Routing protocols for Android based opportunistic networks

Bachelor Thesis

by

Jannik Leßenich

born in

Bergheim

submitted to

Technology of Social Networks Lab

Jun.-Prof. Dr.-Ing. Kalman Graffi

Heinrich-Heine-Universität Düsseldorf

June 2016

Supervisor:

Andre Ippisch, M. Sc.

Abstract

With the vast amount of new technology, developed in the last decades, we now face a technological society. With the possibilities of more and stronger computing devices, the usage of such increases every day. The Internet brought the opportunity of global informational exchange, but with national controls, strong commercialism and fluctuating costs of needed infrastructure, we may need substitutes in certain areas. Opportunistic Networks might be one of them.

To achieve a reliable network, a good routing protocol is necessary. Therefore, fundamentals were clarified and related work considered and rated. Short looks at TTL for OppNets and impact of limited resources were taken and a context aware, quota-based protocol was identified as a promising approach, which led to a proposal of an own protocol called Neighbourhood Aggregating Social and Geographic Label Protocol.

This protocol features functions to aggregate nearby nodes to clusters, to find paths to destinations and propagate the presence of nodes to others, while using social and geographical information in form of labels in a scoring function to calculate similarity to the destination and identify good forwarders, by propagating the own user information through information packages.

The protocol was implemented in a given Android network application, using Tethering Hotspots to connect devices and creating an OppNet. The protocol implementation was evaluated and it was shown, that the implementation is able to spare resources, while having a good delivery quota. Despite this, it was clarified, that the implementation, the helping function for networking and the network application itself still need improvements and that further tests are needed for the calibration of routing parameters.

Acknowledgments

Thanks to all my friends and family for the fun I have with you.

A special thank you goes to my girlfriend for cheering me up and another to my Mom for correcting my grammar mistakes.

I hereby express my gratitude to the creators and developers of Inkscape and LaTeX and all other developers of non commercial or open source projects.

At last, a big thank you to my supervisor A. Ippisch for the interesting thesis topic, the support he gave me and the fun we had at our meetings.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

1.1 Motivation

With the vast amount of new technology, developed in the last decades, we now face a technological society. With the possibilities of more and stronger computing devices, the usage of such increases every day. The Internet brought the opportunity of global informational exchange, but with national controls, strong commercialism and fluctuating costs of needed infrastructure, we may need substitutes in certain areas. Opportunistic Networks might be one of them. This thesis will focus on routing in OppNets, since routing is one of the essential components of a network.

Opportunism derives from the Latin word opportunus, which can mean something like suitable, fit, convenient or favorable and could be understood as the act of (selfishly) taking opportunities at the right time. This describes the necessary procedure in the following defined networks.

1.2 Opportunistic Networks

Opportunistic Networks, short OppNets, are disorganized networks with, not necessarily, permanent active nodes. These nodes use spontaneous connections to transmit data.

Because of their unstable nature, OppNets are in contrast to the hierarchic ordered Internet. The most common imaginable use case for OppNets are those with mobile nodes and wireless connections, which therefore are a special case of the mobile ad-hoc networks (MANETs). In a true OppNet there is no guarantee of a constant or continuous connection between devices, thus this network should be a delay tolerant network (DTN) and could also be considered as a disruption tolerant network. It should generally work as a package switching network. Due to the unknown topology of the network in realistic scenarios, without centralized instances, a circuit switching approach would make no sense. Lest the quota of successful delivered packages suffers from the constantly changing topology, the store and forward technique should be used.

Store and forward is a technique which nodes use, when they currently cannot forward a package. For reasons like the lack of routes or net congestion, they decide to keep it for a certain or unlimited time to wait for a suitable opportunity to transmit the stored package. This standard process, used by routers, gains even more importance in OppNets, since the transmission of a package could easily take days, depending on scenarios.

In further course, mainly decentralized opportunistic routing with mobile nodes will be discussed, as a centralization represents an unwanted dependence.

1.3 Basic Opportunities and Problems of Opportunistic Networks

The development of advanced mobile devices in the last decade, gives us the opportunity to create a giant OppNet. In 2019 there will most likely be up to 6.1 billion smartphone subscriptions [eri15], continuously running, and in addition to that, there are Tablets, Laptops and other devices, which also can be used as a node.

On the one hand the special requirements of an OppNet can cause problems:

Since we are handling a delay tolerant network with store and forward techniques, every node needs a certain size of buffer, depending on network load and usage, to ensure a

properly working network.

Furthermore, a fast and successful communication depends on the amount, movement and placement of nodes in an OppNet. In a scenario with two nodes gathering points X and Y, in which a node A of point X wants to communicate with a node B of point Y, either a shuttle node, carrying packages from X to Y, needs to exist or another gathering point in between to connect them. But even if they exist, for the fastest communication possible, A and B need a stable and static connection, but this cannot be guaranteed in an OppNet. The dynamic topology in an OppNet forces the nodes to a dynamic routing approach and buffering of packages to ensure a high delivery probability. Since transmission speed, deliverability, delivery probability and needed resources are depending on the environment of the OppNet, it is necessary, that a routing protocol is highly adaptive to this special scenario. Every OppNet needs a special amount of nodes with a certain movement and placement to be efficient.

Another point is the additional burden for the used devices. First of all, mobile devices depend on their accumulator, therefore a high energy consumption to create an OppNet is undesirable. The energy consumption should always stay low, so that the user experience with the OppNet stays positive. Another burden is, that extra buffer space is needed. Moreover the nodes are all private used devices, thus the communication needs extensive security against various attack forms.

On the other hand OppNets offer a variety of interesting advantages:

First of all, an OppNet offers independence of infrastructure. Despite the influence of catastrophes, national actions, such as shutting down central instances of the Internet, or simply electricity issues, an OppNet stays operable. People could use this to bypass national or other controls, like censorship or shutting down file sharing services.

Furthermore, in areas with a lack of infrastructure, an OppNet can provide an alternative network, as long as there are suitable mobile devices. It also can relieve existing infrastructure from all non time critical traffic. Local P2P networks can easily be established, even large or worldwide P2P networks are possible, with the right amount of nodes and the right routing algorithm. Moreover use cases like non satellite animal tracking exist.

1.4 Outline

In this Chapter problems and opportunities of OppNets were discussed.

In Chapter 2 the basics and taxonomy of opportunistic routing, the requirements of an opportunistic routing protocol and the impact of limited resources on those, will be discussed.

In Chapter 3 the special case of OppNets on Android devices in comparison to the wanted case will be explained in Section 3.1. Further routing related ideas will be discussed and a routing protocol will be proposed in Section 3.4.

Chapter 4 contains the explanation for the implementation of the proposed protocol in Section 4.2. Additional implementations and functionalities will be named and shown in Section 4.3.

In Chapter 5 test scenarios and the results of these tests on the implemented routing protocol will be described in the Sections 5.1 and 5.2, and the implementation discussed in Section 5.4.

Chapter 6 is the conclusion of this thesis. Possible Future Work will be proposed.

Chapter 2

Fundamentals

In this Chapter we will discuss the basics of OppNets. We will clarify, what we demand of a routing protocol in OppNets and how we approach to achieve such a protocol in Section 2.1. General ideas of opportunistic routing are presented and discussed in Section 2.3 and we take a short look at taxonomy, at scenarios and the problems of limited resources in Sections 2.2, 2.4 and 2.5. In Section 2.6 we name the most promising routing approach.

2.1 Optimal Routing Protocols

There are three basic conditions, that a routing protocol or other protocols from different network layers have to fulfill to be considered as optimal:

- A high quota of successful communication
- Fast transmission
- Resource consideration

A routing protocol can be considered as optimal, if all of the following applies: Approximately 100% of packages are routed correctly. It ensures a maximum of successful

package delivery and the decision to route a package is made immediately, so that occurring transmission delays are not the fault of this protocol. It uses a minimum of resources for this process. Moreover, a successful communication implies, that the integrity, authenticity and confidentiality of a package is secured. All interference in fulfillment of these three conditions, for example a high energy consumption, must be the fault of other protocols and can not be fixed by a routing protocol.
Thus, a routing protocol for OppNets needs a clear priority, since the requirements are not easily fulfilled all at the same time. A protocol should still try to run optimal in a suboptimal environment, with a suboptimal number of nodes and a suboptimal node distribution. A network is only useful, if successful communication can take place. Thus, a high delivery quota should always be the top priority to establish a reliable network. Because we are handling OppNets, a faster transmission will most likely increase the needed resources rapidly, so a careful cost-benefit evaluation is required, when implementing routing features. In Opp Nets, 100% delivery quota of a routing protocol means, that the used routing protocol will route all packages correctly to their destination or a fitting carrier, if possible. If so, a routing protocol might get close to this value and be considered as a good protocol, but the overall delivery probability may suffer from missing paths in certain scenarios.

An optimal routing protocol would also prevent all attacks, which could be based on its structure. Since a lot of safety issues are connected to routing, I propose to use an extra layer of security, which implements some kind of safety protocol, specialized to tackle common risks, problems and any kind of attacks for and on OppNets. Some of these issues are data mining, tailgating, defamation, selfishness and various forms of exploiting, which are explained in detail by Gregwood and Henderson [BH13].

2.2 Taxonomy of Opportunistic Routing

A taxonomy exists for categorization and therefore for simplification of understanding and communication. Thus, a taxonomy for opportunistic routing (OR), which derives from already existing approaches named in [MM13], is proposed here.

There are many characteristics, which can describe a routing scheme and at the moment no standards for OR exist. Since we are talking about OppNets, all handled protocols and ideas can be specified as delay tolerant. One could imagine a non delay tolerant approach and realize, that the delivery probability is ruined in most cases. Now some approaches of categorization are stated.

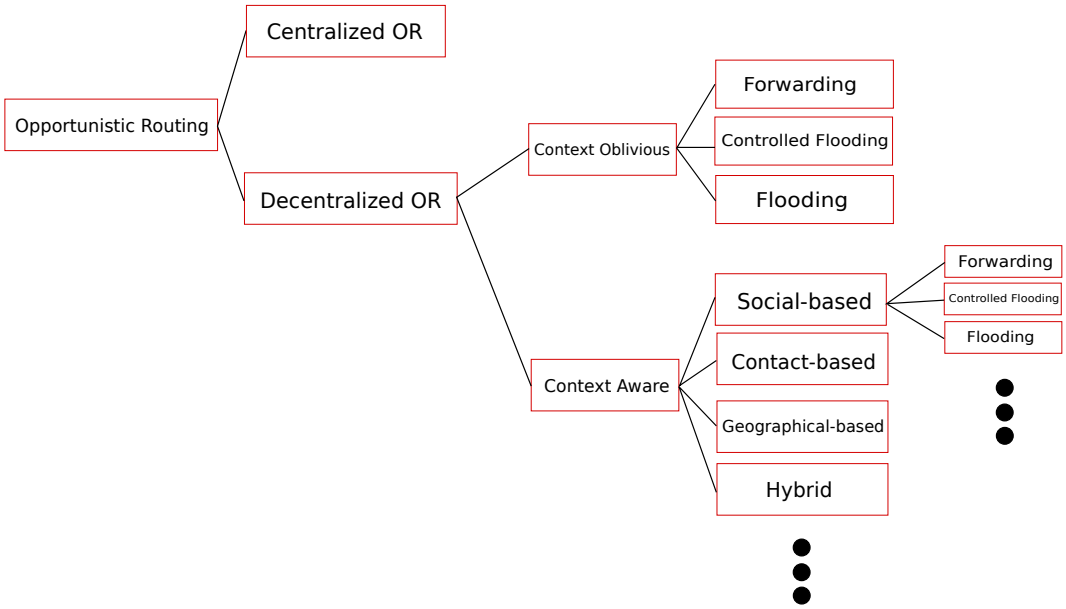
The probably catchiest way to distinguish OR is how packages are handled, as proposed by various sources. If only one copy of a package can be on it's way, we talk about a forwarding-based approach. If there are a lot of, but still limited, copies of a packages in circulation, according to certain rules, the approach is called quota-based or controlled-flooding-based. If there can be an unlimited number of copies, we have a traditional flooding-based approach. The last two cases can also be considered as replication based.

Another approach to distinguish protocols, is where and when a routing decision is made by the protocols. Three common cases are, the sender node could decide the whole path a package travels or every node makes a routing decision for the package or a centralized instance could make all routing decisions. This approach is comparable to the approach of proactive and reactive routing, proposed by [JFP04] and others.

A more promising idea is to distinguish OR decisions based on used information as proposed by [VPI13]. A decision could only be made through abstract, non user dependent, information and therefore we call this context-oblivious routing. Moreover a decision could be made based on contacts, social or geographical data and therefore we would have a context-aware routing. The mentioned approaches are contact-based, social-aware and position- or movement-based. An extension to this approach, is the distinction in partially context-aware and fully context-aware, comparable to the proposal of using the so called knowledge level in [JFP04].

Since a taxonomy exists to improve communication and understanding, it should not be too complex, unless it is necessary. My proposal is to first distinguish between decentralized and centralized OR, next, distinguish in context-aware and context-oblivious schemes and then in forward-, quota- or flooding-based. The protocols used contexts can help for further distinguishing. Figure 2.1 shows the proposal.

Figure 2.1: Proposed Taxonomy



2.3 Related Work

A. Ippisch developed a multilayer framework for opportunistic networks as an Android application [Ipp15], called opptain. The routing concept, which is developed in this thesis, will be implemented to work in his developed network application.

In literature like [WDA+13], many interesting ideas and approaches of opportunistic routing are proposed, but unfortunately most of them are of complete theoretical nature. In the following some of these are discussed.

A. Vahdat and D. Becker did a flooding approach called Epidemic [VB00]. Two nodes, which meet, exchange so called summary vectors, which contain information about their buffered packages and use them, to determine, which packages each are missing and then exchange those packages, so both have all packages after the meeting. Packages are carried like an epidemic and every node met gets “infected” with it. This is a working approach with fast delivery, but the vast amount of resources needed in large networks is undesirable. To tackle this problem, it is proposed, that an upper bound on hop count and buffer space has to be set.

A. Lindgren et al. proposed an extension to Epidemic called PROPHET (Probabilistic ROuting Protocol using History Of Encounters and Transitivity) [LDS03]. An interesting approach, which adds knowledge about certain movement patterns through contact histories. Every node additionally has a delivery predictability vector, which gets actualized with every meeting and is changed, if two nodes do not meet each other for a long time. A forwarding algorithm can use both summary vector and delivery predictability vector to decide which package should be copied for the met node. In theory, PROPHET should save resources while still having a low latency delivery, but as the new introduced vector does not help for a multihop routing decision, many worst case scenarios affecting the resources can be imagined.

Spray routing is an idea proposed by T. Spyropoulos. The simplest version is Spray and Wait [SPC05]. The sender "sprays" the package, which means, that he distributes it L times to carriers, which then carry the package in the, so called, wait phase, and only transmit it, when the receiver is met. There are multiple enhancements to this. One is the Spray and focus [SPC07], which has the same spray scheme but in the focus phase forwarders are used instead of carriers. Spray routing is a simple mechanic of quota-based flooding, which promises low costs, good usage in local OppNets and even good usage in larger OppNets, if the right enhancements are used.

With Label Routing the authors of [HC07b] introduce social aspects to OppNets in form of tagging. Labels are used to determine similarity in context of communities, used as forwarding decision helper. With [HC07a] and [HCY08] they propose a similar form of social driven routing with more practical usage, called Bubble Rap, which could be seen as a social-aware protocol derived from the idea of Label routing. In Bubble Rap community information and node popularity rankings are used to find good forwarders. Packages are only forwarded to more popular nodes or the destination. Every node has one global and many local rankings, one for each community. To route a package, the package is forwarded to more popular nodes until a node from the aimed community is met, in which the destination is present. Then the package is forwarded to more popular nodes inside the community, until a node from the aimed subcommunity is found and so on. This is an interesting approach for resource conserving and effective routing, but is strong dependent on the right use of community information.

HiBop: History-Based opportunistic Routing Protocol, by [BCP07], is a context-aware protocol, which can be considered as an extension to PROPHET. It adds a so called Identity Table and a History Table to the existing scheme, which helps to determine a good forwarder. This Identity Table contains personal information, in which every user can enter the information he wants, like name, e-mail address, mac address, real address and other. The History Table exists to track changes in the Identity Table and therefore track the evolution of the context. HiBOp uses a complex forwarding policy algorithm with weighted attributes and an emission phase, which is comparable to a spray phase, and after that a forwarding and a delivery phase. It uses vast amounts of personal information and a complex algorithm to achieve high efficient routing, but it is to be tested, if high complexity and this amount of information is really needed for fast and efficient routing and it is desirable to use as little personal information as possible.

In [WB+14] adjacent wireless link correlations and their impact to forwarding are discussed and tested. They achieved 38 % less transmissions by using a link correlation-aware design in their tests.

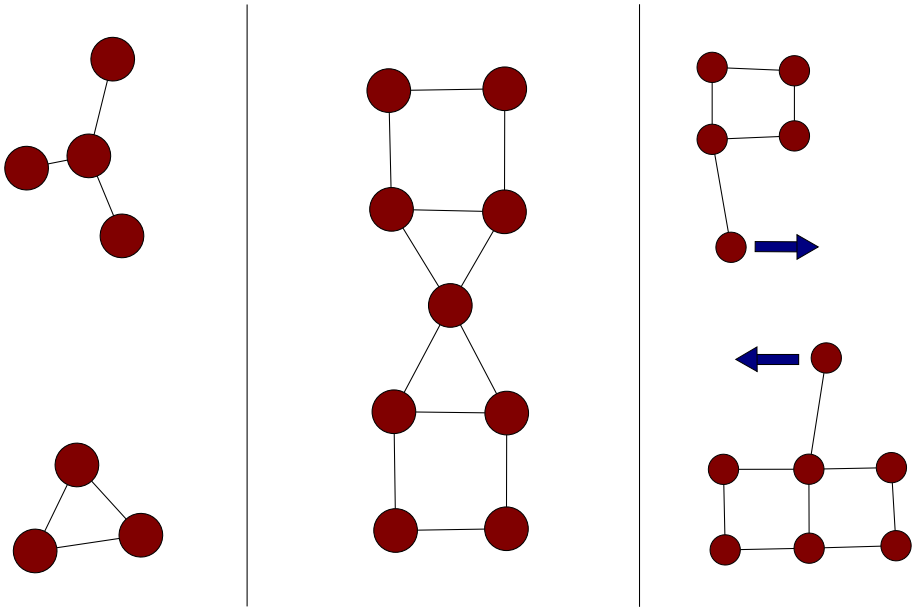
Some honorable mentions are SimBet [DH07], using betweenness centrality and similarity, and a protocol called dLife [MM12], which introduces connection duration to calculate social connections. Another interesting approach, but with centralized instances, is the use of data ferries or similar objects for data collection, proposed by [ZAZ04] and others. In [ZR03] a randomised geographical approach is proposed.

2.4 OppNet Scenarios: Worst-, Ideal- and Urban-Case

Mobility patterns and scenarios are of strong interest, when handling OppNets, and there are already many models, which are OppNet related. In our case, nodes are mobile devices in human hands, so human-like mobility patterns, like Truncated Levy Walk by [RS+11], are our primary interest. In the following, three stereotypical scenarios are proposed. First is the worst-case. An OppNet with a finite number of small and static accumulation points, where the nodes only move inside a boundary, and nodes from different points never meet or do not meet in an appropriate time frame. OR is only

possible in local areas. Second is the ideal-case. Either constant bridges or connections exist between accumulation points or nodes are diffused in a way, that they create a grid-like structure. This allows us to use Internet-like protocols, since we have guaranteed connections. The last is the urban-case. A finite number of accumulation points exist and every node follows certain movement patterns with non movement phases. Nodes move in an appropriate time to other points or at least meet other nodes, which move to this point. Moving accumulation points can exist. Based on these examples, one should see, that by raising the maximum transmission range of nodes, every case can be turned into the ideal-case and certain movement patterns are able to benefit an OppNet.

Figure 2.2: Scenario Cases



From left to right: Worst-, Ideal- and Urban-Scenario

2.5 Impact of Limited Resources

Since nodes in OppNets are usually private mobile devices, which depend on their battery pack as an energy source, the energy consumption to establish an OppNet and to run an OppNet application is very important. The user's experience with OppNets should not be influenced by vast energy consumption. Moreover, memory usage needs to be handled

right but loses importance, as memory space in mobile devices grows larger every year. Dependent on the used protocol, plenty buffer space is needed for high delivery quotas and less latency. I recommend a dynamic approach for memory allocation. Two main points are Energy-Latency Tradeoff and the impact of limited resources on the OppNet, both dependent on the used routing protocol. It should be clear, that we can lower the transmission and delivery latency by using more energy. During the development of a routing protocol, one should keep an eye on energy and memory usage, and decide which consumption degree is acceptable, since many devices differ a lot in their specifications. Another point is the diverse topology. The optimal route, which would be the shortest and fastest, might be so difficult to determine, that it may take more effort, than to take a suboptimal route. A node which is often on the move, could be used as a carrier more often than others, therefore it needs more buffer. This would be a problem, if we use a fixed amount of buffer. As OppNets can be very complex, more special cases like this could encounter.

First off all, there are many ways to examine energy consumption. One way to determine it, which one may find complicated, but may deliver interesting correlations, is to calculate the mean energy distance ratio per bit (EDRb) in J/m/bit [Gao02]. With this and the mean latency, routing protocols could be compared with regard to their energy consumption. A less theoretical and more practical oriented approach is to compare standby time from devices with a running OppNet application and without.

In [MM13] protocols were tested in two scenarios on their delivery quota, their costs and latency with the Opportunistic Network Environment simulator (ONE), see [Kerng] and [KOK09]. Since the buffer was held little at 2 MB and tests were conducted with different time-to-live, up to three weeks, the flooding-based protocols, Epidemic and PROPHET, had the worst results in costs and delivery probability, due to full buffers. Spray and Wait and the social-based competitors Bubble Rap, dLife and the extended version, dLifeComm had better results. Even in the latency category, Spray and Wait resulted a little better than the flooding-based approaches. This shows, that the vast memory consumption of flooding can not compete, if resources are limited. Limited memory can be a serious problem, if one relies on high number of copies as a low latency routing strategy. The simulation had message sizes from 1-100 kB. Comparing to multimedia sized files, the whole network could get congested, even with large buffer

space, if no max-hop-count or other regulation is present.

More limitations are, the number of nodes in the network, their maximum transmission range and their speed. The number of nodes determines, how many opportunities we get to forward a package and indirectly, how high our chance is to find a good forwarder. The maximum transmission range determines, how many nodes we meet and how long we have time to transmit data, moving or not. A small number of nodes could always be compensated by a bigger maximum transmission range. In an ordinary case, two pedestrians pass each other, both at a speed of 2 m/s (7.2 km/h) and a maximum transmission range of 15 meters. This would allow a 7.5 seconds connection. If we take 2.5 seconds tolerance including routing decision and a bandwidth of 8Mbit/s bidirectional, both could transfer 5 MB of data to each other. This would be sufficient, but 30 meter maximum transmission range would already allow 12.5 MB data transmission on the move. There are new standards like 802.11ac, which offer up to 1Gbit/s on a probably small coverage area, described in [Kas13]. In the future we might also see some similar interesting new development in mobile hardware, affecting transmission speed and interference avoidance.

2.6 Most Promising Approach

Since a decentralized routing algorithm is developed, every node should be able to make a routing decision for itself. A spray scheme is promising as an initial routing action, while sending and a context-aware approach should save resources and have a good shipping speed in comparison to the context-oblivious and pure flooding based approaches. Therefore, a quota-based context-aware hybrid is currently the most promising approach. However, one may like the idea of an OppNet with centralized instances, comparable to already good working allocation systems, like goods allocation systems for warehouses and supermarkets or a mail allocation systems.

Chapter 3

Design Proposal

In this Chapter design ideas for OppNet routing protocols will be listed in Section 3.2 and 3.3. These will result in a protocol proposal in Section 3.4. In addition, basic security functions, buffer and time-to-live functionality will be discussed in Section 3.5 and 3.6.

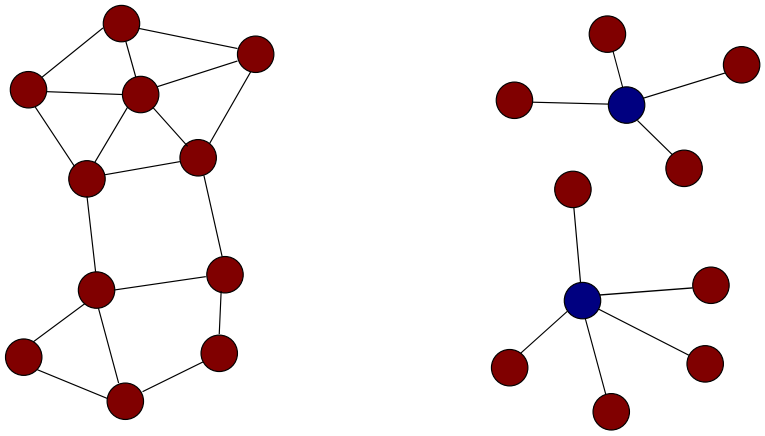
3.1 Special Case Android

In theory nodes can connect to every node in range at the same time, but that is not the current case in an Android environment. In the current version of Android it is not possible to host a hotspot while being connected to another access point and it is also not possible to connect to multiple hotspots at the same time, as explained in [Ipp15]. Since we use tethering hotspot technology for our OppNet, the network gets fragmented in groups with one server and many clients in each group.

3.2 Not Implemented Ideas

One promising routing scheme is the creation of movement profiles. Beside offering new information about movement patterns, this would enable a node to calculate a meeting chance for a peer and the destination, and therefore could decide, if the peer is a good

Figure 3.1: Wanted-Case versus Android-Case



forwarder. All a node needs, is the destination address and a movement profile of it, or at least a known node, has a movement profile for the destination. Since this is a security problem for the user and also needs many resources, this feature will not be implemented. There might be use or need for alternative addressing in the future. Every user could choose a fictional name tag and an additional number id, used to bypass duplicate names, which then can serve together as an alternate address. This feature is interesting but not needed at the moment and therefore not implemented. Some sort of social- based multicast could be desirable in the future but will also not be implemented, due to missing use cases for needed functionality testing. Another valuable function can be the automated analysis of package flow, path length, needed time and other data, interesting for security and optimisation of routing parameters. However, a decent implementation, with testing and evaluation of data, takes too much time to be conducted in this thesis.

3.3 Proposed Design

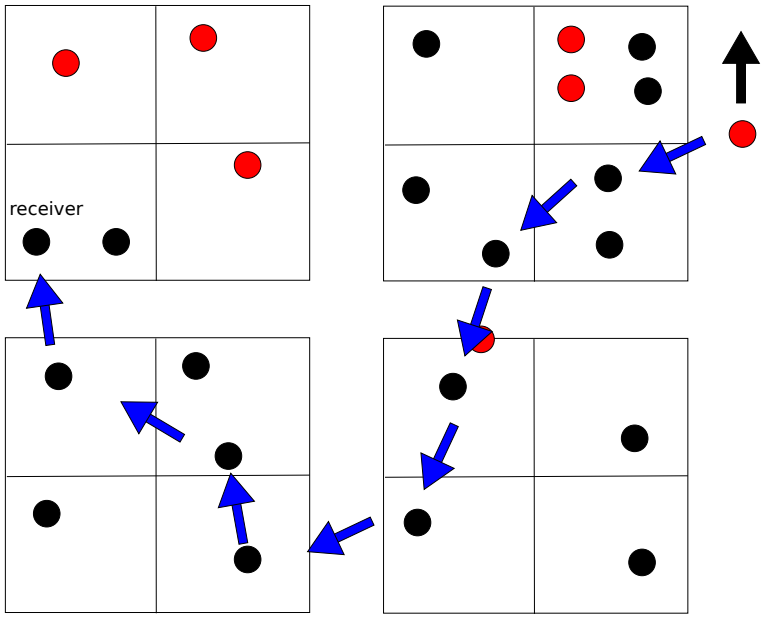
A routing protocol for OppNets should offer a fast transmission in close range and good transmission latency on larger distances. Ultimately a good delivery quota on continental scale could be possible, but with great delays. Moreover, in close and medium range, an algorithm should be able to route between foreign, moving and static nodes.

Our common usecase is a scenario in a bureau building, where many nodes are present. Some of them are moving constantly, but most of them have long resting times and occasionally move through corridors. This shows, that we have a partially context about the whereabouts, which enables us to aggregate nodes to cluster like structures, usable for routing, see Figure 3.2. In a scenario where a node enters a bureau with a not to spray message, destinated to a node present in this building, but meets no node, which knows the destination, leads to the usage of cluster like structures. We may be able to determine a route to a destination over nodes, which are unknown to the destination. If the unknown nodes are in close range to the destination but do not meet the destination in person, they do not know each other via a meeting or their contact summaries. Still, they can be in a cluster with it, because bridge nodes exist between them and the destination.

When two nodes are relatively static to each other, they should enter an aggregation mode, in which they share their decent neighbours and decent neighbours of decent neighbours and so on. If A and B are in aggregation mode to each other and B is neighbour to C, which stays a certain time in range of B and therefore becoming a decent neighbour, B should consider aggregating C into a group with A and himself. A does not know C, but A knows, that it can reach C via B, because B informs him about his new neighbour. When now a new node D meets A and wants to deliver a message to C, A can provide the information, that he is in aggregation mode and C is part of his cluster. Because of that, D should decide to give A a copy of the message or even make him the carrier. This method could be used in large buildings, with many static nodes, but scalability needs to be considered, since a lot of routing information would be needed to be propagated periodically.

In short and mid range scenarios with constant moving nodes, such an aggregation makes no sense. Nodes could be only aggregated through social or geographical boundaries to large area clusters. Nodes can create contact or meeting summaries and send them to peers, making that information usable for routing decisions. A meeting time per day or a meeting count could be used to determine a good forwarder. Node A, which wants to deliver to node C, meets node B, with an average meeting time per day of eight hours with C, which therefore is identified as a good forwarder. Moreover, nodes could collect data about meetings from other nodes, which they meet often. That would enable to calculate a good forwarder, based on the number of nodes this candidate knows, which

Figure 3.2: Path of a Message in a Cluster



Black nodes are aggregating with each other but do not know each other necessarily. Red ones are foreign.

know the destination.

Another approach is to use social labels. These labels could be a fixed information, derived from used social media or entries from the user himself. Meeting nodes could decide, which message each of them can forward, based on their social match to the specified destination. This requires, that the nodes know the labels of the destination or at least meet nodes which do.

For long range routing, a similar method could be used. Users could set two geographical label groups, one set for their home and another for their workplace. This would enable nodes to identify other nodes as good forwarders, in decreasing circles, to a destination. In a scenario with three close cities, a node A from town X is a forwarder to node B from town Z. In a normal case, A would only forward to nodes from town Z or to nodes, which directly know B. But what, if A does not meet such nodes. Node A could use GPS to determine the distance to the approximate location of B and could identify a node C as a good forwarder, since C is from town Y, which is closer to town Z. Therefore, if a location function, like GPS, is available on the used device, it can be used to calculate

the distances from the peers hometown and workplace to the destinations hometown and workplace, if known, and a forwarder could make a routing decision based on that.

3.4 Protocol Proposal

Since a quota-based context-aware hybrid seems most promising, I propose the following protocol as a theoretical approach to the already named design ideas:

NASGL-Protocol: Neighbourhood Aggregating Social and Geographic Label Protocol

When two nodes meet, they periodically exchange all known routing information. Both of them determine the meeting time, keep a list of peers with which they have the longest meeting times and their average meeting time per day. They should also keep the contact/meeting summaries of these peers. When two or more nodes are static to each other, they should enter an aggregation mode and propagate all decent neighbours they know to all other neighbours and to new nodes, which arrive. Every node should have ten geographical labels, five for their home and five for their workplace or similar. These five tags are country, state, city, district and street. District and street should be optional. They also have seven optional social labels, which should have a top to down priority. If available and wanted, location getting hardware can be used.

The routing decision is made by every node by itself and a node has the right to reject a package. The decision is made using following rule:

If one of the following applies, with a top to down priority, the peer is an adequate forwarder or even the destination:

1. Is the peer the destination?
2. Is the peer directly connected to the destination?
3. Are we aggregated with the peer and the destination, and does a path exist from peer to destination?
4. Is the peer aggregated with the destination?

5. Does the peer know the destination or another peer who does, but we do not?
6. Do we both know the destination or other peers who do, but the peer knows more peers, who know the destination and does have more social and geographical similarity to the destination than us?
7. Does the peer not know the destination, but the peer has a closer home or workplace to the destination than we do?

The original sender himself should always keep a copy, forwarders should decide for themselves, if they want to keep a copy. Similarity is to be calculated with the use of the labels, as an additional calculation factor in form of a scoring method.

As an initial routing scheme, a binary or a normal Spray scheme should be used, if we can not answer the first two or four questions immediately. The used quota L can be decided individually, but should not be over a handful of copies to prevent massive flooding. All packages should either implement a max-hop-count or a time-to-live. Furthermore, if we are aggregated with the destination but question three can not be answered with yes, we should not consider the following questions to prevent flooding our cluster needlessly.

3.5 Simple Security Layer Proposal

A simple security layer should be implemented to tackle common risks and security problems in OppNets, which are related to routing and networking. In the following, some features, which should be part of such a layer, are listed. Every peer should be able to reject flooding packages, packages from other protocols or unknown protocols and try to route packages with the protocol used by the sender, if that is possible. Selfishness can be a big problem in OppNets, thus every node could implement a counter for delivered packages to identify selfish nodes. Since this leads to a defamation security problem, if not implemented correctly, explained in [BH13], no such functions should be implemented unless needed. Furthermore, a verification system, comparable to the web of trust, could be implemented to verify non malicious users.

3.6 Time-To-Live, Max-Hop-Count and Buffer Handling

In OppNets, delivery probability and quota can be increased by adjusting TTL, Max-Hop-Count and buffer right. To decide on certain strategies, simulations and real life test in large areas with certain scenarios need to be conducted to gather reliable data. Therefore, I cannot make a reliable proposal on this. However, following statements can be made:

- An OppNet routing protocol might benefit from local TTLs based on routing decisions rather than global TTLs
- The number of (social) hops to reach every person on earth is relatively small, therefore a max-hop-count is to be considered as a decent solution of certain problems (compare "Six Degrees of Separation" and "small-world experiment").
- It might make sense to use global and local TTL and a max-hop-count at the same time.
- In certain scenarios it might not make sense to keep old packages in buffer and refuse to take new packages, for which we might have a significant higher delivery probability. Therefore, buffer should be handled differently in OppNets.
- The routing decision can be a metric to adjust these values.

For example, a node, which forwards packages to the destination, can delete those, if it is certain, that the package transmission was a success. A node, which was only a forwarder due to a geographical similarity could set the TTL of a package to smaller values, if it forwarded the package multiple times to nodes with higher similarity.

Chapter 4

Implementation

In this Chapter the implementation of the proposed routing protocol, described in Chapter 3, will be discussed in detail. In Section 4.1 the used tools and their versions for implementing and testing are listed. The implementation is described with four Subsections, from basics to more detailed descriptions, in Section 4.2. At the end of this Chapter, in Section 4.3, other interesting implemented functions, classes and modified classes will be named.

4.1 Tools

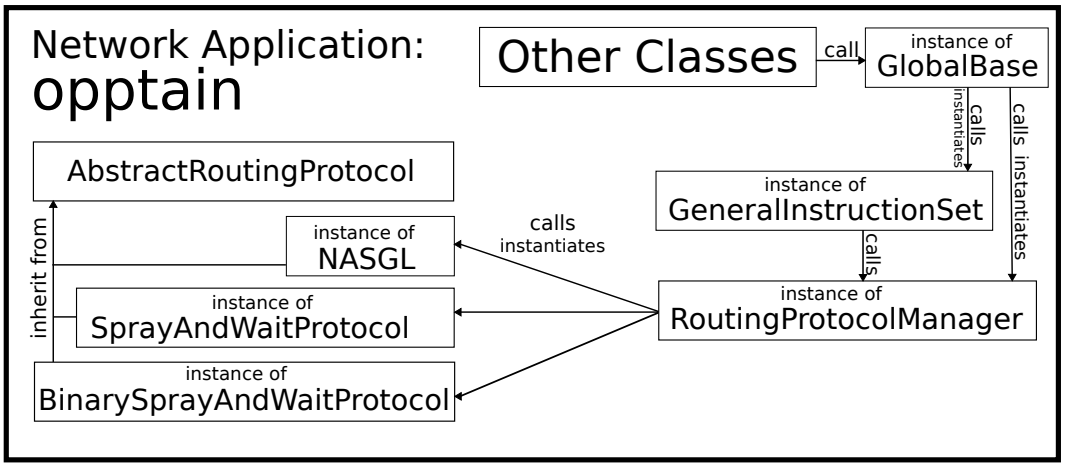
For implementation, the IDE Android Studio Version 1.5.1 for Windows 10 and Version 2.1.0 for Linux Ubuntu, with Java JDK 1.8, was used to integrate my classes into the given network application, developed by A. Ippisch. Android SDK Tools 25.1.6, Android NDK 11.0.0 and Android Support Library 23.2.1 were the current versions. The slf4j logger, which was already in use in the project, was used for debugging in addition to the USB-Debugging function of Android devices. The application is currently developed for Android 4.1 or higher. Git is used as a version control tool for the project. For more detailed information about the network application and licensing see [Ipp15].

4.2 Routing Implementation

4.2.1 Basics

The network application should be able to use multiple protocols, preferably at the same time. Therefore, a *RoutingProtocolManager* and an abstract class *AbstractRoutingProtocol* have been implemented. All protocols implemented in the future should inherit from the abstract class. The protocol manager serves as an interface between the application and the protocols. In the application a protocol manager gets instantiated and the methods of this instance are called when needed. A class, containing general instructions, which are called at certain key points on run time, calls the methods of the routing protocol manager with the needed parameters. The general structure of the implementation can be seen in Figure 4.1 and all implemented and important modified classes can be seen in Table 4.3 and 4.4 at the end of this Chapter. For more detailed information, see the auto-generated digital documentation and the code in the appendix.

Figure 4.1: Classes Overview



4.2.2 Routing Decision and Information Managing

The routing algorithm itself follows the proposed protocol and can be seen in simplified form in Listing 4.1. Needed information is collected before the actual routing decision

is made. The shown algorithms parameter are an *OpptainBundle*, which is nothing more than another form of a package, used inside of the network application, and the address of the peer we are connected with. Depending on the result, the *OpptainBundle* will be added to a list of bundles, which will be offered to the peer later. If the first four if-statements are false, a binary Spray and Wait scheme gets used to share a package unconditionally until the specified quota of copies is reached. The quota will be adjusted, when aggregation is used as a routing decision too, as the aggregated forwarder is also at a better position for spraying. Binary Spray and Wait shares packages with a binary strategy rather than single copies, meaning that it halves the numbers of copies every time it sends, but the peer, which receives these copies, shares these received number of copies again. This allows multihop routing in static environments with the number of maximum hops dependent on the chosen spray quota. The network application instructs the peers to first make their routing decision and offer the chosen packages to each other, then they can decide which packages they need or want to reject, due to their policies. Therefore, the quota will only be adjusted, if the current peer validates the acceptance of the package. Information about peers and ourselves is stored in an instance of the class *RoutingInformationManager*, which contains information shown in Table 4.1. This class contains every peer related information, which is shared via network.

Listing 4.1: NASGL Routing Decision Algorithm

```
1  boolean forward(OpptainBundle bundle, address peerAddress)
2      RoutingInformationManager peerInfo = peerInfoMap.get(peerAddress)
3      address destination = bundle.getDestination()
4      if destination == peerAddress
5          return true
6      end
7      if peerInfo.neighbourList().contains(destination)
8          return true
9      end
10     if myInfo.aggregationStatus and ownClusterMap.contain(destination)
11         if ownClusterMap.contain(peerAddress)
12             and aggregationManager.pathExists(peerAddress, destination)
13             return true
14         else
15             return false
16         end
17     else
```

```
18          if peerInfo.aggregationStatus and peerInfo.clusterMap.containsKey(destination)
19              return true
20          end
21          if bundle.getInfo.get("Quota") > 0
22              package is accepted: bundle.getInfo().putString("Quota", bundle.getInfo.get("Quota")/2)
23              return true
24          end
25          RoutingInformationManager destInfo = createBundlesInfoManager(bundle)
26          forwardersSelf = calculateForwardersScore(myInfo, destination)
27          forwardersPeer = calculateForwardersScore(peerInfo, destination)
28          myScore = score(myInfo, destInfo)
29          peerScore = score(peerInfo, destInfo)
30          if forwardersSelf <= 0 and forwardersPeer > 0
31              return true
32          end
33          if forwardersSelf > 0 and forwardersPeer > 0
34              if forwardersSelf + myScore >= forwardersPeer + peerScore
35                  return false
36              else
37                  return true
38              end
39          else if forwardersSelf > 0
40              return false
41          end
42          if everyInvolvedUsesGeoCoords
43              return amICloserToDestination(coordinatesPeer, coordinatesMe, coordinatesDest)
44          end
45          if peerGeoMatch(peerInfo, destInfo) > ownGeoMatch(myInfo, destInfo)
46              return true
47          else
48              return false
49          end
50      end
51  end
```

There exist some key points in the routing decision, which stop further calculations to prevent flooding. First is the third major if statement, in which we decide not to forward a package, because we are aggregated with the destination, but the peer is not part of the path to it, to prevent flooding around the destination. The second one is the if statement

in the big else block, in which we calculate the scores. We do not forward a package to nodes which are less likely to meet the destination to conserve resources. The third key point is directly after that. We do not forward a package if we know the destination, but the connected peer does not. This should prevent unnecessary package distribution.

A simple scoring function is implemented to calculate similarities between nodes, seen in Listing 4.2. The not shown parameters are a *RoutingInformationManager* instance and an address for the *calculateForwardersScore* and two *RoutingInformationManager* instances for the *score* function, from which it retrieves the compared labels. The ultimate score should represent a possible time frame, in which a peer is in direct or indirect contact with the destination per day. Meeting time of other nodes is worth one fifth of our meeting time. This means, that if three nodes, I know, meet the destination for about one hundred minutes per day, one hour meeting time will be added to my meeting time with the destination. The calculation methods for geographical and social labels compare own labels and the peers labels with the already given labels from the package, which is to be forwarded, in a prioritized order and return a number, which represents the total similarity.Then this number can be used in a routing decision, as seen in Listing 4.1 (l.34).

Listing 4.2: Scoring and Possible-Forwarders Calculation

```
1  variables from RoutingInformationManager instance:
2  Key-Value-Map Keys: Address, Values: contactMap contactSummayMap
3  Key-Value-Map Keys: Address, Values: meeting time ownContactMap
4  function calculateForwardersScore
5      forwarderScore = 0
6      forwarderScore = forwarderScore + value of destination in ownContactMap
7      for every key in contactSummaryMap
8          if value of key contains destination
9              forwarderScore = forwarderScore + ((value of destination in contactMap)/5)
10         end
11     end
12     return forwarderScore
13 end function
14
15 function score
16     myScore = calculate geo match * settable geo modificator
17     myScore = myScore + calculate social match * settable social modificator
```

```
18      return myScore
19  end function
```

4.2.3 Information Exchange and Contact Summaries

Information of peers is distributed by the peers themselves, sending special information packages to their neighbours periodically, and adding their sender data to normal packages they create. Information packages contain all data from the *routingInformationManager* instance, while normal packages only contain geographical and social components of the origins data to reduce unnecessary overhead. Therefore, a sent normal package contains known social and geographical information of the destination and origin from the *routingInformationManager* class instances. Receiver and forwarders can save the origins data to route packages back to the sender. Information is distributed at contact and afterwards in a settable periodical interval by a special thread. This can be changed, in further application development, to send information packages only when some of the data has changed. It is currently not wanted, that peers, which meet, exchange information about other peers, because that could be used to distribute misinformation and therefore is not implemented. Only saved contact summaries from other peers get exchanged as routing information, but will not be saved by the peer who receives them. This also affects the process of saving sender data. In the current implementation, it would be easy for a rooted device to change the sender or destination data in a package to misleading values. This issue needs to be fixed in the future. The *ContactSummary* instance contains the peer's address and a contact key-value-map, which stores a settable maximum amount of addresses of met peers and the relative meeting time in milliseconds. Every peer has its own contact summary and the contact summaries of the peers, with which they have the longest meeting times. These contact summaries get stored in a persistent *SQLite* database. Meeting times get halved every day, time frame settable for debug purpose, so the maximum meeting time at the end of the day can be 48 hours and then halved to 24 hours again. However, this is not a realistic case and even not possible in the current implementation of the network application, due to periodically disconnects in automation mode. The time of the last update gets saved, so the application can check how many days have past, since it was last used and do missed updates.

Table 4.1: Variables in RoutingInformationManager

Type	Name
String	mAdress
GeoTag	mGeoTag
GeoTag	mSecondaryGeoTag
boolean	mUseGeoCoords
double	mLatitude
double	mLongitude
double	mSecondaryLatitude
double	mSecondaryLongitude
List<String>	mTags
ContactSummary	mContactSummary
Map<String, ContactSummary>	mContactSummaryMap
ArrayList<String>	mNeighbourList
boolean	mAggregate
Map<String, HashSet<String>>	mClusterMap

"m" stands for member, meaning it is a private variable.

4.2.4 Routing Maps and Aggregation

To process and update routing maps with information from information packages, an aggregation thread, which follows the algorithm seen in Listing 4.3, is instantiated by the protocol constructor. It optimizes routing maps to a loop free and tree like graph with the algorithm seen in Listing 4.4. This structure can then be used for a recursive path finding algorithm, without actual path collection, because we do not have loops in our map. To find shortest paths and delete dead ones, a modified breadth-first search algorithm is used with unweighted paths and hops instead of depth. The breadth-first algorithm searches until a maximum hop count or a scale boundary is reached. Optimization means, that we get a smallest possible data structure, which contains all information we need to determine, if a peer is a suited forwarder. We may neither know every possible route nor the best route, due to misinformation, but to know one route is more than enough for an OppNet routing protocol. Topology changes may alternate the best route of the aggregated nodes far away, but we will route to a node, which will know those changes and maybe even the current best route. The run time of the routing maps optimization for

the worst case, which is a full meshed map of n nodes, is in $O(n^2)$, while for the best case, in which the map already is optimized, it is in $O(n)$. This is because the algorithm needs to iterate over every entry of every node's hash set at least once and in the worst case, every node has an edge to every other node, meaning every hash set contains n entries, while in the best case, there exist only n-1 edges. To determine decent neighbours, suitable for aggregation, we calculate our current meeting time with peers and when it reaches a certain point, a peer is considered as decent. The function is tolerant to the current topology. The algorithm adds more time than actually has past to cover up absence times, which originate from hotspot and server to non hotspot and client changes, so nodes aggregate with other nodes to which they are periodically connected to. In the current Android case, described in Section 3.1, information about topology changes will most likely travel slow, therefore a settable scale boundary has been implemented to keep the routing maps small. This also prevents, that information packages contain relatively big sized data. A single 64 letter address, with 7 bits per letter, takes 56 byte of space. At events or other gatherings a cluster can easily consist of more than 500 nodes, which would lead to 28 kB overhead, even without adding contact summaries. Hotspot and modern Android devices can easily handle multiple of those packages at the same time, but that would be a waste of energy, invested in sending and iterating over probably useless data.

Listing 4.3: Aggregation Thread Algorithm

```
1  thread start
2      while executeAggregation == true
3          thread sleep for settable interval
4          update List of current neighbours
5          update Map with recently connected peers as keys and meeting time as values
6          if we have a decent neighbour
7              set aggregation status in routing information manager true
8              update the aggregation register, which keeps all decent neighbours
9              overwrite our entry in aggregation map with the aggregation register
10             optimize the aggregation map
11         else
12             set aggregation status in routing information manager false
13         end
14     end
15 end thread
```

Listing 4.4: Aggregation Routing Map Optimization

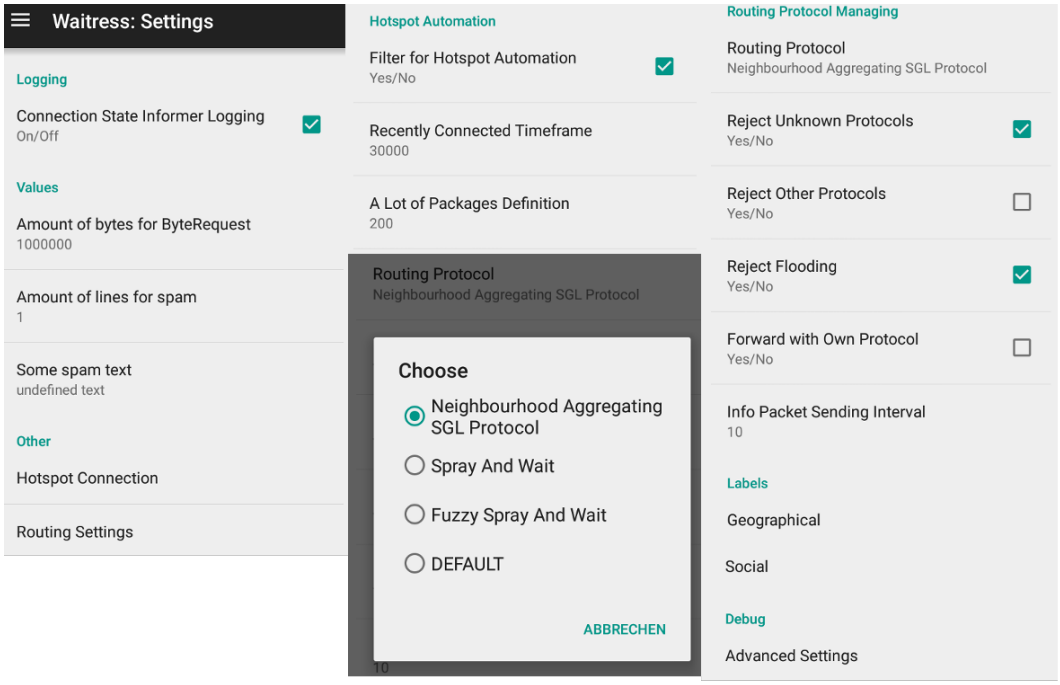
```
1 Key-Value-Map: Keys: Address, Values: HashSet<Address> aggregationMap
2 function optimizeAggregationMap
3     delete addresses from Sets, which are equivalent to their mapped key
4     delete our address from Sets
5     create a key and an empty Set for every value, that is not a key yet
6     modified breadth-first search to delete dead and duplicate paths
7 end function
```

4.3 Other Implemented Functions

Needed constants are either declared in the new *ConstantsManager* class or in the already existing *IntentConstants* class from A. Ippisch. A helper class was implemented to gather information related to hotspot connection, so the application can decide, which hotspot it preferably should connect to or not. Another class was implemented, which tries to get information about the current location in form of latitude and longitude. Then the location can be used as an optional routing information, instead of geographical labels. Moreover, advanced settings for users and debugging were implemented, seen in Figure 4.2, to change variables at run time, choose a routing protocol and save data with help from Android's *SharedPreferences*, which stores Key-Value pairs.

The network application already had functions for manual and automated connection. The automation mode was expanded in a way, that the devices exchange information packages, then normal packages get shared and the connection will be closed and both search for a new connection. They are either connecting to other available hotspots or become hotspots themselves. This decision is made, by using the information gathered from the implemented helper class, which consists of a metric, using the time passed since last connection and the number of packages, which we want to route to a hotspot hosting node. In current manual mode, one user needs to start a hotspot and another to connect to it. One must start a server and another a client. All this can be done via the sidebar of the network application. Information packages will be exchanged on connection but user packages will only be exchanged, if the current client starts the exchange process. The server can either wait for the client to do so or terminate the

Figure 4.2: Settings Examples



connection. A function to create a settable amount of dummy packages with a selectable destination was created for debugging and testing. A device, that is offered an unknown debug package, will make a toast to inform the tester. As a reference protocol for testing, a simple Spray and Wait and a binary Spray and Wait algorithm were implemented.

In the following, tables of all implemented and of all modified classes are shown with their descriptions.

Table 4.2: All Implemented Classes

Name	**Function**
NASGLProtocol	routing
NASGLProtocolHelper	containing long conversion methods
RoutingInformationManager	storing information
ContactSummary	storing meeting times
ContactSummaryThread	calculating mean meetingtime per day
GeoTag	storing geo labels

AggregationManager	building clusters and find paths
RoutingProtocolManager	instantiates routing protocols
AbstractRoutingProtocol	states standard methods
SprayAndWaitProtocol	routing
BinarySprayAndWaitProtocol	routing
ContactSummaryMemory	SQLite
RoutingProtocolMemoryContract	SQLite
ConstantsManager	stores constants for persistence
HotspotDecisionHelper	helps decide for a connection
InfoPackage	send as a routing information package
AdditionalInformationForRouting	information container
InfoPackageThread	sends periodically information packages
LocationDataManager	gathers geographical information
GeneralInstructionSet	logic programmers interface
routingsettings.xml	general settings and debug settings
InfoPackageProto.proto	automated class generation scheme

Table 4.4: Important Modified Classes

Name	**Function**	**Changes**
GlobalBase	keeps instances	instantiating
ClientInfoTcp	networking	apply routing decision
InBetweenTCPClient	automated networking	apply routing decision
StateManager	automatism for networking	use HotspotDecisionHelper
FragmentSettings	gui for settings	add routingsettings.xml
OpptainRemoteService	gathers own packages	add routing info
BundleCalculator	process package offers	rejection method call
IntentConstants	storing constants	new constants
ShellProto.proto	automated class generation	add Info Package

Chapter 5

Tests and Evaluation

In this Chapter the test scenarios are explained in Section 5.1. The results of these tests are structured in four subsections and shown in Section 5.2 and the reflection about the implementation can be found in Section 5.4, where routing, networking and security are discussed.

5.1 Test Scenarios

To evaluate the implementation, the following described tests were conducted on NASGL and binary Spray and Wait.

The first basic test was a direct connection test with two not rooted devices, a Motorola Razr i and a Huawei Y360-U61, both with Android 4.4.2. This test showed, that routing decision, based on destination, and the currently enabled broadcasting works. The second basic test was a test with the same devices and an additional not rooted Sony Z1 Compact with Android 5.1.1. In this test, code were selectively cutted to skip certain decisions and search for bugs in the routing decision algorithm, which were corrected after that. Moreover, working routing decision, based on propagated neighbour lists and working aggregation of three devices, could be shown.

The advanced tests featured up to ten Huawei Y360-U61 devices, which were also not

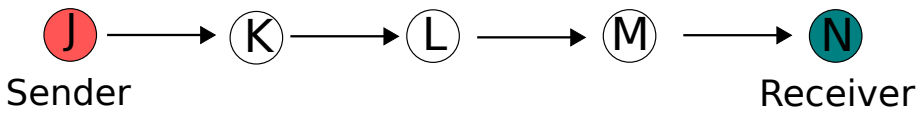
Table 5.1: Dummy Configuration

ID	geo. Label	sec.geo. Label	social Label	sec.soc. Label
J	1,1,1,1,1	A,1,1,1,1	Sender	Cluster
K	1,1,1,1,1	A,1,1,1,1	Test	Hhu
L	1,1,1,1,1	A,1,1,1,1	Test	Hhu
M	1,1,1,1,1	A,1,1,1,1	Test	Hhu
N	1,1,1,1,1	2,1,1,1,1	Hhu	shuttle
O	2,1,1,1,1	3,1,1,1,1	Hhu	shuttle
P	3,1,1,1,1	B,1,1,1,1	Hhu	Cluster2
Q	3,1,1,1,1	B,1,1,1,1	2	Cluster2
R	3,1,1,1,1	B,1,1,1,1	2	0
S	3,1,1,1,1	B,1,1,1,1	Receiver	Hhu

rooted and named J to S. The dummy configuration, which was set for all devices, can be seen in Table 5.1. The spray quota was set to four for NASGL and binary Spray and Wait and all devices were configured to aggregate in 20 seconds or two automated connections. In the following, the conducted scenarios and the order of events are described and the results will be shown in Section 5.2.

The first advanced test featured the five devices J to N. These devices were placed in a row and connected one after another in manual mode to show multihop routing with a spray scheme. J should connect to K, then K should disconnect from J and connect to L and so on, shown in Figure 5.1. Since the spray quota was four and none of the devices were aggregated with the destination, the message was expected to travel two hops to device L and then be forwarded with another routing decision. At the beginning, J gets to know N, then they disconnect and J creates six debug packages for N.

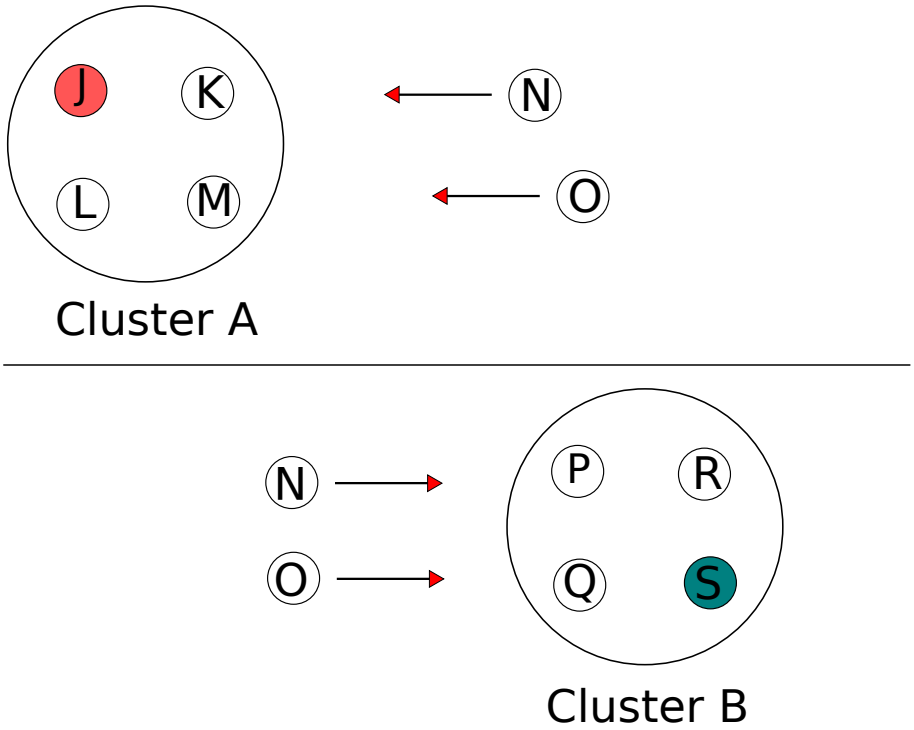
Figure 5.1: Hop Test



In the second test, nodes builded two clusters with four devices each and two nodes working as shuttle nodes between them, as seen in Figure 5.2. This test was conducted in automation mode, but not all devices where in automation mode at the same time,

since the WiFi range of the test devices is too big and they would build one cluster over multiple rooms and floors. First the devices J and S should get to know each other, then S automation gets turned off, while K's, L's and M's will be turned on. J creates three debug packages and it is expected, that J sprays the packages to the other three peers. Then N's and O's automation is turned on to simulate their arrival. After multiple connections, nodes J, K, L and M are to be disabled and another cluster containing P, Q and R is to be established. Later, S should enter the cluster.

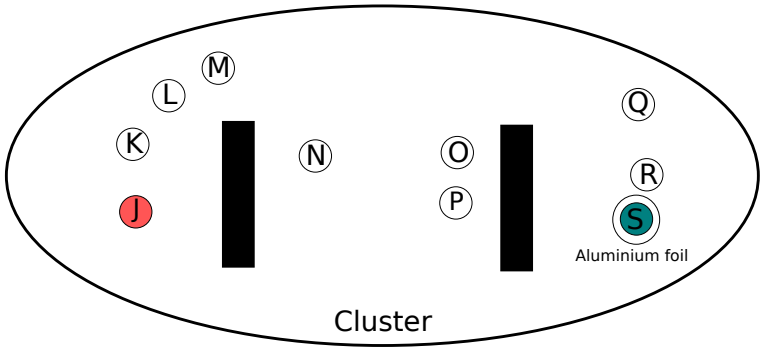
Figure 5.2: Shuttles and Clusters Test



Red is the sender and blue the receiver. The arrows show node movement.

The third test featured a possible cluster of ten devices in automation mode and the receiver only possible to connect to one node. The destination node S was wrapped in aluminium foil to damp the signal strength, and placed at R's side. This should prevent S from directly connecting to the sender J or nodes, which are near J. The scenario is seen in Figure 5.3. J created forty packages, with S as destination, in the binary Spray and Wait reference test and fifty in the NASGL test. This test should show more weaknesses of missing information, while also evaluating the process of cluster package distribution and the automated networking of the network application.

Figure 5.3: Cluster Test



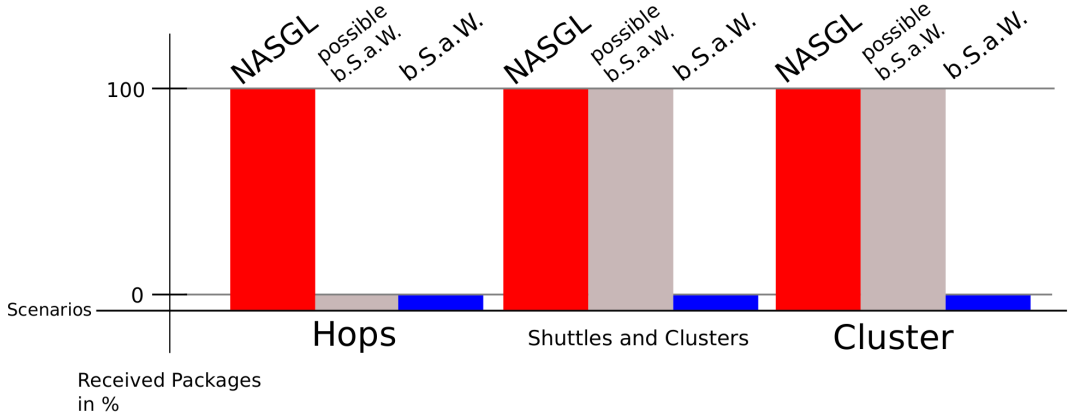
Red is the sender and blue the receiver.

5.2 Results

5.2.1 General Results

A discovery was the moody automation mode. Devices tend to connect to only two or three other devices over and over again, because of the short interval, in which they are supposed not to connect and the current implementation, where a device chooses the first working entry in the WiFi scan list. This may be unwanted but can also benefit the network, because packages will possibly not get shared that much. However, it is certain, that the automation needs to be improved in the future, because the current implementation can prevent aggregation, as seen in some of the test logs. Another aspect is, as expected, that Spray and Wait suffers from static environments and needs a high enough quota to let packages leave this environment or reach the destination in it. The scenarios were specially designed to show flaws of context unawareness, which resulted in bad results for binary Spray and Wait, shown in Figure 5.4. The Figure shows the amount of received packages by the destination of both tested protocols and the amount of theoretical possible packages to be received with binary Spray and Wait.

Figure 5.4: Packages Received



5.2.2 Hop Test

The logs of the first advanced test with NASGL showed, that sender J offered the created debug packages to K and adjusted the quota to two. K forwarded the package to L and adjusted the quota to one. L, now not allowed to forward with the spray scheme to M, used the other options. Because both did not know N, L calculated with geographical labels. Since both have the maximum geographical score, L did not forward to M. Then N was connected to M and L send the packages to M, after it got to know, that N is now a neighbour of M. In the end M forwarded the packages to their destination N. This test showed working routing decisions, based on destination, neighbourhood, spray quota and geographical match. The reference test with binary Spray and Wait showed a clear disadvantage of missing information and small replication quota. Since the reference test was conducted with a quota of four, the debug packages only reached to node L and could not be forwarded further, although a simple information would have been enough to know, that M is a good forwarder.

5.2.3 Shuttles and Clusters Test

The logs of this test on the NASGL protocol showed, that J spread the package via spray scheme and node L forwarded the debug packages to node O but not to node N, because O has a geographical match with the destination, while N does not. As O

and N came in contact with the second cluster, O connected to N, and as N was now aggregated with a cluster, in which the destination is present, N received the packages from O. Later, O connected to P, but P was not yet member of the cluster, due to the unoptimized automation mode. O made the decision to route the debug packages to P based on the scores 18051007 and 18155106. After that, O connected to N, which it was know aggregated with, and now offered N the debug packages again, knowing that the destination is in the cluster and a path from N to S exits. S received its first offer from P, which got his packages from O. The test showed working aggregation, scoring function and that the protocol has the possibility to spare nodes from unnecessary packages, since N only received a copy because of being in a cluster with S. In the test with binary Spray and Wait, J, K, L and M all had a copy when N and O arrived. Therefore, no package could be delivered to the destination. If O and N had arrived at the right time, they may had received a copy and then would be able to forward to destination S, when arriving at the other cluster.

5.2.4 Cluster Test

This test on NASGL showed, that after approximately eight minutes of testing, 3 devices did not receive the packages and some received the packages with a spray decision, because they and the decision making node, were not yet aggregated with the destination. J sprayed two times and L sprayed his remaining copy too, while forwarded once with a decision based on score. However, when the automation mode and the aggregation parameters are optimized, the routing decision algorithm should prevent such behavior and save resources, instead of flooding the destination's cluster. The reference tests showed once more, that Spray and Wait depends on the right quota. Node R only received a small number of packages, while three others received all packages. This was due to concurrent routing decisions, by which R and another node were offered the packages at the same time but the forwarding calculation for the other node was somehow faster, and the quota was adjusted before the rest of the packages could be offered to R. However, node S did not receive this fraction of packages R held, once more due to the unoptimized automation. If this test would have been redone multiple times, S would certainly have gotten packages, but with a quota of just four it is just luck, if the packages can arrive at

their destination, because there are eight other nodes between sender and destination.

5.3 Routing Map Optimization Run Time

Two classes were created to test the time it takes to optimize cluster/routing maps, with the implemented algorithm. One class contains copies of the implemented function, while the other creates random addresses and puts them randomly together to a routing map, creating randomly three to one thousand edges for every node. The mean optimization time for a routing map, in one hundred cycles, with original 100501 entries, including the own entry, and a scale boundary of 500 was 652 milliseconds. It was tested on a Windows 10 Laptop with a Intel(R) Core(TM) i7-2670QM CPU with 2.20 GHz. This shows that the routing map optimization will not be a problem for the usability of the application, since the test showed an unrealistic scenario and the needed time was still acceptable.

5.4 Evaluation Conclusion

5.4.1 Routing and Networking

The goal of the implementation was to create a routing protocol for OppNets, which unites the fast delivery in local areas and good delivery quota of flooding approaches, while being resource considerate like the social-based approaches. The tests showed, that NASGL succeeded at this, since it has the opportunity to spare nodes from unnecessary packages, while still promising a good delivery quota, comparable to flooding with unlimited buffer space. In contrast, flooding approaches easily fail in scenarios with limited buffer space, while Spray and Wait depends on the right quota and node movement. Binary Spray and Wait is able to route in static environment but still needs the right quota. Both are somehow luck dependent, if the quota is not adjusted right, which also can be seen in the shuttle test, where node N could have gotten a package with a spray decision, if it would have arrived at the right time. The implementation of NASGL is working,

but many parameters need further testing and adjusting. The automation itself needs to be improved, while the implemented *HotspotDecisionHelper* class could provide more usable information, for example, it could gather information about the total number of other peers in the local area and adjust values, so that the device connects to many others in a short time or not. It could create a hotspot, if it is in a central position, while with less nodes, the device gets the information to connect longer with other devices and wait for new nodes to appear. Furthermore, the total number of hotspots should be minimized, while all nodes, which can be connected to a hotspot, stay connected. This behaviour would minimize the needed hops, a message has to take to the destination. NASGL spares resources, but the possibilities are still not exhausted, since the routing decision itself can be used for enhanced buffer handling, explained in Section 3.6. These features should be implemented and tested in future versions of the network application.

5.4.2 Routing Related Security

NASGL strongly depends on input by information packages from unknown nodes. This is a security problem, since the input can be nonexistent or corrupted and can cause errors. Input can also be malicious to make nodes do wrong routing decisions. One problem is the aggregation to clusters. A node can set a big or infinite "come back" tolerance carrying its entries into another cluster and gather all packages to destinations, which it has in his cluster map. Moreover, a node can send a fictitious cluster map with a very high number of entries to other packages to increase the calculation time for aggregation maps drastically. In the current implementation, network input related errors are handled and nodes only accept a certain number of entries, dependent on the settable scale values, therefore hindering the described attack forms. However, no security functions exist to detect malicious behaviour. A node may not be flooded but can still be fed with misinformation. The current implementation uses information, which is not specific enough to identify the user or to be used in a commercial way. All in all, the implementation shows, that the right amount of information enhances routing and routing decisions, as proposed by various social routing approaches, while not needing to be too specific about personal information.

Chapter 6

Conclusion and Future Work

This Chapter contains the thesis conclusion in Section 6.1 and possible future work in Section 6.2.

6.1 Conclusion

The goal of this thesis was to develop a reliable routing protocol for Android based OppNets and implement it in the given network application. This has been achieved by considering and rating already existent approaches in literature, discussing fundamentals, taking a look on the impact of limited resources and then choosing a most promising approach. The special case of Android based OppNets was explained and the promising approach was transformed in an own protocol proposal, which was then implemented and tested. The test showed, that the implementation is able to save resources, while still having a good delivery quota. However, by taking a look at possible security problems, TTL and connection behaviour, it was clarified, that there are still many possible improvements of the implementation and the implemented networking helper class for the network application.

6.2 Future Work

The field of practical used OppNets is still pretty new and therefore many OppNet related things exist, which can be developed in the future.

The more obvious things, are the improvements of the implemented routing and networking of the network application from A. Ippisch. Great scale tests could be done by distributing the network application, allowing us to determine the best parameters for the developed protocol with help of an integrated automated evaluation tool. The TTL and buffer handling can be adjusted to use routing decision history for improvements. A time stamp and cryptography system for routing information could be developed, which would allow us to propagate information about peers through the network and then only using the newest set of data with secured integrity. In general, more security functions should be implemented, which concern routing and OppNet related security weaknesses. Moreover, the already existing OppNet Android applications could be improved or new applications could be developed, which use the network application. This would create more use cases and may feature socialcast and multicast for community related content and also alternative addressing functions.

A more distinct future work, is the development of network application for other devices, like iPhones or desktop computers, working with each other. This would enhance the possible number of nodes and therefore making OppNets more reliable, while enhancing the usability. Another interesting feature would be a multihop bridge to the Internet, allowing multiple users to route via their peers to a hotspot with an Internet connection and use this Internet connection for themselves.

Bibliography

[BCP07] C. Boldrini, M. Conti, and A. Passarella. Impact of social mobility on routing protocols for opportunistic networks. *First IEEE WoWMoM Workshop on Autonomic and Opportunistic networking (AOC 2007)*, 2007.

[BH13] Greg Bigwood and Tristan Henderson. Incentive-Aware opportunistic Network Routing. In *Routing in Opportunistic Networks*. Springer Science+Business Media, 2013.

[DH07] E.M. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing(MobiHoc '07)*, 2007.

[eri15] Ericsson Mobility Report. `http://www.ericsson.com/res/docs/2015/ericsson-mobility-report-june-2015.pdf`, June 2015. Last checked: April 14th 2016.

[Gao02] J.L. Gao. Analysis of energy consumption for ad hoc wireless sensor networks using a bit-meter-per-joule metric. Technical report, California Institute of Technology, 2002.

[HC07a] P. Hui and J. Crowcroft. Bubble rap: forwarding in small world dtns in every decreasing circles, 2007.

[HC07b] P. Hui and J. Crowcroft. How small labels create big improvements. *Perva-*

sive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on, pages 65–70, 2007.

[HCY08] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing(MobiHoc '08)*, pages 241–250, 2008.

[Ipp15] Andre Ippisch. A fully distributed multilayer Framework for Opportunistic Networks as an Android Application. Master's thesis, HHU Düsseldorf, 2015.

[JFP04] S. Jain, K. Fall, and R. Patra. Routing in delay tolerant networks. In *Proccedings of the 2004 conference on applications, technologies, architectures, and protocols for computer communications*, pages 145–158, 2004.

[Kas13] Michael Kassner. Cheat sheet: What you need to know about 802.11ac. `http://www.techrepublic.com/blog/data-center/cheat-sheet-what-you-need-to-know-about-80211ac/`, June 2013. Last checked: April 24th 2016.

[Kerng] Ari Keränenr. ONE-Simulator Homepage. `https://www.netlab.tkk.fi/tutkimus/dtn/theone/`, ongoing. Last checked: April 24th 2016.

[KOK09] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA, 2009. ICST.

[LDS03] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob Comput Commun Rev 7(3)*, pages 19–20, 2003.

[MM12] Waldir Moreira and Paulo Mendesr. Opportunistic routing based on daily

routines. *World of wireless, mobile and multimedia networks (WoWMoM)*, pages 1–6, 2012.

[MM13] Waldir Moreira and Paulo Mendes. Social-Aware Opportunistic Routing: The New Trend. In *Routing in Opportunistic Networks*. Springer Science+Business Media, 2013.

[RS+11] Injong Rhee, Minsu Shin, et al. On the levy-walk nature of human mobility. *IEEE/ACM Transactions on Networking (TON)*, pages 19:3:630–643, 2011.

[SPC05] T. Spyropoulos, K. Psounis, and C.Ragavendra. Spray and wait: efficient routing in intermittently connected networks. *ACMSIGCOMM workshop on delay tolerant networking (WDTN '05)*, pages 252–259, 2005.

[SPC07] T. Spyropoulos, K. Psounis, and C.Ragavendra. Efficient routing in intermittently connected mobile networks: the multiple-copy case. *ACM/IEEE Trans Netw*, pages 16:77–90, 2007.

[VB00] A. Vahdat and D. Becker. Epidemic routing for partitially connected ad hoc networks. Technical report, Duke University, 2000.

[VPI13] Anshul Verma, K.K. Pattanaik, and Aniket Ingavale. Context-based Routing protocols for OppNets. In *Routing in Opportunistic Networks*. Springer Science+Business Media, 2013.

[WB+14] S. Wang, A. Basalamah, et al. Link-Correlation-Aware Opportunistic Routing in Wireless Networks. *IEEE Transactions on Wireless Communications*, pages 14:1:47–56, 2014.

[WDA+13] I. Woungang, S.K. Dhurandher, A. Anpalagan, A.V. Vasilakos, et al., editors. *Routing in Opportunistic Networks*. Springer Science+Business Media, 2013. ISBN 978-1-4614-3513-6.

[ZAZ04] W. Zhao, M. Ammar, and E Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proceedings of the 5th*

ACM International Symposium on Mobile Ad Hoc Networking and Computing(MobiHoc '04), pages 187–198, 2004.

[ZR03] M. Zorzi and R. Rao. Geographic random forwarding (GeRaF) for ad hoc and sensor networks: multihop performance. In *IEEE Trans Mob Comput*, pages 2(4):337–348, 2003.

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, 29. Juni 2016 Jannik Leßenich

Diese CD enthält:

- eine *pdf*-Version der vorliegenden Bachelorarbeit
- die LaTeX- und Grafik-Quelldateien der vorliegenden Bachelorarbeit samt aller verwendeten Skripte
- die Quelldateien der im Rahmen der Bachelorarbeit modifizierten Software opptain
- die zur Auswertung verwendeten Log Dateien
- die Websites der verwendeten Internetquellen