V1), positioned 100 meters east of one of these RSUs (R1), which is the source of the geocast packet. The destination area is 2 km east of the transmitting car (green rectangle). The RSU directly to the west of the car (R1) will receive the geocast message and immediately rebroadcast this message with a rhl of 0. R1 will also transmit the message to all RSUs covering the destination area, and to the RSU directly east of the source car (R2). R2 will broadcast the message with a rhl of 0 to cancel any CBF operations of cars in its reach (V2). The RSUs covering the destination area (R4) will also broadcast the message with rhl of 0 as there is full coverage.

Example 2: This example will use the same scenario as example 1 but reducing the coverage to half (removing every second RSU, as shown in Figure 7.7b). RSUs are now spaced 1600 meters apart. V1 will again transmit a message which is received by R1. The message will be forwarded to R3 and R5, which are the nearest RSUs on both sides of a destination area (green rectangle). These RSUs will broadcast the message so it can be forwarded through normal CBF towards the destination area. V3 will receive the message from R3, and in turn V4 will receive it from V3. The transmission from V4 will be received by V5, the only car in the destination area. In this scenario there is a risk of not being able to cancel CBF forwarding, as we see with V2 to the east of the original source. We still have the benefit of bridging gaps between cars and reduced end-to-end latency for the message compared to pure CBF.



## 7.5 Evaluation

To evaluate our proposal for an RSU assisted geocast system we have evaluated it in a simulation environment. We will first describe the environment used, followed by the different evaluation scenarios used and finally the results.

### 7.5.1 Simulation Tools

Our simulation environment consists of three main tools: OMNeT++, SUMO and Veins. These tools work together to perform the entire simulation.

We used OMNeT++ [43] as our network simulator. OMNeT++ is a discrete event simulator that comes with all the tools to accurately simulate both wired and wireless communication networks. We use it specifically to simulate the communication between vehicles and between vehicles and RSUs.

To get an accurate representation of traffic on a highway we use the SUMO traffic simulator [42]. This program allows us to simulate most traffic situations. We specifically use SUMO to simulate a two lane highway with realistic vehicle arrival rates.

We build our simulation code on the basis provided by Veins [44]. This framework for OMNeT++ provides us with an implementation of IEEE 802.11p [36] with the Wave protocol stack [35] as described in [67] and [68]. We have extended this system by implementing CBF for the vehicles as specified by the ETSI ITS geonetworking standard [28]. We have added RSUs and given them the ability to communicate with each other over a wired network next to their IEEE 802.11p based wireless capabilities. CBF for these RSUs has been implemented as specified in Section 7.4.1.

To connect SUMO with the Veins framework we use TraCI [69]. This protocol and associated set of tools allows us to use SUMO vehicle data in our Veins simulation. It also allows the network simulation to control the simulated vehicles in SUMO but we make no use of this functionality in the simulations of our protocol, we only use the current position of vehicles.

### 7.5.2 Simulation Values

All vehicles in the simulation send a Basic Safety Message (BSM) frame every 0.5 seconds, this is WAVE's equivalent of the ITS G5 Cooperative Awareness Message (CAM). This BSM contains, among other information, the current position and speed of the vehicle. Vehicles and RSUs store (some of) the information in these packets to keep track of surrounding vehicles. The most interesting part of this information for our CBF implementation is the location of vehicles. The RSU based part of our algorithm uses this information to

decide on hop limits and if it will transmit a packet at all. RSUs are connected to each other on a fixed network with negligible delay.

We use the following settings for Veins: We set the transmission power to 20mW and the bit-rate to 6 Mbps in Veins' IEEE 802.11p mac layer. For the physical layer we set receiver sensitivity to -89 dBm and the thermal noise to -110 dBm. Most of these values influence the effective transmission range in the simulator. Using the values mentioned above the maximum transmission range is effectively 400 meters during all our simulation runs. In the remainder of this chapter  $d_{trans}$  can be assumed to be 400 meters when mentioned in relation to the simulation environment. We do not make use of channel hopping in WAVE.

All simulated vehicles have identical properties in the simulation. Vehicles have a maximum acceleration of 2.6 m/s<sup>2</sup> and a maximum deceleration of 4.5 m/s<sup>2</sup>. The maximum speed of the vehicles is limited by the maximum speed of the road (120 km/h), but the vehicles speed is multiplied by a random factor that is normally distributed with a mean of 1 and a standard deviation of 0.1. Vehicles start at the maximum allowed speed and only slow down, and speed up again, in response to other traffic. Due to the randomized speed differences, vehicles can and will overtake each other during the simulation just like real road traffic. All vehicles use SUMO's default car-following model to govern the follow and overtaking behaviour.

### 7.5.3 Evaluation Scenarios

We will evaluate our algorithm in several scenarios with different traffic densities and RSU coverage situations. All our simulations use the same road: An 8 kilometre segment of a two lane highway with a maximum speed of 120 km/h.

We use different traffic densities (given by the inter-arrival rate of vehicles) in different RSU coverage situations. We define our coverage scenarios by the distance between neighbouring RSUs. These distances are based on the transmission distance ( $d_{trans}$ ), which is 400 meters at the maximum in our simulations. We evaluate 4 different coverage scenarios:

- Full RSU coverage (RSUs are spaced  $2 \times d_{trans}$  away from each other),
- Half RSU coverage (RSUs are spaced  $4 \times d_{trans}$  away from each other (essentially removing every second RSU),
- One quarter RSU coverage (RSUs are spaced  $8 \times d_{trans}$  away from each other),
- No RSU coverage (purely contention based forwarding).

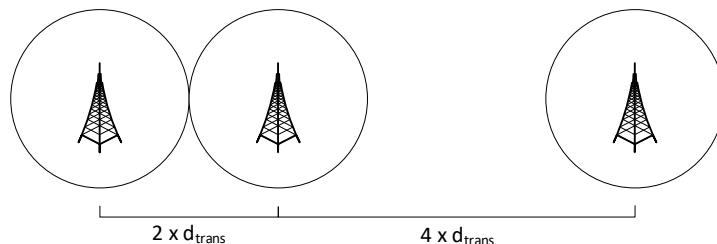


Figure 7.8: RSU distances

We show RSUs with full coverage ( $2 \times d_{trans}$  distance) and with half coverage ( $4 \times d_{trans}$  distance) in Figure 7.8. As mentioned before  $d_{trans}$  is equal to 400 meters in our simulations.

For the full coverage scenario RSUs are positioned at 1200, 2000, 2800, 3600, 4400, 5200, 6000, 6800 and 7600 meters. For half coverage the RSUs at position 1200, 2800, 4400, 6000 and 7600 meters are used. The one quarter coverage scenarios uses only the RSUs positioned at 1200, 4400 and 7600 meters.

In each simulation run we select a single vehicle that will be the initial source of the geocast packet. This transmission is triggered at 402 seconds into the simulation. The position  $p$  of this initial transmitter is selected based on the run id  $r$ :  $p = 1200 + 32 \cdot r$ . The first run ( $r = 0$ ) has the initial source at 1200 meters and the last run ( $r = 49$ ) has it at 2768 meters, for a total of 50 different positions. We select the car closest to this location in the simulation as the initial transmitter.

We use an destination area that is between 200 and 1600 meters long (in steps of 200 meters) and is located 0 to 3200 meters from the initial transmitter (in steps of 200 meters). We start at 1200 meters to give the simulation some time to establish a more realistic traffic pattern, vehicles will have some time to overtake slower vehicles for example.

We perform these tests under different traffic conditions by varying the arrival rate of vehicles in our simulation. We use the inter-arrival rates shown in Table 7.2 for our simulations. These numbers are again based on the INWEVA 2017 working day hourly data [66] as used in Section 7.3.1. In stead of the inter-vehicle distance used before we now use the inter-arrival rate of vehicles. SUMO will use these inter-arrival times to generate the vehicles that are put on our two lane highway. We run 3600 simulation per inter-arrival time and coverage scenario for a total of 43,200 simulation.

All vehicles periodically broadcast CAM messages that include, among other things, their current location and speed. Our RSUs need to know this information, as they make decisions based on the presence of vehicles in their coverage area. For each simulation run we let SUMO run for 400 seconds before

Time of day	Vehicles/h	Inter-arrival rate $\mu$ (s)	$\lambda$
8:00 - 9:00	2679	1.34	0.74
9:00 - 10:00	1765	2.04	0.49
13:00 - 14:00	2100	1.71	0.58
20:00 - 21:00	1177	3.06	0.327

Table 7.2: Inter-arrival rates of vehicles used in the simulations, based on traffic on the A1 near Hengelo

we start the actual simulation. This gives the vehicles generated some time to spread out and fill the simulated road. At 400 seconds we start the network simulator, and we broadcast our geocast message at 402 seconds. This two second gap ensures that RSUs have enough time to receive multiple beacon messages from all vehicles in their range.

### Packet Loss

There are multiple ways in which packets might not be received by vehicles or RSUs. In general there can be two main causes for undelivered packets:

1. Vehicle out of range / no RSU coverage,
2. A collisions not corrected by a retransmission.

The first issue can occur when there is a gap in the forwarding chain that could not be bridged. Another reason can be that there is no RSU coverage in the destination area, and no vehicle-to-vehicle path from another RSU to the destination.

The second problem occurs when two packets are transmitted at the same time. In normal CBF a timer will trigger when the sender does not overhear its message being forwarded, transmitting the message again to increase the probability of delivery. Our RSUs do not rebroadcast messages as this introduces complexities in the forwarding cancelling system.

On roads with multiple lanes the situation might occur that there are two cars forwarding a message if they are driving next to each other. This causes the CBF algorithm to roughly have the same time-out on forwarding. If the timing is right this might cause cars further on the road to receive the packet twice, causing them to cancel their timers and stop message forwarding. In the unlikely event they start transmitting at the same time this will cause a collision.

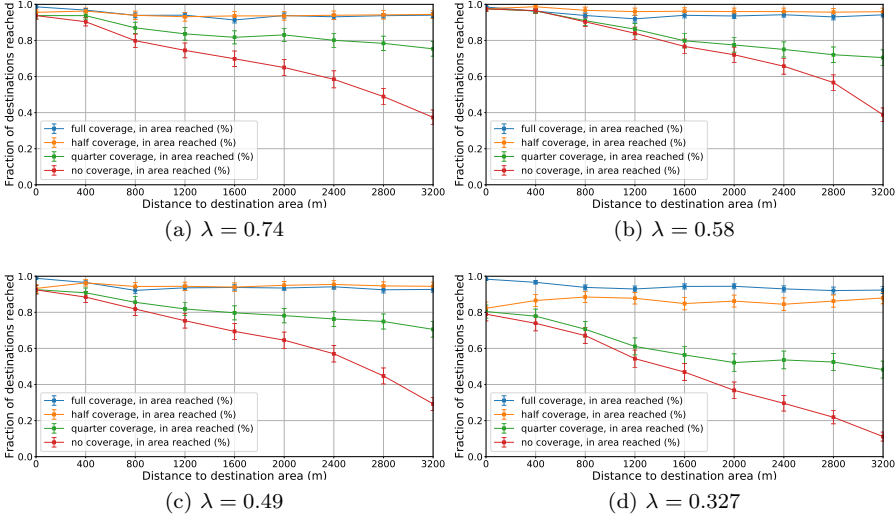


Figure 7.9: Geocast delivery ratios for different distances to the destination area

### 7.5.4 Simulation Results

To evaluate the RSU assisted geocast we will focus on delivery ratios, the number of transmissions needed and the delivery delay. These numbers will give us insight into how effective our solution forwards messages, and the overhead it introduces in terms of wireless transmissions compared to the baseline of normal CBF (no RSUs present). Solutions in which vehicles actively participate in routing packets will always have the benefit of more efficient routing towards an RSU and will also not require transmissions to ‘cancel’ CBF. As such, we choose not to directly compare against such solutions as our proposal will on average be less optimal.

#### Delivery Ratios

For the delivery rate we look at the fraction of nodes present in the destination area that received the geocast message. We take the number of vehicles that have received the geocast message and divide this by the number of vehicles present in the destination area for a simulation run. We ignore runs in which no vehicles were present in the destination area.

We show the delivery ratios of our simulation runs in Figure 7.9 as the fraction of vehicles in the destination area reached. The error-bars represent the

95% confidence interval of the results. These four figures each show the results of a different vehicle inter-arrival rate. We show the full coverage situation (blue line), half coverage (orange line), one quarter coverage (green line) and finally no RSU coverage (red line). The no coverage situation is pure vehicle-to-vehicle CBF.

We see that in all cases the full coverage scenario (solid blue line) allows the geocast message to reach (almost) all vehicles. The loss that does occur is due to collisions with CAM messages. This highlights the main downside of the RSU based transmission, there is no redundancy. In normal CBF operation a retransmission will occur to deliver the message. We chose not to implement this on the RSU to reduce complexity around the ‘cancel forwarding’ packets.

The half coverage scenario (solid orange line) is more reliable in the high traffic density simulation and only less reliable in the lowest traffic density simulations ( $\lambda = 0.327$ ). This is caused by packets being routed to the destination area from two sides in the case the destination area is between or even covered by two RSUs. The cost of this is a significantly higher number of transmissions, 3 times those of the full coverage scenario. The slightly lower delivery fraction at 0 meters to the destination area is caused by there always being at least two transmissions in the full coverage scenario (vehicle and RSU), the scenarios with less coverage might not have these ‘cancel’ transmissions.

We can see that for the one quarter converge situation (green line), the delivery ratio is highly dependent on the distance to the destination area. The delivery rate follows that of no RSUs for the first 1200 meters. This makes sense as that is the (average) distance that has no RSU coverage in this scenario. The further the destination area is from the source the better this coverage situation performs compared to no coverage. One quarter coverage still has a 70% delivery rate at 3200 meters to the destination area in most traffic densities, only dropping to 50% in middle of the night traffic (Figure 7.9d).

In general we see, as was expected, that we have a very high reliability with 100% RSU coverage. The scenario with 50% RSU coverages also has close to 100% delivery rate, only dropping to just under 90% for the least busy traffic situations. For the coverage situation with one quarter of the RSUs we see that delivery becomes less reliable with distance, as can be expected. With no RSU coverage the distance correlation is even higher, resulting in very low delivery ratios at larger distances.

There is another variable in our simulations that has an effect on the delivery ratios: the destination area size. In Figure 7.10 we plot the delivery ratio against the distance to and size of the destination area. We can see that the effect of the area size is (close to) non existent for the full, half and quarter RSU coverage scenario but increases with decreased coverage. Especially the CBF only scenario without RSUs is greatly affected by the size of the area. The reason that the destination area size only affects the no coverage scenario

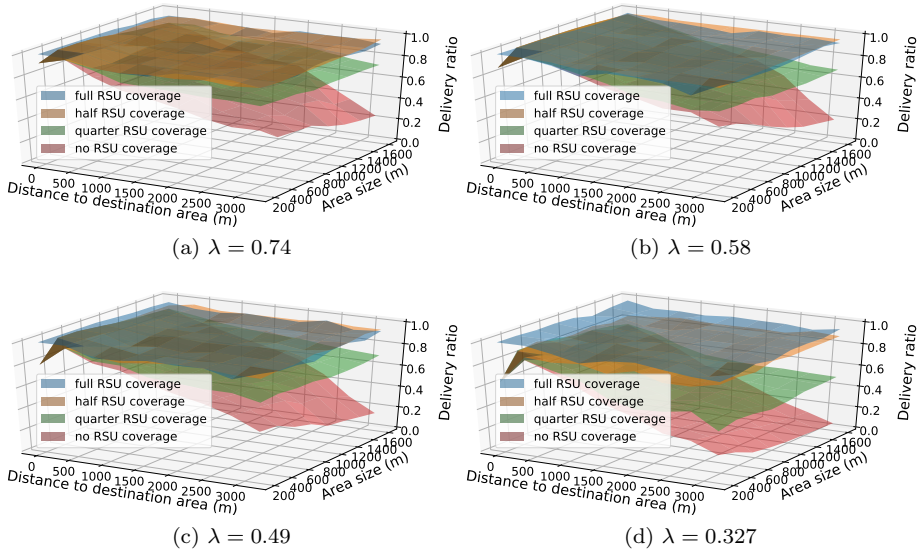


Figure 7.10: Geocast delivery ratios per distance to the destination area and area size

is that it is the only scenario where there is only a single delivery path to all vehicles in the destination area. The scenario with the least coverage (one quarter of the RSUs active) will always have at least two RSUs transmitting the message, leading to at least two CBF paths to the area, resulting in more consistent delivery ratios.

### Number of Transmissions

We count the number of transmissions per simulation run to compare the efficiency of the different coverage scenarios in this regards. This number is simply the total number of geocast transmissions made by vehicles and RSUs in the simulation.

We show these numbers in Figure 7.11, where we plot the number of transmissions against the distance to the destination area. As with the delivery ratios we show the full coverage situations as a blue line, the half coverage situation as a orange line, the one quarters coverage situation as a green line, and the no coverage situation as a red line. The error bars represent the 95% confidence interval of the results.

We can see that the full coverage scenario (blue line) has a consistent number of transmissions in al traffic densities. The number of transmissions is always

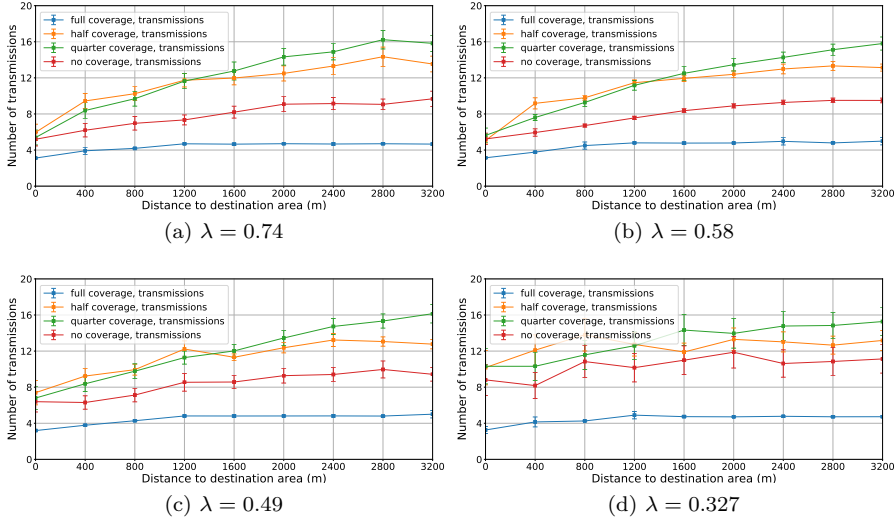


Figure 7.11: Number of transmissions needed to reach destination area

three at 0 distance to the destination area: The initial transmission of the source vehicle, the cancelling transmission of the RSU, and finally another cancelling transmission of the next RSU in line. The total number of transmissions increases with distance until it stabilizes at an average of 5 transmissions. Two of these are caused by the initial transmission and the cancelling transmission of the receiving RSU. The other three are a combination of the one to three RSUs needed to cover the destination area and a possible ‘cancel’ transmission from the next RSU as seen from the initial receiving RSU towards the destination area. This behaviour is consistent over all inter-arrival rates as the presence of vehicles is irrelevant in the full coverage scenario.

The half (orange line) and quarter (green line) coverage scenarios show a similar number of transmissions for all traffic densities. We see an increase of the number of transmissions with distance, caused by three things: i) The message taking multiple hops to reach the initial receiving RSU, as coverage of the initial source is not guaranteed in these scenarios. ii) A later ‘cancel’ message from the next RSU as seen from the initial RSU. iii) Messages being transmitted by RSUs at both sides of the destination area. Note that the delivery rate of the quarter coverage scenario is lower than the half coverage scenario as we have shown in Figure 7.9. We can not conclude that quarter coverage is better than half coverage based on the roughly similar number of transmissions as the delivery rate is lower.



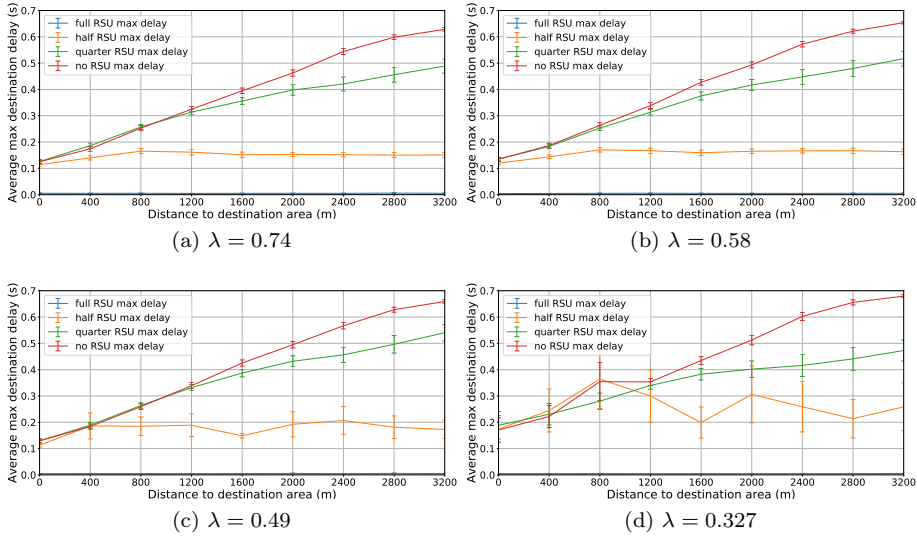


Figure 7.12: Geocast delivery delay over distance

Note that the no coverage scenario (red line) has a lower number of transmission as the half or quarter coverage scenarios. This is a result of the lower delivery ratios of pure CBF as have been shown in Figure 7.9. If a message did not reach the destination area there are also no corresponding transmissions.

Overall full RSU coverage gives the lowest number of transmissions. It is important to note that this number is consistent with the size of the destination area. Had we chosen even larger destination areas, or smaller RSU coverage areas due to a lower transmission range and more RSUs, the number of transmissions would have scaled accordingly.

### Delivery Time

Another important aspect of packet forwarding is the time it takes to deliver a message inside the destination area. Specifically, we measure the time (in seconds) that is needed to reach vehicles inside the destination area. We ignore runs in which no vehicle inside the destination area was reached.

The average delivery times of our simulation runs can be seen in Figure 7.12, where the blue line represents the full RSU coverage scenario, the orange line represents the half coverage scenario, the green line the one quarter coverage scenario and the red line represents no RSUs. The error bars represent the 95% confidence interval of the results. Each sub-figure shows the results of a

different vehicle inter-arrival rate.

We observe that the delivery time using infrastructure assisted geocast with full coverage (blue line) is almost instantaneous. This makes sense as there is only a very low delay following the initial RSU receiving the message. The main factor of the delay is the forwarding-distance-based timer of CBF, which does not come into play in our full coverage scenario.

The half coverage scenario (orange line) is more interesting. The delay is relatively consistent. This effect is due to the situations where the destination area is (partially) between two RSUs. The message is forwarded by vehicles on the path to the last nodes. This vehicle-to-vehicle forwarding path has a worst-case distance of 800 meters, where the only forwarding path is from one RSU up to the edge of the other RSU's coverage area. Note that in the low traffic density simulation (Figure 7.12d) the confidence interval is relatively wide due to the placement of the vehicles.

In the one quarter coverage scenario (green line) we see an increase in delivery time with the distance to the destination area. There does not appear to be a maximum value as with the half coverage scenario, although there is noticeable less increase per distance after 1600 meters. This is caused by the relative positions of RSUs and the initial transmitter in these simulations. The first RSU is relatively near the initial transmitter, the next RSU is 3200 meters further down the road. As the initial transmitter's position moves in the different simulations between the first and second RSU, a distance of 1600 meters is on average just before the second RSU.

The no coverage scenario (red line) has a delivery delay that, as could be expected, scales with the distance. The increase in delay looks linear except near the end of our distance scale. This is caused by packets never reaching the destination area, remember we only measure delivery time inside the destination area. Packets that do reach the destination area likely have a faster path due to a more ideal traffic distribution.

Overall we can conclude that the traffic density has almost no impact on the delivery delay in any of our simulation scenarios. All delivery delays are close to identical with the exception of the lowest traffic density where the size of the confidence interval shows that there is a wider range in delays due to the positions of the relatively small amount of vehicles on the road.

## 7.6 Open Issues

Our current proposal and the related simulations have some shortcomings that would need to be addressed before infrastructure assisted forwarding could be reliably used in practice.

Our algorithm relies on knowledge about coverage areas, especially around

the edges of a destination area. The estimate we currently use based on the radius given a certain transmission range would likely not be accurate enough in reality. We would also need information on the road location, as most roads in the real world are not straight lines.

The evaluation is based on a simulation that makes certain assumptions that are not necessarily true in reality. We constantly assume the transmission range has a fixed value, while in reality this value might be affected by traffic density, height of the antenna, curvature of the road and the presence of buildings or hills.

We also assume that a road is a straight line. As we already noted for the coverage information, this is rarely the case in the real world. While the effect on delivery ratios is likely not that great we should take into account that the simulation does not guarantee the exact same performance in the real world.

All RSUs that participate in the system need knowledge of the coverage area of all other RSUs. This could be provided by a routing system such as we have presented in Chapter 5. Assuming RSU placement is mostly static, coverage information could also be distributed by hand to all RSUs. Both of these options require time from a road maintainer to implement and maintain which could slow down or even prevent adoption.

## 7.7 Summary and Conclusion

In this chapter we have presented an algorithm for infrastructure-assisted geocast in which RSUs help forward geocast messages from vehicles towards a destination area. Our algorithm does not require changes in the vehicle for ETSI ITS geonetworking, it only requires additions to RSUs.

We have shown that this relatively simple algorithm can help increase the probability that messages are delivered. Even when the RSU coverage of a road is only 50%, delivery ratios are only slightly below 90% for a low traffic scenario, and above 90% for denser traffic. The size of a destination area seems to have little influence on the delivery ratio.

We have also shown that the overhead of infrastructure assisted geocast in terms of the number of wireless transmissions done is very low. Full RSU coverage has around half of the transmissions needed by normal CBF. Half RSU coverage has about 50% more transmissions compared to pure CBF but has a much higher delivery ratio.

Infrastructure assisted geocast also reduces the delivery delay of geocast packets. The delay with full RSU coverage is almost fully dependant on the transmission delay between RSUs, and as such is close to 0 seconds. The half coverage delay is mainly determined by the CBF timer delay and is around 0.16 seconds given our simulation values. The benefit compared to pure CBF

is dependant on the distance between the source and the destination area, but even at close range we have smaller delivery delay using the infrastructure.

We can conclude that infrastructure assisted geocast can greatly increase the delivery ratio and reduce delivery delay of geocast packets. The downside is that RSUs and the infrastructure between them are required for this system to function, where pure CBF requires no fixed infrastructure. Due to this trade-off RSUs might only be feasible in certain places where high reliability is required. In these places infrastructure assisted geocast can help reduce wireless traffic and increase reliability.

---

## Conclusions and Future Work

*In this final chapter we will briefly summarize the previous chapters of this thesis and draw conclusions based on the results of our work. We will conclude by discussing possible future work that is relevant for geocast.*

1. Introduction	
2. Background & Related Work	
3. Geographic Addressing	
4 Geographic Forwarding Trees	
5. Geographic Routing Algorithm Design	6. Geographic Routing Implementation and Evaluation
7. Infrastructure Assisted Contention-Based Forwarding for Geocast	
8. Conclusions and Future Work	

## 8.1 Introduction

In this final chapter we will conclude this thesis. We will start by summarising the work we have done in the area of network-layer geocast. We continue by looking back at our research questions and how we have answered them throughout the thesis. Finally, we look towards the future and the work that remains to be done for network-layer geocast support.

## 8.2 Summary

In this thesis we have presented a geocast system that can function both inter- and intra-network. We have described the design of this system from addressing and routing to an actual application. Our proposal addresses four areas related to geocast: i) we have designed an addressing system, ii) we have looked into the most optimal forwarding tree for geocast, iii) we have designed and implemented two geographic forwarding algorithms and iv) we have also developed an application for fixed network geocast in the form of infrastructure assisted forwarding for geocast in VANETs.

### Addressing

In Chapter 3, we have presented an addressing system that can address rectangular areas of different sizes anywhere on the planet. This is done by dividing the world into four rectangles, and each of these rectangles in turn into another four, and so on.

This addressing system also allows combinations of neighbouring rectangles to be aggregated into a single address. This aggregating property allows the system to scale beyond a single network, as adjacent areas can be advertised as a single address.

Our addresses can be represented in binary by simply representing each of the four rectangles on a given level by four bits. Each bit can be set to one if the rectangle is addressed. By setting multiple bits in the same level we can aggregate rectangles together, as described above.

The system also allows relatively fast lookups, overlap of two areas can be found by simply performing a bitwise AND operation on both addresses. If at least a single bit remains on each level that had a bit set on the shortest address there is a match.

Due to only addressing rectangular regions (or combinations of them), this system can not describe any arbitrary region with 100% accuracy. There is always some extra area in the addressed region, compared to the original area. We have evaluated our addressing proposal on this property and shown that on average the target region is around 30% of the addressed region.

## Forwarding Trees

Before we could develop a forwarding algorithm we had to find out which type of forwarding tree is best for geographic routing. In Chapter 4 we have evaluated three different forwarding trees on how efficient they are for geographic routing.

We have compared a shortest path tree, minimum spanning tree and Steiner tree on the number of links used when constructing a tree from a source to a set of destinations. We have compared both geographically clustered and randomly distributed destination sets. We have performed this evaluation on a set of real-world networks.

We have shown that while the Steiner tree has slightly lower link usage compared to a shortest path tree, the difference for geographically clustered destinations is minimal. We conclude that the shortest path tree is likely the best choice for a geographic routing algorithm. This choice also has the benefit of lower computational complexity for forwarding devices, as constructing a Steiner tree is an NP-complete problem.

## Geographic Forwarding

In Chapters 5 and 6 we have described the design, implementation and evaluation of two geographic forwarding algorithms that construct a shortest path tree. The main challenge of these chapters was doing so with limited network knowledge. Without full network knowledge a router can not know its place on the forwarding tree and it is likely to make un-optimal or even redundant forwarding decisions.

In Chapter 5 we have described the design of two geographic forwarding algorithms. One algorithm is distance-vector-based, the other is based on path information. We describe the process we have used and decisions we have made during the design of both algorithms.

We have evaluated both algorithms on their link usage, comparable to the evaluation of Chapter 4. We have shown that, despite only having limited network knowledge, our path-based algorithm has comparable link usage to a shortest path tree. Our distance-vector-based algorithm, which has more limited information, has a higher link usage but can come close to the link usage of a shortest path tree in smaller networks.

We have described an implementation of both algorithms in Chapter 6. In this chapter we have validated the link usage of the implementation against the graph-based evaluation of the previous chapter and we have evaluated the convergence characteristics of both algorithms.

We have evaluated both implementations in emulated networks. This is the same set of networks used in the evaluation of Chapters 4 and 5. We have validated the results of the previous chapter by showing that our implementations

have similar link usage. We have also shown that both algorithms converge relatively quickly, and have limited packet loss while doing so.

### Infrastructure-Assisted Geographic Forwarding

In Chapter 7 we have presented a modification for CBF in vehicular networks that allows fixed infrastructure to assist in the forwarding of geocast packets. Our approach utilises geocast-enabled infrastructure to assist, or even completely replace, ad-hoc based message forwarding.

We have modified the Contention Based Forwarding (CBF) procedure in Road Side Units (RSUs) for ETSI ITS-G5 GeoNetworking by allowing them to selectively prevent mobile node forwarding depending on RSU deployment. We do this without modifying the CBF procedure in the mobile nodes. The RSUs can then forward geocast packets among themselves using the addressing and forwarding methods described in the previous chapters. The RSUs that receive packets on their fixed interfaces can transmit them, effectively resuming ad-hoc CBF nearer to the destination.

We have evaluated our proposal in a simulation environment, where we have tested our system using different traffic densities and different RSU coverage scenarios. In our evaluation we have shown that our method significantly increases packet delivery rates and reduces wireless load given full RSU coverage. In scenarios with limited RSU coverage we have still shown a higher delivery ratio, but at the cost of an overall higher wireless load.

## 8.3 Conclusions

The main research question we have asked at the beginning of this thesis was: *How can an efficient system for network-layer geocast be designed?*. To better answer this question we have divided it into 4 sub-questions, each addressing specific problem areas, which we will now discuss.

Our first research questions was: *How can arbitrary-sized areas be efficiently addressed?* We have shown that the addressing system we have presented in Chapter 3 can address arbitrary areas anywhere on the planet with relatively good accuracy. This system allows addresses to be easily aggregated which allows efficient advertising of areas, which in turn allows the system to be easily scalable. The lookup operation can be also be very efficient, as it is based on a simple bitwise AND operation on two addresses. The addressing system can do this while still fitting within an IPv6 address (using less than 128 bits). This property makes it possible to deploy our system on a wide scale without major modifications to the IPv6 protocol. It should be noted that while our addressing system has many desirable properties, the resulting addresses waste space in that they have a coverage accuracy of on average 30%.



Given a destination area, the resulting address covers the entire destination but 70% of the addressed area is actually outside of the original destination area.

Our second research question was: *Which forwarding tree is the most efficient for geographically scoped destinations?* To answer this question we have compared the shortest path tree, Steiner tree and minimum spanning tree. We have shown that there is little difference in terms of the link cost of a shortest path tree compared to a Steiner tree for geographically clustered destinations. The minimum spanning tree compares very unfavourable on the link cost metric, it is only competitive on very specific networks. When comparing the load distribution over the network the shortest path tree has a distribution similar to the Steiner tree. The minimum spanning tree has some edges that see no load while others have a high load, as can be expected from this type of tree. In general we consider this load concentration an undesirable property, as a single link could be overloaded while there is enough free bandwidth in the network. In terms of computational complexity the minimum spanning tree is best as it can be entirely pre-computed. Constructing the Steiner tree is an NP-complete problem making it an unlikely candidate for a forwarding algorithm. The shortest path tree is the best option overall due to generally good overall link cost and fair link usage distribution.

We have answered our third question "How can packets be efficiently routed towards a geographical area?", in Chapters 5 and 6. We have done this by designing a path-based and distance-vector-based forwarding algorithm that construct shortest path trees. Our path-based forwarding algorithm shows similar link usage as the graph based evaluation of the shortest path tree predicted. The distance-vector based algorithm performs somewhat worse, but the algorithm does require less information and computational resources. We have also made prototype implementations of both algorithms. We have shown that these have the link usage we expect based on the static evaluation of just the algorithms. We have also shown that these implementations converge relatively quickly following link loss in the network.

Our final questions was: *How can fixed network geographic routing be applied to the vehicular networking domain?* We answer this question in Chapter 7 by designing an infrastructure assisted CBF method and evaluating our proposal in a simulator. We have shown that our infrastructure assisted forwarding proposal greatly reduces wireless traffic when there is full RSU coverage. Our proposal also reduces the time it would normally take CBF to deliver the packets due to mostly bypassing the retransmission timer.

In this thesis we have presented three important parts of a network-layer geocast system: addressing, routing and delivery to end-hosts. Together these parts form a complete network-layer geocast system, with which we have answered our main research question. In our system it is possible to send a

geocast message, which will be delivered to another host on the same or even a different network.

## 8.4 Future Work

The solutions for the geocast problem presented in this thesis are not complete. There are many areas that are still open for improvement.

For large scale geocast we will need a more advanced addressing system. Our current addressing proposal has many benefits, such as simple aggregation and prefix matching, but it has the downsides of being restricted to rectangles and addressing a significant amount of addressed area outside of the destination. Further work is needed to look into more area-efficient addressing schemes that would preserve the positive properties of our proposal. Another option might be to have an extra geocast header containing a more exact area description, although this would be at the cost of extra space and complexity.

We have shown a delivery method specifically designed for a single vehicular network system, but have not generalised this to something that would be applicable to all sorts of devices. There are several methods we can imagine that could potentially solve this remaining problem. One straightforward option to deliver geocast messages to end-hosts would be to define geocast specific multicast groups to which devices interested in geocast packets could subscribe. It is also possible to simply broadcast the geocast packets on the local network, although this is likely not a good idea for wireless networks. We could also use unused bits in our geographic addresses to specify a group of receiving devices. Further work is needed to find the most appropriate method to deliver geocast packets to end-hosts, especially in the wireless domain.

As we have noted in Chapter 6, there are open issues regarding the possibility of denial of service attacks. Geocast without restrictions would make it possible for any user on a geocast enabled network to flood areas with traffic. Geocast traffic should likely be restricted to certain sources, or some form of authentication system should be put in place to limit the possibility of flooding. As noted before, solutions to these problems were out of scope for this thesis, but this is something that needs to be addressed before geocast can be widely used in most networks.

While we have designed an efficient forwarding algorithm and comparing two area descriptions can be done efficiently, we are still lacking an efficient way to find all routers covering an area. An efficient data structure is needed to allow routers to quickly search through their coverage information and match areas to the address of a received packet.

If we want to extend our network-layer geocast solution to a truly Internet-wide system, several other improvements will also be needed. We would need

to better evaluate our routing protocols on how well they perform for inter-domain routing and possibly make modifications to better support this. If these remaining challenges can be solved it would be possible to extend geocast to the entire Internet.



---

## Bibliography

- [1] G. Karagiannis, G. Heijenk, A. Festag, A. Petrescu, and A. Chaiken, "Internet-wide geo-networking problem statement," Tech. Rep., 2013. [Online]. Available: <https://tools.ietf.org/html/draft-karagiannis-problem-statement-geonetworking-01>
- [2] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, "A comprehensive survey on vehicular ad hoc network," *Journal of network and computer applications*, vol. 37, pp. 380–392, 2014.
- [3] Nationaal Coördinator Terrorismebestrijding en Veiligheid (NCTV), "Over NL-Alert," accessed 30-8-2019. [Online]. Available: <https://crisis.nl/nl-alert/over-nl-alert/>
- [4] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil, "Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions," *IEEE Communications Surveys Tutorials*, vol. 13, no. 4, pp. 584–616, Fourth 2011.
- [5] J. C. Navas and T. Imielinski, "GeoCast - Geographic Addressing and Routing." in *MOBICOM*, L. Pap, K. Sohrawy, D. B. Johnson, and C. Rose, Eds. ACM, 1997, pp. 66–76.
- [6] A. Festag, R. Baldessari, W. Zhang, L. Le, A. Sarma, and M. Fukukawa, "Car-2-x communication for safety and infotainment in europe," *NEC Technical Journal*, vol. 3, no. 1, pp. 21–26, 2008.
- [7] M. L. Sichitiu and M. Kihl, "Inter-vehicle communication systems: a survey," *IEEE Communications Surveys Tutorials*, vol. 10, no. 2, pp. 88–105, Second 2008.
- [8] T. Imielinski and S. Goel, "DataSpace - Querying and Monitoring Deeply Networked Collections in Physical Space." in *MobiDE*. ACM, 1999, pp. 44–51.
- [9] T. Fioreze and G. J. Heijenk, "Extending DNS to support geocasting towards VANETs: A proposal." in *VNC*. IEEE, 2010, pp. 271–277.

- [10] Y. Ko and N. H. Vaidya, "GeoTORA: A protocol for geocasting in mobile ad hoc networks," in *Network Protocols, 2000. Proceedings. 2000 International Conference on*. IEEE, 2000, pp. 240–250.
- [11] J. Li, J. Jannotti, D. S. De Couto, D. R. Karger, and R. Morris, "A scalable location service for geographic ad hoc routing," in *Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM, 2000, pp. 120–130.
- [12] B. Karp and H.-T. Kung, "Gpsr: Greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM, 2000, pp. 243–254.
- [13] F. Durr and K. Rothermel, "An overlay network for forwarding symbolically addressed geocast messages," in *Proceedings of 15th International Conference on Computer Communications and Networks*. IEEE, 2006, pp. 427–434.
- [14] T. Fioreze and G. J. Heijenk, "Extending the Domain Name System (DNS) to provide geographical addressing towards vehicular ad-hoc networks (VANETs)." in *VNC*, O. Altintas, W. Chen, and G. J. Heijenk, Eds. IEEE, 2011, pp. 70–77.
- [15] R. Baldessari, B. Bödekker, M. Deegener, A. Festag, W. Franz, C. C. Kellum, T. Kosch, A. Kovacs, M. Lenardi, C. Menig *et al.*, "Car-2-car communication consortium manifesto," *DLR Electronic Library* [<http://elib.dlr.de/perl/oai2/>](Germany), vol. 3, no. 4, p. 4, 2007.
- [16] Y. Toor, P. Muhlethaler, A. Laouiti, and A. De La Fortelle, "Vehicle ad hoc networks: Applications and related technical issues," *IEEE communications surveys & tutorials*, vol. 10, no. 3, pp. 74–88, 2008.
- [17] NATIONAL GEOSPATIAL-INTELLIGENCE AGENCY, *World Geodetic System*, 2015 (accessed Januari 14, 2016). [Online]. Available: <https://www.nga.mil/ProductsServices/GeodesyandGeophysics/Pages/WorldGeodeticSystem.aspx>
- [18] T. Hain, "Internet-Draft: An IPv6 Geographic Global Unicast Address Format," Tech. Rep., 2010.
- [19] T. Imielinski and J. C. Navas, "GPS-based geographic addressing, routing, and resource discovery," *Communications of the ACM*, vol. 42, no. 4, pp. 86–92, 1999.
- [20] G. Niemeyer, "Geohash," 2008. [Online]. Available: <http://geohash.org/>

- [21] T. Imielinski and J. Navas, "Gps-based addressing and routing," Internet Requests for Comments, RFC Editor, RFC 2009, November 1996. [Online]. Available: <http://www.rfc-editor.org/rfc/rfc2009.txt>
- [22] J. C. Navas and T. Imielinski, "On reducing the computational cost of Geographic Routing," *Rutgers University, Department of Computer Science, Tech. Rep. DCS-TR-408*, 2000.
- [23] D. Moscoviter, M. Gholibeigi, B. Meijerink, R. Kooijman, P. Krijger, and G. Heijenk, "Improving spatial indexing and searching for location-based DNS queries," in *International Conference on Wired/Wireless Internet Communication*. Springer, 2016, pp. 187–198.
- [24] C. Maihofer, "A survey of geocast routing protocols," *IEEE Communications Surveys Tutorials*, vol. 6, no. 2, pp. 32–42, Second 2004.
- [25] M. Di Felice, L. Bedogni, and L. Bononi, "Group communication on highways: An evaluation study of geocast protocols and applications," *Ad Hoc Networks*, vol. 11, no. 3, pp. 818–832, 2013.
- [26] B. T. Sharef, R. A. Alsaqour, and M. Ismail, "Vehicular communication ad hoc routing protocols: A survey," *Journal of network and computer applications*, vol. 40, pp. 363–396, 2014.
- [27] J. Liu, J. Wan, Q. Wang, P. Deng, K. Zhou, and Y. Qiao, "A survey on position-based routing for vehicular ad hoc networks," *Telecommunication Systems*, vol. 62, no. 1, pp. 15–30, 2016.
- [28] ETSI, "ETSI EN 302 636-4-1 - V 1.3.1: Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1: Media-Independent Functionality," ETSI, Tech. Rep., 2017.
- [29] J. Moy, "Ospf version 2," Internet Requests for Comments, RFC Editor, STD 54, April 1998, <http://www.rfc-editor.org/rfc/rfc2328.txt>. [Online]. Available: <http://www.rfc-editor.org/rfc/rfc2328.txt>
- [30] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," Internet Requests for Comments, RFC Editor, RFC 4271, January 2006, <http://www.rfc-editor.org/rfc/rfc4271.txt>. [Online]. Available: <http://www.rfc-editor.org/rfc/rfc4271.txt>
- [31] IANA. (2014) IPv4 Multicast Address Space Registry. [Online]. Available: <http://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml>

- [32] ——. (2014) IPv6 Multicast Address Space Registry. [Online]. Available: <http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml>
- [33] B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, R. Parekh, Z. Zhang, and L. Zheng, “Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised),” Internet Requests for Comments, RFC Editor, STD 83, March 2016.
- [34] A. Adams, J. Nicholas, and W. Siadak, “Protocol independent multicast - dense mode (pim-dm): Protocol specification (revised),” Internet Requests for Comments, RFC Editor, RFC 3973, January 2005.
- [35] “IEEE Guide for Wireless Access in Vehicular Environments (WAVE) - Architecture,” *IEEE Std 1609.0-2013*, pp. 1–78, Mar. 2013.
- [36] IEEE 802.11 Working Group and others, “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 Amendment 6: Wireless Access in Vehicular Environments,” *IEEE Std*, vol. 802, no. 11, 2010.
- [37] V. Reding, “On the harmonised use of radio spectrum in the 5 875-5 905 MHz frequency band for safety-related applications of Intelligent Transport Systems (ITS),” *Official Journal of the European Union*, vol. L220, pp. 24–26, 08 2008.
- [38] A. Festag, “Cooperative intelligent transport systems standards in europe,” *IEEE Communications Magazine*, vol. 52, no. 12, pp. 166–172, December 2014.
- [39] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, “The Internet Topology Zoo,” *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1765–1775, october 2011.
- [40] A. A. Hagberg, D. A. Schult, and P. J. Swart, “Exploring network structure, dynamics, and function using NetworkX,” in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Pasadena, CA USA, Aug. 2008, pp. 11–15.
- [41] Mininet Project, “Mininet.” [Online]. Available: <http://mininet.org>
- [42] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, “Microscopic Traffic Simulation using SUMO,” in *The 21st IEEE*



- International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: <https://elib.dlr.de/124092/>
- [43] A. Varga and R. Hornig, “An overview of the OMNeT++ simulation environment,” in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, p. 60.
- [44] C. Sommer, R. German, and F. Dressler, “Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis,” *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, January 2011.
- [45] B. Meijerink, M. Baratchi, and G. Heijenk, “An Efficient Geographical Addressing Scheme for the Internet,” in *International Conference on Wired/Wireless Internet Communication*. Springer, 2016, pp. 78–90.
- [46] P. Coschurba, K. Rothermel, and F. Dürr, “A fine-grained addressing concept for geocast,” in *International Conference on Architecture of Computing Systems*. Springer, 2002, pp. 101–113.
- [47] J. Postel, “Internet Protocol,” Internet Requests for Comments, RFC Editor, STD 5, September 1981, <http://www.rfc-editor.org/rfc/rfc791.txt>. [Online]. Available: <http://www.rfc-editor.org/rfc/rfc791.txt>
- [48] S. E. Deering and R. M. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” Internet Requests for Comments, RFC Editor, RFC 2460, December 1998, <http://www.rfc-editor.org/rfc/rfc2460.txt>. [Online]. Available: <http://www.rfc-editor.org/rfc/rfc2460.txt>
- [49] B. Meijerink, M. Baratchi, and G. Heijenk, “Evaluation of geocast routing trees on random and actual networks,” in *International Conference on Wired/Wireless Internet Communication*. Springer, 2017, pp. 127–142.
- [50] M. Doar and I. Leslie, “How bad is naive multicast routing?” in *INFOCOM’93. Proceedings. Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking: Foundation for the Future, IEEE*. IEEE, 1993, pp. 82–89.
- [51] H. F. Salama, D. S. Reeves, and Y. Viniotis, “Evaluation of multicast routing algorithms for real-time communication on high-speed networks,” *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 332–345, 1997.
- [52] J. C.-I. Chuang and M. A. Sirbu, “Pricing multicast communication: A cost-based approach,” *Telecommunication Systems*, vol. 17, no. 3, pp. 281–297, 2001.

- [53] U. T. Nguyen and J. Xu, "Multicast routing in wireless mesh networks: Minimum cost trees or shortest path trees?" *IEEE Communications Magazine*, vol. 45, no. 11, pp. 72–77, 2007.
- [54] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta informatica*, vol. 15, no. 2, pp. 141–145, 1981.
- [55] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.
- [56] B. Meijerink, M. Baratchi, and G. Heijenk, "Design & analysis of a distributed routing algorithm towards Internet-wide geocast," *Computer communications*, vol. 146, pp. 201–218, 2019.
- [57] B. Meijerink and G. Heijenk, "Implementation and Evaluation of Distributed Geographical Routing," in *International Conference on Wired/Wireless Internet Communication*. Springer, 2018, pp. 121–133.
- [58] Linux Foundation. (2009) netem. [Online]. Available: <http://www.linuxfoundation.org/collaborate/workgroups/networking/netem>
- [59] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R\*-tree: an efficient and robust access method for points and rectangles," in *Acm Sigmod Record*, vol. 19, no. 2. Acm, 1990, pp. 322–331.
- [60] B. Meijerink and G. Heijenk, "Infrastructure Support for Contention-Based Forwarding," in *Vehicular Networking Conference*. Accepted for publication, 2019.
- [61] S. Kuhlorgen, I. Llatser, A. Festag, and G. Fettweis, "Performance Evaluation of ETSI GeoNetworking for Vehicular Ad Hoc Networks," in *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, May 2015, pp. 1–6.
- [62] P. Li, T. Zhang, C. Huang, X. Chen, and B. Fu, "RSU-Assisted Geocast in Vehicular Ad Hoc Networks," *IEEE Wireless Communications*, vol. 24, no. 1, pp. 53–59, February 2017.
- [63] D. Borsetti and J. Gozalvez, "Infrastructure-assisted geo-routing for cooperative vehicular networks," in *2010 IEEE Vehicular Networking Conference*, Dec 2010, pp. 255–262.
- [64] Y. Wu, Y. Zhu, and B. Li, "Infrastructure-assisted routing in vehicular networks," in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 1485–1493.

- [65] H. Füßler, J. Widmer, M. Käsemann, M. Mauve, and H. Hartenstein, “Contention-based forwarding for mobile ad hoc networks,” *Ad Hoc Networks*, vol. 1, no. 4, pp. 351–369, 2003.
- [66] “Verkeersintensiteiten INWEVA werkdag 2017,” Rijkswaterstaat, Tech. Rep., 2018.
- [67] D. Eckhoff, C. Sommer, and F. Dressler, “On the Necessity of Accurate IEEE 802.11p Models for IVC Protocol Simulation,” in *75th IEEE Vehicular Technology Conference (VTC2012-Spring)*. Yokohama, Japan: IEEE, 5 2012, pp. 1–5.
- [68] D. Eckhoff and C. Sommer, “A Multi-Channel IEEE 1609.4 and 802.11p EDCA Model for the Veins Framework,” in *5th ACM/ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2012), 5th ACM/ICST International Workshop on OMNeT++ (OMNeT++ 2012), Poster Session*. Desenzano del Garda, Italy: ACM, 3 2012.
- [69] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, “TraCI: An Interface for Coupling Road Traffic and Network Simulators,” in *Proceedings of the 11th Communications and Networking Simulation Symposium*, ser. CNS ’08. New York, NY, USA: ACM, 2008, pp. 155–163. [Online]. Available: <http://doi.acm.org/10.1145/1400713.1400740>



---

## List of Acronyms

<b>AS</b>	Autonomous System
<b>BSM</b>	Basic Safety Message
<b>BGP</b>	Border Gateway Protocol
<b>CAM</b>	Cooperative Awareness Message
<b>CBF</b>	Contention Based Forwarding
<b>DNS</b>	Domain Name System
<b>DV</b>	Distance Vector
<b>ETSI</b>	European Telecommunications Standards Institute
<b>GPS</b>	Global Positioning System
<b>IETF</b>	Internet Engineering Task Force
<b>I2V</b>	Infrastructure-to-Vehicle
<b>IP</b>	Internet Protocol
<b>IPv4</b>	Internet Protocol version 4
<b>IPv6</b>	Internet Protocol version 6
<b>ISP</b>	Internet Service Provider
<b>ITS</b>	Intelligent Transport Systems
<b>OSPF</b>	Open Shortest Path First
<b>PIM</b>	Protocol Independent Multicast
<b>PIM-DM</b>	PIM Dense Mode
<b>PIM-SM</b>	PIM Sparse Mode
<b>RA</b>	Route Advertisement

**rhl** remaining hop limit

**RP** Rendezvous Point

**RSU** Road Side Unit

**V2V** Vehicle-to-Vehicle

**V2I** Vehicle-to-Infrastructure

**V2X** Vehicle-to-everything

**VANET** Vehicular Ad-hoc Network

**WAVE** wireless Access for Vehicular Environment

---

## Publications by the Author

Meijerink, B. & Heijenk, G. “Lossless Multicast Handovers in Proxy Fast Mobile IPv6 Networks” in *International Conference on Wired/Wireless Internet Communication*, 2015, pp 341-354

Meijerink, B.; Baratchi, M. & Heijenk, G. “An Efficient Geographical Addressing Scheme for the Internet” in *International Conference on Wired/Wireless Internet Communication*, 2016, pp 78-90

Meijerink, B.; Baratchi, M. & Heijenk, G. “Evaluation of geocast routing trees on random and actual networks” in *International Conference on Wired/Wireless Internet Communication*, 2017, pp 127-142

Meijerink, B. & Heijenk, G. “Implementation and Evaluation of Distributed Geographical Routing” in *International Conference on Wired/Wireless Internet Communication*, 2018, pp 121-133

Meijerink, B.; Baratchi, M. & Heijenk, G. “Design & analysis of a distributed routing algorithm towards Internet-wide geocast” in *Computer communications, Elsevier*, 2019, 146, pp 201-218

Meijerink, B. & Heijenk, G. “Infrastructure Support for Contention-Based Forwarding” in *Vehicular Networking Conference*, 2019, Accepted for publication





---

## Open Data Management

Almost all the results presented in this thesis are based on data generated by simulation data or the results of running implementations in emulated network environments. We provide the code and the instructions to run this software online: <https://berndmeijerink.nl/thesis/software>. Based on this software and the variables presented throughout this thesis a user should be able to generate identical results. The reader can find the software used in the evaluation of each chapter by following these links:

Chapter 3: <https://berndmeijerink.nl/thesis/software/addressing>

Chapter 4: <https://berndmeijerink.nl/thesis/software/forwarding-trees>

Chapter 5: <https://berndmeijerink.nl/thesis/software/routing-protocol>

Chapter 6: <https://berndmeijerink.nl/thesis/software/routing-software>

Chapter 7: <https://berndmeijerink.nl/thesis/software/rsu-cbf>

Each page includes instructions on how to run the software and information on the software's dependencies.