Sebastian Schellenberg

Naming and Address Resolution in Heterogeneous Mobile Ad hoc Networks

Naming and Address Resolution in Heterogeneous Mobile Ad hoc Networks

Sebastian Schellenberg

Universitätsverlag Ilmenau
2016

Impressum

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über http://dnb.d-nb.de abrufbar.

Diese Arbeit hat der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität Ilmenau als Dissertation vorgelegen.

Tag der Einreichung:	2. April 2015
1. Gutachter:	Univ.-Prof. Dr. rer. nat. Jochen Seitz (Technische Universität Ilmenau)
2. Gutachter:	Univ.-Prof. Dr.-Ing. Andreas Mitschele-Thiel (Technische Universität Ilmenau)
3. Gutachter:	Prof. PhD. Mainak Chatterjee (University of Central Florida (UCF))
Tag der Verteidigung:	26. Oktober 2015

Technische Universität Ilmenau/Universitätsbibliothek
Universitätsverlag Ilmenau
Postfach 10 05 65
98684 Ilmenau
www.tu-ilmenau.de/universitaetsverlag

Herstellung und Auslieferung
Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
www.mv-verlag.de

ISBN 978-3-86360-129-4 (Druckausgabe)
URN urn:nbn:de:gbv:ilm1-2015000570

Titelfoto: photocase.com | AlexFlint

für meinen Vater

Abstract

This doctoral thesis deals with naming, addressing, and address resolution in heterogeneous networks. The use cases for the motivation of the topic are disaster scenarios like natural catastrophes or terror attacks. Such events could damage infrastructure in parts or completely, especially communication infrastructure. For enabling a stable communication nevertheless, which is important for both rescue teams and victims, so called Mobile Ad hoc Networks (MANETs) could be used. However, central entities should be eliminated broadly in such types of networks.

Therefore, the main focus of this thesis lies on the MANET-adapted substitution of the Domain Name System (DNS), which is widely used in the common infrastructure-based Internet. Additionally, the introduction of an identifying addressing layer and the need for a mapping of the introduced node identifications to frequently changing local addresses—mainly addresses of the Internet Protocol (IP)—is the main use case for the proposed framework.

The presented decentralized system is based on the underlying routing protocols. Due to the strict integration of the name resolution functionality in the methods of the routing, we were able to eliminate centralized entities completely. The solution is furthermore combined with an adaptive routing framework. This provides the possibility to change the routing protocol during runtime based on the scenario. Therefore, we can improve the performance of the routing and the name resolution at the same time.

We show the concept and the implementation of the framework in the Click Modular Router (Click). Afterwards, we evaluate the system and show the overall performance of the approach. The used metrics are the additional generated traffic, the delay until the name is resolved, and the success rate. For simulation, we used the Network Simulator 3 (ns-3) combined with the integrated Click implementations.

In the last part of the thesis, we show three possible extensions to our basic framework. The first one deals with the reduction of generated traffic through host name hashing. Additionally, we show how to realize a service discovery mechanism based on the proposed name resolution system. The last extension deals with the enhancement of a location-aware name resolution or service discovery, respectively.

Zusammenfassung

Die vorliegende Dissertation befasst sich mit der Adressierung und Adressauflösung in heterogenen Kommunikationsnetzen. Die Anwendungsfälle für die Motivation des Themas stellen Katastrophenszenarien wie Naturkatastrophen oder Terroranschläge dar. Nach solchen Ereignissen kann die Infrastruktur und speziell die Kommunikationsinfrastruktur teilweise oder ganz beschädigt sein. Um trotzdem eine funktionierende Kommunikation sicherzustellen, die sowohl für die Einsatzkräfte wie auch für die Opfer unerlässlich ist, können sogenannte Mobile Ad-hoc-Netze (MANET) zum Einsatz kommen. Die Benutzung dieser infrastrukturlosen Vernetzung verlangt aber die weitgehende Eliminierung von zentralen Einheiten.

Daraus resultierend befasst sich die Dissertationsschrift mit der Ersetzung des in Infrastrukturnetzen gebräuchlichen Domain Name Systems (DNS) durch eine MANET-taugliche Alternative. Zusätzlich wurde durch die Einführung einer den Knoten identifizierenden Adressschicht die Notwendigkeit zur Auflösung der Identitäten zu den lokalen, vom Internet Protokoll (IP) eingesetzten, Adressen gegeben.

Das in der Arbeit vorgestellte dezentrale System zur Namensauflösung basiert vollständig auf den eingesetzten Wegewahlprotokollen. Durch eine strikte Integration der Funktionalität der Namensauflösung in das Routing kann komplett auf zentrale Einheiten verzichtet werden. Die Lösung wird weiterhin kombiniert mit einem adaptiven Routingkonzept, welches es Knoten ermöglicht, während der Laufzeit von einem Routingprotokoll zum anderen zu wechseln. Dies bietet eine bessere Reaktionsmöglichkeit auf Änderungen des Szenarios und dadurch eine erhöhte Performanz für die Wegesuche und die Namensauflösung.

Der konzipierte und im Click Modular Router (Click) implementierte Ansatz wird weiterhin im Laufe der Arbeit evaluiert und auf die Performanz bezüglich des zusätzlich generierten Verkehrs, der Zeitverzögerung bis zur Auflösung eines Namens zu einer Adresse und der

Erfolgsrate getestet. Hierzu werden Simulationen mit dem Netzwerksimulator 3 (ns-3) durchgeführt.

Im letzten Teil der Arbeit werden drei mögliche Erweiterungen des präsentierten Systems gezeigt. Die erste zielt auf die Verbesserung der Performanz durch Hashing des aufzulösenden Namens. Die zweite Erweiterung beschäftigt sich mit der Realisierung eines Dienstlokalisierungsmechanismus, basierend auf dem System zur Namensauflösung. Zusätzlich wird die Erweiterung hin zu einer Standort-sensitiven Namensauflösung bzw. Dienstlokalisierung aufgezeigt.

Danksagung

Nach vier Jahren intensiver Arbeit an meiner Dissertation möchte ich mich bei all jenen bedanken, die mich beim Erstellen dieser Arbeit auf unterschiedliche Weise inspiriert und unterstützt haben.

In erster Linie möchte ich mich bei Professor Jochen Seitz, für die hervorragende Betreuung und die Möglichkeit als wissenschaftlicher Mitarbeiter im Fachgebiet Kommunikationsnetze arbeiten zu können, bedanken. Er hat mir in regelmäßigen, konstruktiven Gesprächen immer das Gefühl vermittelt, auf dem richtigen Weg zu sein und mich an keiner Stelle im Gesamtprozess am letztendlichen Gelingen zweifeln lassen. Während meiner Forschung ließ er mir immer größtmöglichen Freiraum in der Entwicklung meiner thematischen Schwerpunkte.

Mein Dank gilt auch allen Mitarbeitern des Fachgebiets Kommunikationsnetze für drei wunderschöne Jahre in konstruktiver und angenehmer Arbeitsatmosphäre. Besonders möchte ich die fachliche Zusammenarbeit mit Thomas Finke und Silvia Krug hervorheben. Durch fachliche Diskussionen und gegenseitige Unterstützung konnte eine Vielzahl gemeinsamer Projekte und Veröffentlichungen entstehen. Auch die entspannte und freundschaftliche Atmosphäre im Büro mit Dominik Schulz sowie die täglichen Kaffeerunden mit Markus Hager haben stets für gute Stimmung gesorgt.

Einen wichtigen Anteil an der Entstehung der Dissertation trägt Professor Andreas Mitschele-Thiel, der als Initiator und Leiter der Graduiertenschule MOBICOM für ein perfektes Umfeld sorgte. Hervorzuheben ist nicht allein die hervorragende technische und finanzielle Unterstützung durch die Graduiertenschule, sondern im Speziellen das strukturierte Promotionsprogramm, welches eine intensive Fokussierung auf das eigene Thema sowie den regelmäßigen themenübergreifenden Austausch mit Doktoranden angrenzender Fachbereiche ermöglichte. In diesem Rahmen hatte ich zudem die Chance, einen dreimonatigen Forschungsaufenthalt an der University of Central Florida durchzuführen.

Professor Mainak Chatterjee möchte ich für seine Bereitschaft die externe Betreuung zu übernehmen danken. Die umfangreiche Unterstützung in Vorbereitung und Durchführung des Auslandsaufenthalts in den USA, der herzliche und offene Umgang sowie eine Reihe interessanter Gespräche mit ihm und seinen Team sind besonders positiv zu erwähnen. Gerade in dieser Zeit war es mir möglich, mich voll und ganz auf das Schreiben der Dissertation zu konzentrieren.

Ein großer Dank gilt meiner Familie und meinen Freunden für die Unterstütung, den nötigen Rückhalt und gelegentliche Nachsicht. Ein besonderer Dank gilt meiner Frau Jennifer dafür, dass sie mir stets den Rücken freigehalten und das Redigieren der Arbeit übernommen hat.

Contents

1 Introduction

In this chapter, we will give an introduction to the work. We show the motivation for the topic including our use case scenario as well as the general goals of the work. Finally, we show an overview of the structure of the thesis.

1.1 Motivation and Scenario

In the recent decade, many disasters happened like the flood and nuclear catastrophe in Japan in 2011, hurricane Katrina in the United States in 2005, or several terror attacks. Such catastrophes have horrible effects on the life and property of the victims. A working communication could support rescuers to save hundreds of lives and aid people more effectively. Unfortunately, such occurrences often damage the infrastructure of communication systems and leave rescuer and victims without the possibility to communicate with each other over long distances. A mobile radio system for instance could be partially destroyed or fully out of order.

This doctoral thesis was created in the context of the International Graduate School on Mobile Communications (MOBICOM) [1]. This graduate school deals with (technical) communication in disaster scenarios, including three clusters: self-organized robust infrastructure networks of the fourth generation (4G), cognitive radio systems, and robust infrastructure-less communication. This thesis is part of the efforts in the third mentioned group.

One way to reestablish a communication after the infrastructure backbone was damaged is using the technology of a Mobile Ad hoc Network (MANET). With such a networking technology, the users are able to deploy a network without the help of any backbone. The devices themselves build the backbone and forward upcoming data on the multi-hop principle [2].

Especially for disasters, this could be the only technology to quickly establish a connection at all. Unfortunately, this networking approach comes up with several challenges. As the Internet was built on the idea of a static and centralized use case, most of the mechanisms and functions no longer work efficiently.

An example challenge and the one we dealt with in this thesis is the addressing and address resolution problem. As the founders of the Internet did not imagine moving nodes at all, they gave the commonly used address at the Networking Layer, the Internet Protocol (IP) address, two tasks at a time, i.e., localization and identification [3]. The appearance of mobile devices such as mobile phones, Personal Digital Assistants (PDAs), notebooks, or other devices rebutted the assumption of non-moving nodes. If a node changes its location but wants to keep its identity, it notwithstanding has to accept a new address. The problem occurs that nodes probably get a new IP address with every network change they do. If for instance a node is using an Internet connection in some infrastructure network like Long Term Evolution (LTE) but has then the possibility to change to a local connection at home, it definitely has to change the IP address. This affects the currently used connections and the possibility to be found again for a connection establishment in the future.

Several approaches to deal with these problems were proposed but mainly for the usual use case of an infrastructure network with centralized entities solving this problem. As we need to replace such entities with decentralized approaches, new solutions need to be considered.

In disaster scenarios, we assume a highly mobile scenario with nodes moving around and frequently changing the network and therefore the IP addresses on Layer 3. Unmanned Airborne Vehicles (UAVs) could be used for example to locate people or establish any MANET connection between the participating nodes [4].

This results in a heterogeneous network built by victims, rescue teams, and UAVs. Figure 1.1 shows an example scenario with three subnetworks. The left one represents victims with their smartphones trying to call for help. The central network represents a partially working infrastructure-based network with a central base station and some mobile nodes. The right network coordinates the rescue teams.

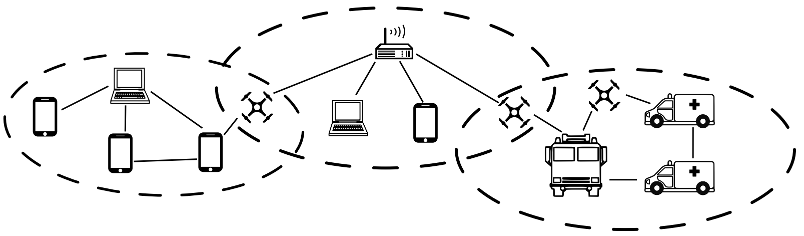

Figure 1.1: Possible network topology in disaster scenarios with victims, rescue teams, and unmanned air vehicles

Some quickly moving nodes, like UAVs, could fly around and frequently enter and leave networks. They could also participate in several networks at once, serving as gateways to establish communication between spatially separated subnets. Furthermore, some networks could merge or split over time.

This leads to the problem that some nodes are addressed by several IP addresses at the same time or with changing IP addresses over time. Nodes not equipped with our system consequently might think that former communication partners became unreachable, while they are still present just with a new local address.

To ensure a stable communication and addressing of the rescue teams member devices or the victims, the usual IP addresses alone cannot be used. This leads to our main goals of this thesis, which we present in the next section.

1.2 Goals of the Work

To overcome the problems described in the previous section, we set the following goals to be reached by the thesis.

Firstly, we want to create an intelligent addressing scheme providing consistent naming and addressing for the MANET scenario. It has to be determined, whether it is possible to change the properties of the current IP addresses or whether we have to introduce new abstract identities for every node. A node's identity address should remain the same no matter where the node is located and how often it switches the subnetwork.

Secondly, changes of the local address due to network changes of nodes should be available everywhere and on time. Therefore, a name resolution and node discovery approach adapted to the MANET scenario needs to be developed to map the static identities to the frequently changing local addresses. Both, the addressing scheme and the name resolution approach, should be appropriate to the disaster scenario and as resource-aware as possible [5].

1.3 Organization of the Thesis

The thesis is organized as follows. In Chapter 2, we discuss the term of addressing in different layers and the mapping from one address to another. A special attention is given to special challenges for name resolution in MANETs. The following Chapter 3 discusses the state of the art for name resolution approaches and the need for a further approach. Chapter 4 deals with routing in MANETs as our developed name mapping approach is based on routing protocols. In Chapter 5, the basic concept for realizing name resolution over an adaptive routing framework is shown and explained. Afterwards, the developed system is simulated and evaluated in Chapter 6 including a comparison to selected state of the art approaches. In Chapter 7, several extensions and enhancements to the presented framework are shown and simulated. This includes the discussion of hostname hashing to reduce traffic, a service discovery mechanism based on the name resolution framework, and the introduction of location-awareness. The thesis is finally concluded in Chapter 8 and future work is shown.

2 Fundamentals on Addressing and Name Resolution

This chapter deals with the terms of addressing and address resolution. We introduce concepts for addressing in layered networks and show special challenges for the scenario of a MANET.

2.1 Addresses and Address Resolution in Layer-Structured Communication Systems

Communication systems are commonly described by so called layers. The tasks of a communication system is subdivided into several levels while every level provides its own specific functionality. This gives the developer the possibility to concentrate on only one service in one layer without caring about other services provided by an upper or lower layer. The most common layer structure is the Open Systems Interconnection (OSI) model [6] standardized by the International Organization for Standardization (ISO). Within this model, seven layers are defined from the Physical Layer, dealing with the transmission of signals, up to the Application Layer, providing services and applications for the user. Every layer makes use of a service provided by the layer below.

Most of the communication today is based on the Internet Protocol [3]. For such systems, a new layer model was designed based on the OSI model. The three upper layers and the two lower were combined to one layer, respectively. This results in the 4-layer-model for Internet communication as defined in Request For Comments (RFC) 1122 [7], utilizing mainly the Transmission Control Protocol (TCP) at the Transport Layer—also known as the TCP/IP model (cf. Figure 2.1).

In every layer, communication partners are somehow addressed. In the second layer, Media Access Control (MAC) addresses are used to identify different interfaces. This is either done by the customarily 48 bit address or the new extended 64 bit address [8]. At Layer 3, the widely used Internet Protocol (IP) uses the corresponding IP addresses, which have a length of either 32 bit in version four of the IP (Internet Protocol version 4 (IPv4) [3]) or 128 bit in the current version six (Internet Protocol version 6 (IPv6) [9]). In the Application Layer, programs are addressed by human readable names like Uniform Resource Identifiers (URIs) [10]. We discuss further details on the process of address assignments later in Section 2.3.

At every layer, the communication partners use the regarding addresses to communicate. If user A wants to write an e-mail to user B, which would be an application on the Application Layer, it addresses user B with its Application Layer e-mail address and sends along its own e-mail address for replies. If a node wants to send an IP packet to a sink, it addresses the destination with its IP address and sends along its own one.

However, every communication is finally done at the Physical Layer. That means that addresses from the upper layers have to be mapped to addresses from lower layers. E.g., hostnames at Layer 7 are mapped to IP address before IP packets can be sent. If the user gives the browser the command to access a website like www.tu-ilmenau.de, the browser needs the IP address of the server hosting this homepage before it can establish a connection. This is commonly done by the Domain Name System (DNS) [11] and is further described in Section 2.4.

Afterwards, the next hop to reach this destination has to be found by some routing protocol. If the IP address of the next hop is known, the interface of this node has do be discovered. The IP address of the node is therefore resolved to the regarding MAC address of the interface. This is done by the Address Resolution Protocol (ARP) [12] in IPv4 networks or the Neighbor Discovery Protocol (NDP) [13] in IPv6 networks.

In ARP for instance, a node searching for a MAC address of a communication partner where the IP address is known, broadcasts a packet to request this mapping to all direct neighbors. The node that owns this IP replies with its MAC. Finally, the message can be sent to the next hop. This is done hop by hop until the packet, in this case a request for a webpage, reaches the destination.

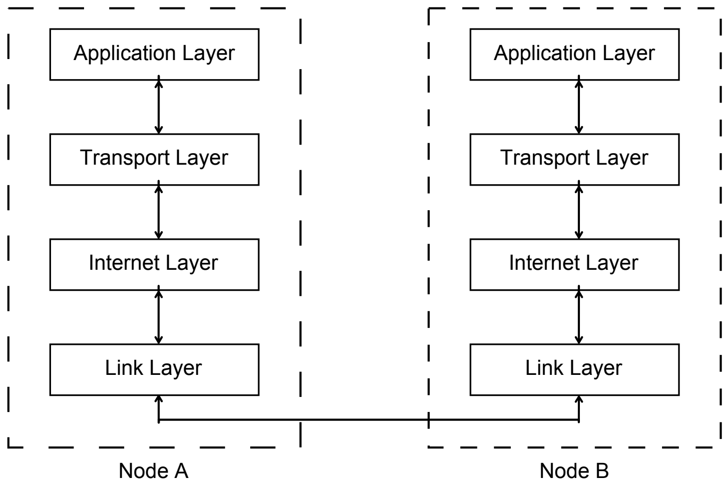

Figure 2.1: Layered Internet model with two nodes

The more layers we assume, the more address resolution we need. If we assume an Internet model as shown in Figure 2.1, we have four layers and therefore four addresses, i.e., hostnames for the Application Layer, ports for the Transport Layer, IP addresses for the Internet Layer, and MAC addresses for the Link Layer, which results in three address resolution processes.

2.2 Term Definitions

In this section, we want to define some important terms used throughout this thesis.

An **address** is a number or string to identify a Service Access Point (SAP) in any layer considering a layered model.

If we talk about which addresses are used in which layer, we are going to use the term **addressing**. The addressing for the Network Layer for instance consists of local IP addresses defined by the used Internet Protocol.

The process of allocating an address to a SAP in one node or choosing the address by the node itself, respectively, is called **address assignment** in this thesis. Selected methods for address assignments are shown in Section 2.3.

The general term to resolve an address at Layer N to an address of the underlying Layer N-1 is **address resolution**. We call Application Layer addresses in this thesis **hostnames** or just **names**. The main focus of the thesis lies on the resolution of such names to Layer 3 addresses. For a better understanding and to distinguish between the Layer 7 and Layer 3 addresses, we consequently call IP addresses just **addresses** or **local addresses**. In this context, we talk about **name resolution**, meaning that the name is resolved to the (local) address. However, generally hostnames and IP addresses both are just addresses located in different layers. To avoid confusion, we use both terms address resolution and name resolution interchangeable in this thesis.

This term definition fits to the guidelines given by the RFC describing the Internet Protocol. Here, it is said: ”A distinction is made between names, addresses, and routes. A name indicates what we seek. An address indicates where it is. A route indicates how to get there.” [3]. This is of course limited to the Network Layer view and ignores addresses at other layers. However, as we also use the Internet layer model we use the terms **names** for Application Layer addresses and **address** for a Network Layer address as well as **routes** for the path through an IP layer network.

2.3 Address Assignment

In the previous section, the terms *addresses* and *address resolution* in layered communication systems were introduced. This section deals with the question how those addresses are assigned to the instances of the layers in every node. The presented examples are shown in the context of the TCP/IP layer model with four layers.

In the Link Layer, commonly MAC addresses are used (cf. Section 2.1). There is no protocol to actually assign MAC addresses to interfaces. Every interface gets a previously selected MAC address burned-in by the producer of the device. Producers get the name space from the Institute of Electrical and Electronics Engineers (IEEE) Registration Authority. Usually, a MAC address has a length of 48 bit where the first 24 bit identify the producer and the last 24 bit the produced device. Furthermore, every MAC address is globally unique. Therefore, every device is identified by a unique address and address conflicts are prevented. However, it is possible for the user to change the burned-in address manually. With the coming of more and more interfaces in small devices like sensors, the previously intended name space of 48 bit is meanwhile extended to 64 bit [8] and even considered to be 128 bit in the near future.

In the Network Layer, address assignment is done by external protocols or firmly assigned by the users itself. Every node participating in the Internet needs a globally routable IP address. For normal users, this address is assigned by the provider. Telecommunication companies have to get an address space from the Internet Assigned Numbers Authority (IANA) [14]. The 32 bit long IPv4 addresses are usually divided into a part defining the network and the other part defining the host. The point where the address is split is given by the so called subnet mask. Due to the lack of available address space in IPv4, addresses were extended to 128 bit in IPv6.

For private networks, link local address spaces are defined. In our scenarios, we let the nodes use such link local addresses as we consider infrastructure-less networks without providers. In such networks, every node has to choose an address from the address space, which is not used by any other node. The easiest way to do so is assigning IP addresses by a centralized entity that ensures that no address is used twice. An example for such a protocol is the Dynamic Host Configuration Protocol (DHCP) [15]. Therein, a centralized server assigns the IP address with the subnet mask to the nodes as well as a possible gateway and DNS server address.

In MANETs, centralized entities are generally not applicable. Therefore, the usage of a DHCP server is not advisable. Several papers were published dealing with decentralized name assignment. A literature survey on local addressing in MANETs was done by P. Roensch during his student paper (Hauptseminar) [16] supervised by the author of this thesis. Furthermore, a good survey on address assignment protocols was given by Bernardos et al. [17]. The briefly presented protocols in this subsection should serve as a guideline which basic mechanisms are used in literature, without claim of completeness.

In the **IETF zeroconf** approach [18], a new member of the network selects a link local IP address randomly. Thereafter, it asks the network whether another node already uses this IP address. If no other node complains, it keeps the address and can start any communication. Otherwise, it selects another random IP address and asks again until a valid one is selected. In the **MANETconf approach** [19], a new node, here called the *requester*, asks a member, called the *initiator*, to find a valid address. Every node runs a table about possible IP addresses and already allocated ones. The initiator asks all nodes about a selected new address for the requester. If no one complains, the allocation is distributed and the requester can use the new IP. In the **buddy** protocol [20], every node has a range of possible IP addresses and uses one of them for own communication. If a new node wants to participate, it contacts an old node which transfers the half of its address range to the new one. The new node now uses one of the addresses in its new allocated address range and keeps the rest unused. This procedure is repeated for every new member node, until only the self-used address is left. Such a node cannot accept new members and a new node has to find another buddy node. Other notable approaches are are the **Prophet** [21] or the **OASIS** [22] protocol.

Throughout this thesis, we do not consider the local address assignment. We assume, that this necessary process is defined by the local network which might stay outside our control. In our simulations, we assume that every node has a previously assigned address in advance.

In the Transport Layer, sockets are addressed with ports. There are well-known ports for widely used applications like port 25 for Simple Mail Transfer Protocol (SMTP) or port 80 for Hypertext Transfer Protocol (HTTP) TCP connections. Other ports than the well-known registered ports have to be negotiated before communication. The in-node mechanism to allocate incoming traffic to the correct port decides based on the couple of the port and the Network Layer IP address.

With the couple of port and IP address, any resource in the Internet is accessible. Therefore, there is no further need for addresses, at least for the inter-machine communication. However, for humans it is hard to find out or remember IPs and ports, respectively. For better human understanding, human readable addresses are used at the Application Layer. Users type in www.tu-ilmenau.de in the browser instead of an IP address and e-mail programs also use addresses like first.last@tu-ilmenau.de. In this context, addresses at Layer 7 are called names. In RFC 3986 names are standardized to URIs [10]. The most common used URI subset is the Uniform Resource Locator (URL) [23]. The basic form of a URL is built on two parts, the *scheme* defining the protocol and the *scheme-specific-part* identifying the resource. Examples for schemes are *http* for using HTTP or *mailto* for e-mail usage. In the scheme-specific-part, information like a login, the host as an IP address, or the Transport Layer port is delivered. Every application or the developer of the application, respectively, is free in selecting. However, a URL has to be registered in the DNS to be resolved to a machine-usable Network Layer address (cf. Section 2.4).

2.4 The Domain Name System (DNS)

The main focus of this thesis lies on name resolution in MANETs. This section deals with the well-known and widely used Domain Name System providing name resolution in the infrastructure-based Internet. As described in the last section, users usually use human readable names to access Application Layer services. Those names have to be mapped to the regarding local IP addresses, which is the main task of DNS. The protocol is specified in the RFCs 1034 [11] and 1035 [24].

DNS is a centralized and hierarchical directory service. Centralized means that requesting nodes send upcoming name requests to a central entity, which replies with the IP address. However, there are thousands of such name servers dealing with separated name spaces but structured in a hierarchy. The Domain Name System uses a tree data structure where each nodes deals with a DNS zone.

A zone consists of one or several domains and sub-domains. The highest domain is the root domain, followed by Top Level Domains like the *.de* domain, followed by Second Level Domains and so forth. Each leaf consists of several or none resource records, holding information to a hostname such as the IP address. Resource records have a standardized format specified in RFC 1035 [24].

The name resolution process itself works as follows. If the client wants to resolve *www.tu-ilmenau.de*, it first requests the root domain name server for the address of the *.de* Top Level Domain name server. Then it requests the found server for the address of the *tu-ilmenau* Second Level Domain name server and afterwards for the *www* domain. It has to be guaranteed that all servers are available and have all relevant information up-to-date.

It should be mentioned that separated networks can run independent name servers for own purpose, e.g., in corporate networks. The address of the internal server has to be known for any participating node, for example by transmitting it via DHCP.

Although the DNS system is widely used and commonly accepted, we cannot rely on this system in MANETs. In highly mobile scenarios without a steady infrastructure, DNS servers would be a single point of failure. If a node hosting the DNS service dies or leaves the network unexpectedly, name resolution could be impossible for the remaining nodes. Therefore, we have to find decentralized solutions practicable for our scenario.

2.5 Addressing Scheme for Disaster Scenario

In the last section, we presented the DNS and showed why it is not applicable for the MANET scenario. To come up with an appropriate solution, we have to consider addressing, address assignment, and address resolution as a whole. In this section, we discuss possible solutions and propose an addressing scheme that is applicable for the disaster scenario using MANETs.

We assume a network stack similar to the TCP/IP protocol stack. There are lots of new proposals published changing the usual stack structure. The Forwarding on Gates (FOG) approach [25] for instance goes completely away from layered communication structures and uses gates instead. However, the communication in our MOBICOM system should be as compatible as possible to non-MOBICOM nodes. As we want to fix existing broken infrastructure networks and furthermore want to contain victims and their devices in our communication system, we decided to base our framework on the commonly used TCP/IP stack. Victims usually use conventional devices like smartphones or notebooks and therefore use custom routing and networking protocols.

If we analyze the local addressing, assuming IP addresses, we have more or less no influence on the address assignment. If the MANET is not created by our devices and software, we have to take the addresses we get. The addressing could either be done by some kind of a DHCP server or by any MANET-adapted addressing protocols as presented in Section 2.3.

However, we can determine that the local addressing can hardly be influenced regarding one subnet. If a node leaves one network and joins another, it is likely that it gets a new local address and is therefore no longer reachable under the old one. The reason is the dual function of the IP address serving as identification and localization of the node (cf. Section 1.1). To reach a node based on a static address, the identification of the nodes must be decoupled from the IP address. This is commonly known as the *Identification/Localization Split* in the literature. An example approach is the Locator/Identifier Separation Protocol (LISP) [26].

Another approach to overcome the problem is the Mobile IP protocol, which is available for IPv4 [27] and IPv6 [28]. This protocol was designed to allow nodes communicating based on a static *home address* without caring about subnetwork changes. Local addresses valid in foreign networks are called *foreign addresses*. A so called *Home Agent* monitors which foreign address a mobile node currently has. Packets arriving in the home network are tunneled by the Home Agent, which is basically a router, to the foreign network. The approach works in infrastructure networks and provides a stable communication based on home addresses. However, in MANETs this approach has to fail because the several Agents again act as a single point of failure. If the agent fails, the whole Mobile IP communication fails. Furthermore, we assume all nodes to be highly mobile. This means that home agents could leave their home networks, too. Therefore, we do not consider this approach to solve the problem mentioned in the introduction (cf. Section 1.2).

Another approach is the so called Host Identity Protocol (HIP) [29]. This protocol introduces a new sublayer between Layer 3 and Layer 4, sometimes called Layer 3.5. The task of this intermediate layer is to decouple the identification and localization functionality of the IP address by introducing new addresses called Host Identity Tags (HITs). The HIT acts as the node's identity and remains static. The Transport Layer establishes communication based on the couple of a HIT and a port instead of an IP address and a port. The local packet delivery is still done by IP at the Network Layer.

Figure 2.2 shows the stack changes if using HIP.

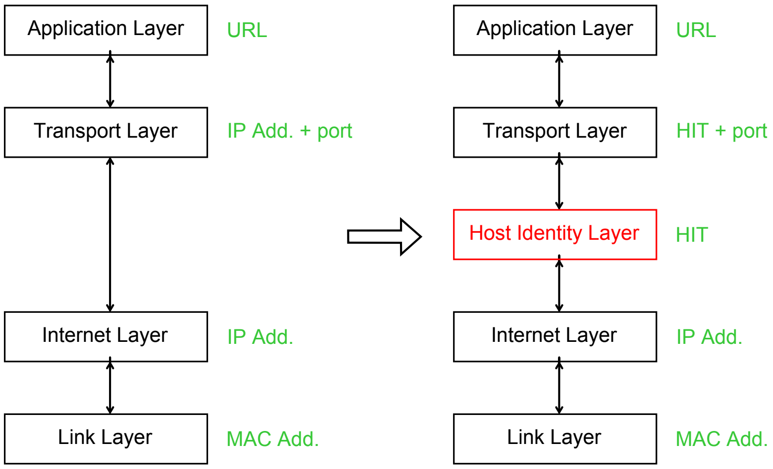

Figure 2.2: Changes in the Internet layer model if HIP is used

The intermediate Host Identity Layer knows the regarding IP address for a given HIT and forwards the packets to the Network Layer with the right IP address. This absolutely solves the problems described in Section 1.2. However, HIP is reliant upon an external address resolution protocol mapping the HITs to changing IP addresses. In infrastructure networks, this can easily be done by the Domain Name System [30] with minor changes on the protocol.

As pointed out in Section 2.4, the centralized DNS is not applicable for MANETs. Therefore, the proposed mechanisms lead to a so called chicken-egg-problem. Introducing a new identifying layer with (almost) stable identities for every node solves the problem of changing IP addresses at Layer 3. But it just outsources the problem to a reliable and MANET-aware name resolution mechanism. We can say that every ID/locator split must lead to some kind of name resolution.

However, if we want to use the current protocol stack with the Internet protocol, we must deal with some kind of ID/locator split to have identity-based communication. We consider our identities as located in the Application Layer. We do not want to destroy the common stack and therefore do not use approaches like the HIP. The assignment of the identities to the nodes as well as consistency and security problems are not considered in this thesis and part of possible future work (cf. Section 8.2). In this thesis, we do not specify how the creation of the identity layer is done but point out that it has to be done, at least for the nodes in the MOBICOM system.

In the following chapters, we concentrate on the name resolution problem. We show the state of the art and our own concept including a validation.

2.6 Relation between Names and Local Addresses

Until now, we proposed the addressing scheme containing an identifying layer in the Application Layer and some local Network Layer addressing, especially IP addressing. This section deals with the possible relations between the Identification (ID) layer and the local addresses.

The main usage of the scheme is the introduction of identifying names being unique for every node. Therewith, every node can be identified or just found even if the IP address changes frequently. However, we assume some nodes to be multihomed. E.g., multicopter used in our MOBICOM system could have several interfaces to couple several networks. Therefore, they are reachable under several local addresses that could be heterogeneous.

If we reflect upon the relation the other way around, one IP address should be associated to only one unique ID. The relation of IDs to local addresses is therefore 1:m. It can be discussed, whether different logical (overlay) networks could result in different IDs per node and, therefore, per IP address. For reasons of simplicity, we only assume one ID overlay in our scenario for our MOBICOM system.

Another possible use case is service discovery (cf. Section 7.2). As services are also identified by names we can use the naming layer for this purpose, too. Nodes hosting a defined service could be multihomed again. They can decide, whether they want to provide that service for all subnetworks or only for selected ones. Furthermore, the same service identified by the same service ID could be provided by several nodes simultaneously. Therefore, the relation between service names and local IP addresses is n:m. Based on that, further mechanisms can be realized such as load balancing. For the user, it is not relevant, which node is the service provider. If we assume a technical service like an Internet gateway, it is just important to find a service provider at all. For physical services like medical aid, load balancing or spatial information are more important (cf. Section 7.3).

We also proposed further possible use cases for our framework in some conference papers. Examples are the group member management in reliable multicast [31, 32] and the underlay model for Delay Tolerant Networking (DTN) [33]. However, this is not part of this doctoral thesis.

2.7 Discussion

To conclude this section, we want to figure out two major points.

Firstly, we want to utilize the widely used TCP/IP stack in our system to be as compatible as possible to other devices. Therefore, we have to deal with the double-functionality-problem of IP addresses serving as a locator and identification at the same time. We cannot assume that we have influence on local addressing in every subnetwork. Hence, an ID/locator split is advisable. Stack changing approaches to overcome this problem are not factored in because of the compatibility issue.

Secondly, decoupling the identification of the nodes from the Network Layer address to an upper layer always needs a working mapping algorithm to resolve the identities to the IP addresses. Centralized approaches like DNS or other solutions like Mobile IP are not applicable in MANETs. This results in the need for an adapted name resolution protocol as vehicle for realizing a ID/locater split.

Therefore, we want to discuss the state of the art regarding name resolution approaches in MANETs in the next chapter.

3 State of the Art for Name Resolution Protocols in MANETs

This chapter deals with the related work regarding adapted name resolution in MANETs. We clustered the state of the art approaches in three groups: the centralized but modified, the multicast-based, and the routing-based solutions.

3.1 Centralized but Modified Solutions

The obvious first starting point to deal with the described problems is to extend the already existing DNS system. By putting some efforts into extensions to the basic system, some problems could be diluted. In infrastructure networks, DNS servers are located on single nodes at a fixed place in the Internet. Those servers are then grouped hierarchically. By taking away the hierarchy, one server has to deal with all requests. If this node dies, no name can be resolved at all.

More robustness can be added to the system with more redundancy. Instead of having the information at only one point, several DNS servers could share the task. This would lead to a high effort to keep all servers updated and consistent, but it would relieve problems. However, if all redundant DNS servers are located in one part of the network and the network splits in such way that all DNS servers are in the one half and none in the second half, one part is again without a possibility to resolve names. Especially in highly mobile MANETs, such scenarios could be assumed.

An intelligent service placement could be an effective countermeasure. Redundant DNS servers would now be placed on nodes selected through an intelligent network planning in a way that no network part is left without. Possible approaches for intelligent service placement were given by Wang et al. [34] and Derhab et al. [35].

However, all solutions can not eliminate single points of failure definitely. Assuming highly mobile nodes with the possibility to move with great speed, it might not be possible to copy services in time. Furthermore, it is still possible that nodes fail from one moment to another without previous signs.

The work proposed by Ahn et al. [36] gives an example for a modified but centralized DNS approach for MANETs. In this approach, the network dynamically decided on new DNS servers if the network splits and one part would be without DNS support afterwards. Whenever a node recognizes that no DNS server is present, it turns into a server by its own, announces that in the network via broadcast, and receives register messages by all participating nodes. By appearing high traffic and frequently splitting and merging networks, we assume a high overhead for node selection and name binding registration. Furthermore, we assume a high delay during the process of setting up the new server if a node requests a name mapping in the meantime. However, the name resolution process itself is less traffic consuming and the delay is only depending on the number of hops to the DNS server. The presented evaluation shows that their protocol outperforms the Multicast Domain Name System (mDNS) approach (cf. Section 3.2). However, they did not describe how the movement scenario was and whether the network was fully connected all the time.

3.2 Multicast-based Name Resolution

The approaches shown in Section 3.1 are based on unicast communication. Nodes register name mappings to a central DNS server and request for resolution with a unicast packet. Challenges coming with such approaches are: to have at least one server in each subnet, to have a consistent knowledge in each server, and to know the IP address and route to the central entity.

If we want to avoid centralized entities, we have to distribute the knowledge and use distributed communications. This leads to either broadcast or multicast message sending. The straightforward approach is simple flooding. We just flood a request into the entire network and wait for a reply. At least the node hosting the name itself is able to reply. The reply could be sent unicast or again broadcast. We can limit the traffic by just letting a group of nodes participating in the name resolution process. This leads to multicast communication for the request and optionally the reply. Two approaches working with this basic approach are the mDNS [37] approach and the Ad-hoc Name Service (ANS) [38] system.

The mDNS protocol was designed by the Internet Engineering Task Force (IETF). A multicast group is responsible for name resolution. If a node wants to resolve a hostname, it sends a message to the multicast group. If one node is aware of the mapping, it replies with a multicast message to the whole group again. Assuming all nodes to participate in the name resolution process, this protocol leads to simple flooding for request and reply. However, in any case this protocol leads to an excessive traffic generation. In case of a disaster scenario, we want to keep the network as resource-aware as possible but reliable at the same time. The mDNS approach gives reliability but at the cost of an overloaded network.

Furthermore, a reliable multicast communication is a difficult task by its own [31, 32, 39]. Nodes could join and leave the multicast group dealing with name resolution frequently. A requesting node might not be aware of all multicast group members. New members can join the group in a foreign network, which leads to an incomplete picture. The message delivery itself cannot be guaranteed, too. Especially if message ferries are used, a request or a reply might not reach the destination. Multicast protocols using a tree structure could fail in MANETs if parts of the tree fail.

However, assuming a working mDNS, the protocol can easily be extended to support service discovery (cf. Section 7.2). An example is the DNS Service Discovery (DNS-SD) protocol [40].

The ANS approach is especially designed for IPv6 networks and uses the same basic mechanism as mDNS. It is based on the Auto Configuration mechanism provided by IPv6. The ANS system consists of so called resolvers and responders. The ANS resolver process can be started on every node that wants to resolve a name. It afterwards sends a request to a multicast group. Each node in this group hosts an ANS responder to reply on requests. Each node that wants to run a responder has to join the group. Therefore, the same multicast handling problems as in mDNS apply.

The reply in ANS is sent via unicast to the requesting node. The advantage is that this limits the traffic consumption during the replying process. The drawback is the lack of distributed knowledge in the multicast group. One ANS responder is always responsible for a subset of name mappings. In mDNS, the reply is sent multicast to the whole group. This leads to more traffic but gives the whole group knowledge about a mapping for future requests.

For sparse requests, sending the reply unicast might perform best. For frequent requests in the network, flooding the reply prevents future flooding of requests. In order to optimize the generated traffic, a hybrid approach might perform best; but this is not yet proposed in literature.

3.3 Routing-based Name Resolution

The approaches presented in Sections 3.1 and 3.2 are all located in the Application Layer. The task of name resolution is seen as a classical task for an application as it is seen in the standard DNS protocol. Hostnames are addresses in the Application Layer and they have to be mapped to IP address located in the Network Layer before a connection can be established.

An approach to come away from this view is transferring the task to the Network Layer instead. The ARP that resolves Internet Layer addresses to MAC addresses is also located in the Link Layer. Consequently, using Network Layer mechanisms to resolve hostnames can also work.

Engelstad et al. first came up with this idea [41]. They used the routing, which is located in the Network Layer and responsible for finding a route to an IP address, to transport DNS requests. The approach is motivated by the fact that the broadcast or multicast approach could need an extra route discovery process after the name resolution process if the backward route is not discovered by accident, too.

Engelstad et al. took DNS request called name request piggybacked with the route requests of the Ad-hoc On-demand Distance Vector (AODV) protocol. If this double request reached the requested node, it responded with a name reply taken piggyback with the AODV route reply. We will take over the main principle of this approach in the reactive scheme of our name resolution framework presented in Chapter 5.

However, the approach has some shortcomings. First of all, Engelstad et al. are limited to the reactive AODV protocol in their system. As presented in Section 4.5, reactive routing is not always best performing in MANETs. For certain scenarios, a reactive route discovery might perform best and AODV is the most developed and evaluated example. However, the approach lacks flexibility to react on scenario changes. Engelstad et al. presented, how their system interacts with infrastructure-based networks using normal DNS [42]. However, proactive networks—e.g., based on the Optimized Link State Routing (OLSR) protocol—and inter-domain name resolution with heterogeneous networks is not covered.

Furthermore, we identified some other disadvantages. Engelstad et al. use the reactive mechanisms of AODV to integrate name resolution and it definitely works. However, the approach does not always follow the AODV main ideas in every aspect. The AODV route discovery is not necessarily always performed by the destination itself. If an intermediate node has some route knowledge, it can answer requests immediately. This decreases the traffic and delay at the same time. Using that idea for the name resolution, too, can increase the performance of the approach (cf. Subsection 5.5.4 and Section 6.3).

The Lightweight Underlay Network Ad hoc Routing (LUNAR) approach by Jelger et al. even goes one step further by merging not only routing and name resolution but also the mapping of IP addresses to MAC addresses [43]. The resulting new layer builds an underlay for the Internet Layer. This paper integrates the name resolution functionality into the OSI Layer 2.5 including the routing. However, this makes the system totally incompatible to the standard stack. As one of our design criteria was to be as compatible as possible, to integrate victims and rescue teams into our network without bigger problems, we abdicate to follow such ”new Internet” approaches.

3.4 Discussion

In this chapter, we showed related work regarding name resolution approaches adapted to MANET scenarios. As the most promising cluster, we identify the routing-based solution for various reasons.

Firstly, routing is needed anyhow in any communication system at the Network Layer if we do not want to use simple flooding. By using routing for name resolution, we can benefit from mechanisms which are available and used anyhow.

Secondly, routing is well developed with special protocols for almost every scenario. We can benefit from the optimized routing performance without bigger additional efforts and we can take most of the performance evaluation in the state of the art as being proofed.

Thirdly, the task of mapping a name to an address and mapping an address to a route is similar. By integrating both tasks, we can save one step and therefore delay and traffic.

Fourthly, routing is per se self-organizing and fully distributed. By adapting the information management of the name resolution framework, we can easily realize a system without any central entity which is essential for our MANET scenario.

In the next chapter, we want to discuss fundamentals about routing in MANETs in general. This is the basis for our routing-based name resolution framework which is discussed in Chapter 5.

4 Fundamentals on Routing in Ad-hoc Networks

This chapter addresses the routing in networks, which is essential for our name resolution over adaptive routing approach. Routing in general is shown as well as the special routing approaches for MANETs, which can be done proactively or reactively, respectively. In the last section, the switching between different routing protocols called adaptive routing is introduced.

4.1 Term Definitions

Normally, networks are not fully meshed, which means that not every node can reach everyone with one hop. If a node in Europe wants to communicate with one in America, several intermediate nodes are needed in order to forward data. Networks are formed by routers interconnecting subnetworks into bigger networks like the global Internet.

If one node wants to send a packet to a known destination, it does not know in which direction it should forward the data, assuming that it has more than one neighbor. This leads to the problem of route finding. The easiest routing is flooding. Here, every packet is sent to every neighbor which forwards the packet to all of its neighbors and so on. This ensures that the destination surely gets the data but it generates massive traffic. It is obvious that flooding does not scale, therefore only one path to the destination should be used and this path needs to be discovered.

If a source node (S) for instance has two direct neighbors, like in Figure 4.1, but wants to communicate with another non-neighboring destination node (D), it has to choose one of its direct neighbors to forward the packets. A route discovery could result in more than one possible route. The process of choosing the best one, e.g., the node with the shortest path to the destination, is called routing. A routing protocol should chose the best route with respect to an appropriate metric, e.g., the transmission delay or any costs.

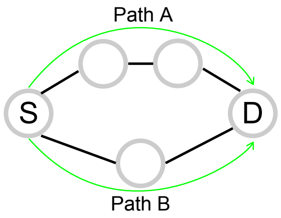

Figure 4.1: Example for multiple paths in a network

Finding routes is primarily a task for the Network Layer, which is the Layer 3 in the OSI model [44]. However, recent approaches for Layer 2 protocols, like the Transparent Interconnection of Lots of Links (TRILL) protocol [45], use different connections on Layer 2 for communication. Therefore, routing could also be a task in this layer. There are also approaches for overlay routing [46], where routing is performed in layers higher than the Transport Layer. This is for example used in Peer-to-Peer (P2P) networks and Delay Tolerant Networks. However, in this thesis we see routing as an exclusive task for the Network Layer to find routes to given IP addresses.

4.2 Introduction to Routing in MANETs

Routing in MANETs is a more challenging task. Nodes are moving and found links break frequently. In a highly mobile network, networks can even split into subnetworks, which leads to several broken links at once. Therefore, adapted routing protocols for such a special scenario were designed.

In general, the route finding process can be done either proactively or reactively. In proactive networks, route information is distribute periodically and in advance. The lower the refresh period is set, the more link break prone the routing works. With this mechanism, every node should have global knowledge about routes to all members in the network, periodically updated. Reactive routing only searches for the route on-demand. Only if a node really needs the route to transmit packets to a destination, a query is sent by the routing protocol. This can potentially help to minimize the traffic. For networks with only sparse application traffic, the routing overhead can therefore be decreased to almost zero.

Both approaches have their advantages and disadvantages. While the proactive protocols cause high periodical traffic, the delay until the route is found is zero. On the other hand, the delay is in the worst case maxed in reactive routing as the request has to reach the destination node and the reply needs to be sent back to the source. But it can significantly decrease the traffic load for the network.

It cannot be answered, which method is better. This is highly demanded by the use case of the network and the needs of the users. If the user wants to have real-time communication, proactive routing could be the better choice. If the network should work as energy-efficient as possible, reactive routing could be better. This problem is discussed later in Section 4.5

Switching the currently used routing protocol adaptive to the current scenario could increase the performance of the network (cf. Section 4.5). Therefore, nodes have to be able to speak several routing protocols, a distributed switching decision must be made, and already gathered information must be transformed to the formats of the new protocol.

In the upcoming sections, reactive and proactive routing is presented on the examples of AODV and OLSR followed by the introduction of adaptive routing frameworks.

4.3 Reactive Routing by the Example of AODV

In comparison to the proactive routing approach presented later in Section 4.4, routes are only discovered on-demand in reactive routing protocols. The most commonly used example is the AODV routing protocol, which is presented in this section. Information for this section were taken from the AODV RFC 3561 [47]. The discovery process subdivides into the requesting and the responding process, which are presented in the next two subsections.

4.3.1 Requesting Process

Every node maintains a routing table with information about routes to nodes in the network. The routing table contains the next hop to reach the destination and information about the up-to-dateness of the route as well as the number of hops to the destination. As long as a route is available, AODV does not come into action. If the node wants to establish a connection to any node where no actual route is available, the requesting process begins. The data packet is held as long as no valid route is discovered. The route discovery process can also be initiated by a packet from a foreign node that assumes a valid route over this hop that might no longer be valid.

To ask for a route, the requesting node sends out a packet called Route Request (RREQ). This is done by broadcast to every neighbor of the node. The routing message itself is the payload of a conventional User Datagram Protocol (UDP) packet which is sent via IP. The structure of the RREQ packet can be seen in Figure 4.2.

A particularity of AODV is the usage of *sequence numbers*. A sequence number is ascending placed in every request to indicate the freshness of the packet. It is also stored in the routing table. If another RREQ arrives from the same originating node but with a now higher

0 – 7	8					– 15 16 – 23	24 – 31
Type	J	R	G	D	U	Reserved	Hop Count
RREQ ID							
Destination IP Address							
Destination Sequence Number							
Originator IP Address							
Destination Sequence Number							

Figure 4.2: Structure of the AODV route request packet [47]

sequence number, the routing table entry is refreshed. If the sequence number stored in the routing table is greater than the one in the RREQ, the existing route is more up-to-date and is therefore kept.

With the *Hop Count* field, the receiving node knows how many hops the originator is away and can therefore compare different discovered routes. The *Originator IP Address* is the IP address of the requesting node and can be used for discovering the route to that node. The *Destination IP Address* is the IP address of the requested node.

It should be noted that the requesting node can set the *D* flag to indicate that only the destination itself is allowed to answer this request. The other flags are either reserved for multicast purpose, *J*oin flag and *R*epair flag, or for other AODV mechanisms that are not mentioned in detail in this thesis.

If no Route Reply (RREP) is arriving after a defined time, a new RREQ broadcast can be initiated. If a maximum number of requests is reaches, the requesting node gives up and assumes the destination to be unreachable.

4.3.2 Responding Process

Every node that has routing information for the requested IP address in the RREQ answers with a responding RREP. This could either be the requested node itself or any other intermediate node. The destination only flag in the RREQ can even force the network to only let the destination answer. The structure of a RREP packet can be seen in Figure 4.3.

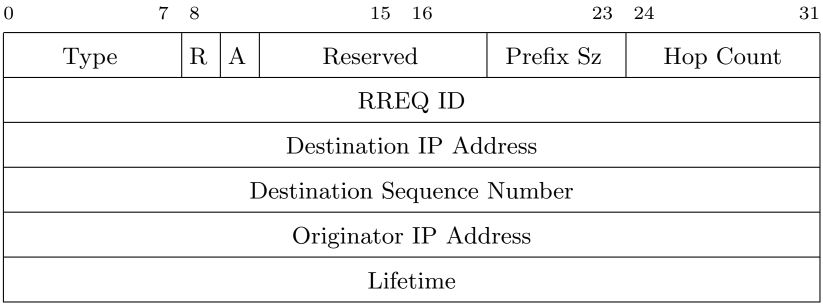

Figure 4.3: Structure of the AODV route reply packet [47]

The replying node can set the *A* flag, which means that the RREQ originator must send a RREP Acknowledgment back. This is optional and used if the link is unreliable. The *Destination IP Address* and *Destination Sequence Number* is sent along as well as the *Originator IP Address* which is the destination of the RREP. The *Lifetime* field gives the time in milliseconds for how long this route is considered as valid.

It could occur that the requesting node gets back several RREPs from different nodes. In this case, the node selects the route with the least costs and stores it in the routing table. One node is answering a request only ones, even if the same request arrives at the node over different routes.

4.3.3 Knowledge Distribution

The previously described discovery routine is done for every packet that needs to be delivered to an IP address with an unknown route. In the worst case scenario with zero precedent knowledge, every RREQ has to reach the destination node which responds with the RREP. However, in AODV nodes save all routing information they can get for future route discovery. Every overheard packet is used to update the own routing table.

A RREQ for instance is broadcast in the entire network. Every node that receives a request knows that it has a valid route to the originator and updates its routing table accordingly, no matter whether it can answer the request or not. Furthermore, every node that overhears the RREP from the destination knows that it has a valid route to this node, no matter whether the RREP is for that node or not.

Not only that intermediate nodes can use that knowledge for own future purpose, they can also answer future requests sent by other nodes. Considering an intermediate node that overheard a RREQ/RREP communication of two partners, it can afterwards answer RREQs from third nodes requesting a route to both the source and the destination. This saves traffic and lowers the delay at the same time, as replies reaching the new requesting node quicker and the distribution of the RREQ broadcast stops at the intermediate node.

4.3.4 Route Errors

In MANETs, it is generally assumed that nodes are moving and routes break up frequently. However, as routes are only discovered on demand, nodes could have stored active routes in their tables that are no longer available. A following unsuccessful connection establishment could result in a new route discovery process and a new route could be found, but this leads to even higher delays.

To overcome this, AODV has implemented a mechanism to inform other nodes of broken links. For instance, if Node A knows that it can reach Node C over Node B it stores that information in its routing table. If Node B realizes that the link is broken, it informs Node A that the previously discovered route to Node C is now unavailable. It does so by sending a so called Route Error (RERR) packet to Node A. Figure 4.4 shows the general structure of this message.

0 7	8	15 16 23	24 31
Type	N	Reserved	DestCount
Unreachable Destination IP Address			
Unreachable Destination Sequence Number			
Add. Unreachable Destination IP Address (if appl.)			
Add. Unreachable Destination Sequence Number (if appl.)			

Figure 4.4: Structure of the AODV route error packet [47]

With this packet, nodes can inform other nodes about several broken links at once. For instance, if Node C further couples Node D, this route is also no longer available and Node B informs Node A about two broken routes. The *N* flag can be set, if the node has already fixed the route and that this route should not be deleted from the routing table.

4.3.5 AODV-based Derivatives

Several extensions to the basic AODV protocol were proposed. The Ad hoc On-Demand Multipath Distance Vector (AOMDV) protocol proposed by Maria et al. [48] extends the basic AODV to provide multipath usage. In the basic AODV implementation, only the first received RREQ is answered with a single RREP, which is then used to calculate the route at the source node. That means that only the fastest path is discovered. In AOMDV all RREQs from the same originator coming from different neighbors are answered. AOMDV saves all received routes, which means that nodes take care of alternate paths. However, due do duplicated RREQs this extension leads to a higher overhead during the route discovery process but to a better efficiency regarding packet delivery afterwards [49].

Another extension is the Multicast Ad hoc On-Demand Distance Vector (MAODV) [50]. Here, the standard AODV is extended in order to enable multicast communication, which is not supported in the basic protocol.

There are several other improvements proposed based on the standard AODV or on derivatives, like the Low Latency extension for AOMDV [51]. In this protocol, some routes are guessed out of context in order to prevent future request phases, even if there is no guarantee that they actually exist. However, the adaptive routing framework that is the base for the name resolution system proposed in this thesis includes only the standard AODV as proposed in RFC 3561.

4.4 Proactive Routing by the Example of OLSR

While AODV is based on a request-response-mechanism that is initiated on demand, OLSR is based on precedent distribution of knowledge. This is done periodically, which means that changes in the topology are noticed by all nodes every period. The simplest way is flooding a packet through the whole network by every node. This would reach every other node and routes are established. However, OLSR implements an intelligent flooding mechanism that is presented in this section. Information for this section were taken from the OLSR RFC 3626 [52].

4.4.1 Preliminary Consideration

In OLSR, the knowledge distribution is done in several steps. Every individual step involves message sending, either unicast to direct neighbors or broadcast into the entire network. All packets are sent via UDP encapsulation at Layer 4 and the IP at Layer 3. All OLSR messages follow the same packet design. They are using the same header carrying the different messages. Figure 4.5 shows the standardized header of an OLSR message.

0 7	8 15	16 23	24 31
Message Type	Vtime	Message Size	
Originator Address			
Time To Live	Hop Count	Message Sequence Number	
MESSAGE			

Figure 4.5: Structure of the OLSR message header [52]

The packet structure contains a field to identify the *Message Type*, the *Message Size*, a *Time To Live* as well as the *Hop Count*, and the IP address of the originator. All this information has to be filled in for every message sent by any OLSR node.

4.4.2 Neighbor Discovery via Hello Messages

The process of distribution of global routing knowledge is subdivided in three steps. The first step is the neighbor discovery. Every nodes sends so called Hello messages periodically to all the neighbors via broadcast. The Hello message is sent with a time to live of 1. Therefore, the packet dies after the first hop. The Hello procedure ensures that every node is aware of all the nodes in the neighborhood. The packet structure can be seen in Figure 4.6.

0 7	8 15	16 23	24 31
Reserved		Htime	Willingness
Link Code	Reserved	Link Message Size	
Neighbor Interface Address			
Neighbor Interface Address			
...			

Figure 4.6: Structure of the OLSR Hello message [52]

With the *Willingness* field, a node can indicate that it does not want to be chosen to forward any data. The other fields are not explained in detail in this thesis. However, it should be noted that every node sends along all the 1-hop neighbors it already knows, within a variable number of *Neighbor Interface Address* fields.

Therefore, every Hello receiving node does not only knows every 1-hop neighbor after a Hello period, but additionally all 2-hop neighbors and the routes to them. A route to a 2-hop neighbors means knowing the 1-hop neighbor that has a connection to the node two hops away. This is important for the Multipoint Relay (MPR) selection that is described in the following subsection.

4.4.3 Multipoint Relay Selection

The main advantage of the OLSR routing is the intelligent flooding mechanism to distribute routing information in the whole network. This is done via so called *Multipoint Relay (MPR)* nodes. Due to the first process of Hello messaging, every node knows all of its direct neighbors and a list of all of their direct neighbors, which synonymously means all 2-hop neighbors (cf. Subsection 4.4.2).

In the second step, every node selects one or more 1-hop neighbors to reach all of the 2-hop neighbors. Those nodes are called MPRs. The best case would of course be that the node can reach all known 2-hop neighbors over only one node out of the set of 1-hop neighbors.

Figure 4.7 shows a example configuration. If we observe the upper right node, here called S for source node, it has two 1-hop neighbors and two 2-hop neighbors. But obviously it reaches both 2-hop neighbors over only one of the 1-hop neighbors and therefore selects this node as its MPR.

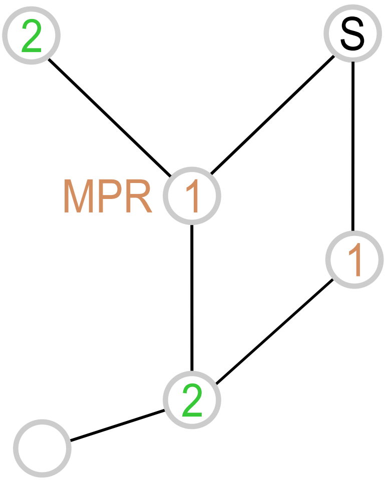

Figure 4.7: Example for a MPR selection

Every selected MPR is informed about the selection with a special message. We do not discuss the OLSR message design for this purpose as it is not relevant for the name resolution scheme based on this routing protocol.

4.4.4 Intelligent Flooding via Topology Control Messages

The third and most important step is the intelligent flooding. In the first step, all 1-hop and 2-hop neighbor were discovered; and in the second step, the best MPRs were selected. The task of the MPRs is to steer the flooding. OLSR is designed to flood routing information as efficiently as possible. That is why only MPR nodes are allowed to broadcast topology information.

Every MPR creates so called *Topology Control (TC)* messages. Those messages contain the list of all nodes that selected this node to be the MPR. The TC message is then distributed in the entire network. After the broadcast, every node is able to calculate a route to every other node.

The fact that only MPRs are allowed to broadcast messages limits the generated traffic in the network. This mechanism works best in large and dense networks as the MPR selection limits the number of transmissions in relation to the total number of nodes best here.

The TC message format is rather simple and shown in Figure 4.8. Every message contains one or several *Advertised Neighbor Main Addresses* and an *Advertised Neighbor Sequence Number (ANSN)* associated with the advertised neighbor set.

0 7 8 15	16 23 24 31
ANSN	Reserved
Advertised Neighbor Main Address	
Advertised Neighbor Main Address	
...	

Figure 4.8: Structure of the OLSR TC message [52]

All nodes receiving a TC message consequently save all addresses in their routing table to be reached over the node where the TC message arrived from. This finally results in routing tables filled with routes to every node connected in the network within each node. If the next TC message arrives, the entries are updated if the arriving information is more up-to-date.

4.5 Adaptive Routing

Routing protocols are highly developed and optimized. Unfortunately, they only perform best in a special scenario. The two chosen routing protocols AODV and OLSR are adapted to the special use case of a MANET. However, while AODV is recommended for large and sparse networks with a low mobility, OLSR is recommended for smaller, dense, and highly mobile networks [53, 54].

If the scenario is changing, it could be worth to just change the routing protocol currently in use. Routing frameworks providing such a possibility are called *adaptive routing systems.*

In the literature, some proposals were given to enable a routing protocol switching. A literature survey was done by W. Li during his student paper (Hauptseminar) [55]. The work was co-supervised by T. Finke and the author of this thesis. As the adaptive routing is only a kind of a tool for the Name Resolution over Adaptive Routing approach (cf. Chapter 5), state of the art protocols are only presented in a nutshell in this section.

Nada et al. [56] proposed a framework, where nodes can switch between different routing protocols. A particularity is here that no protocol has to be modified for usage in this framework. However, every node in the network has to switch to the same routing protocol at one time and the complete routing information from the current routing table has to be converted to the new routing table of the upcoming protocol in use. Therefore, a global decision has to be made by all nodes.

Hoebeke et al. [57, 58] proposed a protocol, where every node can use its own routing protocol. Therefore, multiple routing mechanisms can be in action at the same time in one subnet. Furthermore, no global decision has to be made. However, all protocols must be modified and adapted for usage in this framework. Hence, this is no plug-and-play solution for foreign nodes.

In the Zone Routing Protocol (ZRP) proposed by Beijar et al. [59], every node has a reactive and a proactive routing zone. A node can modify the radius of its zones depending on the scenario. Therefore, there is a kind of an adaptive routing with more than one routing algorithm, too, at least to a certain degree.

The Chameleon Routing Protocol (CML) proposed by Ramrekha et al. [60] also uses a centralized monitoring agent to steer the selection of the routing protocol.

In the adaptive routing framework presented by Finke et al. [61, 62], routing protocols do not have to be switched globally in the whole network. Every node is able to chose its own routing. The system itself is able to work with neighbors using different routing as the node itself. However, the author assumes the building of clusters in one area of the network with the same input for the *Switching Decision Maker*. Furthermore, a design criteria is that no routing protocol needs to be changed in its basic mechanisms. Therefore, it combines the main advantages of the two approaches presented by Nada et al. and Hoebeke et al.

Due to team work in the Graduate School that funded this work, we chose the framework of Finke et al. to be the basis for the following name resolution framework, including the reactive AODV and the proactive OLSR as routing schemes. This choice does not imply the supremacy of this approach over the others in the state of the art.

5 Name Resolution over Adaptive Routing

In this chapter, we describe the framework to provide name resolution in networks using adaptive routing. We introduce the simulation environment containing the Network Simulator 3 (ns-3) and the Click Modular Router (Click). Afterwards, details on the implementation of the proactive and reactive name resolution scheme as well as to the adaptive routing framework are given.

5.1 Basic Idea

In this section, we show the basic idea of integrating name resolution into an adaptive routing framework.

5.1.1 Preface and Design Criterions

As shown in Chapter 3, integrating name resolution into routing mechanisms can provide a MANET-adapted name mapping even for highly mobile scenarios. Due to the very high development of routing protocols, many research has been spent to push the performance of routing in MANETs further and further. By integrating name resolution into the routing, the name resolution can take advantage of the routing performance. That means that the name mapping should work es efficient, stable, and scalable as the routing itself. As route discovery is needed anyhow in any non-flooding-based network, integrating both tasks should not increase the traffic or delay too much. In Chapter 6, we show simulations to proof the following assumptions. We previously published some content of the following section in a conference paper [62].

The basic idea to provide name resolution in MANETs is to integrate the task into the routing mechanisms [62]. Several different routing protocols are available with proposed extensions. Every protocol is designed to work best in a defined scenario. We limit our investigations to two well-known and widely used examples, i.e., AODV and OLSR. Both protocols and their mechanisms were described in Sections 4.3 and 4.4, respectively. We chose those two because of their different ways to do routing, which is reactively and proactively, respectively (cf. Section 4.2). Other protocols use either similar mechanisms or hybrid approaches, but this couple is fine to show the major impact.

Usually, name resolution and route finding is done sequentially. Fist, a given hostname has to be mapped to the regarding local IP address and, second, the route to the found IP address has to be discovered. We strictly integrate both task to have a system that searches for **a route to a name**. This is done without changing the whole stack. The communication is still done by the Layer 3 protocol, in our case IPv4, using local addresses, in our case IP addresses. We are not introducing intermediate layers like in the HIP, which would destroy the common TCP/IP stack. This keeps us as compatible as possible to foreign nodes, e.g., victims trying to participate in the rescue network or vice versa.

Another design criteria was to keep the name resolution as close to the routing as possible. Generally, we considered different ways for routing-based solutions. The proactive routing could be extended with a reactive name discovery, which would lead to a hybrid approach. However, we wanted to be as close as possible at the basic routing ideas for each protocol with all advantages and disadvantages coming along.

5.1.2 Routing-based Name Resolution Using Reactive Routing

For the usage of a reactive protocol for name resolution, the needed functionality has do be integrated in the AODV packets being used to realize the on-demand route discovery. This is done via a RREQ packet to send an inquiry and via a RREP to answer the request. The RREQ is sent broadcast, while the RREP is sent back unicast. The idea to use those two packets to resolve names was first published by Engelstad et al. [41] discussed in Section 3.3. However, we add some further extensions to improve the performance (cf. Section 5.5 and Chapter 7).

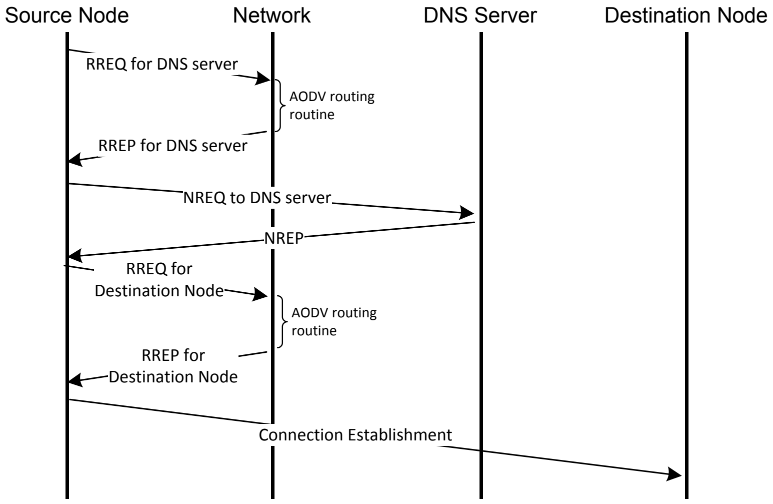

Figure 5.1: Message sequence chart for DNS-based name resolution with reactive routing [62]

Figure 5.1 shows the message sequence chart for a name resolution using a central DNS server and underlying reactive routing. As can be seen, the procedure contains four steps. Firstly, the requesting node has to find a route to the DNS Server, which consists of a RREQ sent broadcast by the requesting node and the waiting procedure for the unicasted RREP. Secondly, the requesting node sends the name request unicast to the DNS server and gets a unicast reply back. At this point, the name is successfully resolved. Afterwards, the node has to find the route to the found IP address, which means sending out RREQ and waiting for a RREP, again. The last step is then the connection establishment with the found destination.

It should be noted that this represents the working scenario where the source node, the DNS server, and the requested node are all connected and available and the DNS server got informed about the mapping. Furthermore, we assume the IP address of the DNS server to be known in the network. Otherwise, the process of finding such a server would lead to a kind of name resolution, too, which again leads to a chicken-and-egg problem. Predefined and announced IP addresses for the (several) DNS servers are not feasible as the might change their addresses, too.

Figure 5.2 shows the message sequence chart for name resolution integrated into the reactive routing, in this case AODV. It can be seen that the integration of the name resolution into the reactive routing can save two steps.

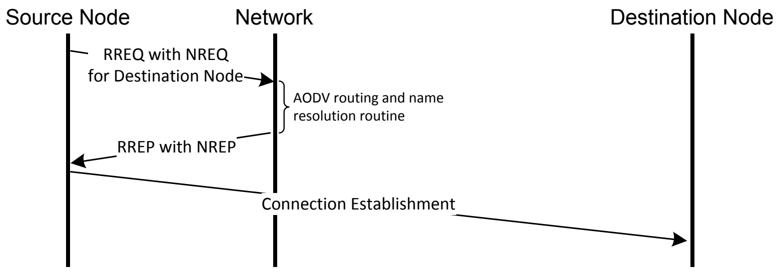
Source Node
Network
Destination Node
RREQ with NREQ for Destination Node
AODV routing and name resolution routine
RREP with NREP
Connection Establishment

Figure 5.2: Message sequence chart for routing-based name resolution with reactive routing [62]

As proposed by Engelstad et al., the Name Request (NREQ) is taken piggyback with the RREQ, which results in a RREQ/NREQ packet. The same is done with the two replies, the route reply and the Name Reply (NREP), which results in a RREP/NREP packet. The doubled request is sent to the node hosting the name, which is responding with a double reply. The communication with the DNS server is no longer needed and the name resolution is completely decentralized and integrated in the routing.

The elimination of two steps saves both traffic and delay. Less packets have to be sent and less steps have to be temporized. We show details on the concept and implementation in Section 5.5.

5.1.3 Routing-based Name Resolution Using Proactive Routing

Routing protocols are always optimized to a defined network scenario. If the scenario changes, the network could benefit from switching the routing protocol. That is why we use an adaptive routing framework in our system (cf. Section 4.5) [61, 62].

The second routing protocol in use is the proactive OLSR. In proactive routing protocols, information about routes are periodically distributed over the whole network. That means that every node has a periodically updated global view on the whole network and should know all routes to all participating nodes beforehand. To integrate name resolution in this routing approach, name mappings must be distributed over the whole network beforehand, too. This is done by a periodical distribution of all name mappings in the network. We show further details on the implementation in Section 5.6.

Figure 5.3 shows the message sequence chart for name resolution using a central DNS server and proactive routing. As can be seen, the node knows the route to the DNS server in advance and can immediately send the DNS request. After receiving the reply, it can immediately initiate the connection establishment as the route to the destination is also known in advance.

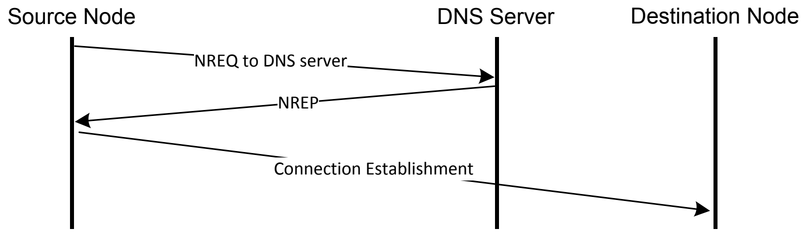

Figure 5.3: Message sequence chart for DNS-based name resolution with proactive routing [62]

Figure 5.4 shows the message sequence chart for name resolution integrated into the proactive routing, in this case OLSR.

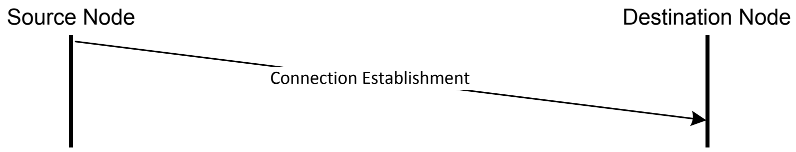

Figure 5.4: Message sequence chart for routing-based name resolution with proactive routing [62]

Due to the proactively distributed name mapping knowledge, the node can immediately establish the connection with the destination without any delay. The (negative) influence on the generated traffic is shown in Chapter 6.

5.1.4 Discussion

In the last two subsections, we showed theoretically that with the integration of name resolution into the routing some intermediate steps can be saved in both proactive and reactive routing. This leads to a lower latency for both schemes. In the proactive scheme, the delay is the only decreased unit by the disadvantage of more periodically generated load. A centralized approach generates of course less traffic than the decentralized solution as all direct communication with the central server is done unicast. But as centralized entities might fail frequently in our assumed scenario, an intelligent proactive distribution is needed.

The comparison of the presented reactive and the proactive scheme shows that the delay in proactive routing-based name resolution is zero while it is relatively high in reactive routing-based systems. This is archived at the expense of more generated traffic. While the proactive module generates periodical traffic no matter if name resolution is really needed or not, a reactive system leads to zero traffic if no node wants to resolve a name. This creates a diametrically opposed relationship between the two units. To select the optimal scheme for the current scenario, both values can be used as an input for the metrics of the adaptive routing framework. The decision then influences the routing performance and the integrated name resolution (cf. Section 4.5).

During the thesis, we will show the concept and implementation of routing-based name resolution on the example of the reactive AODV and the proactive OLSR routing protocols. Generally, we are not limited to these two protocols but chose them to show the different behavior of the two different fundamental routing approaches for MANETs. Integrated name resolution in other routing protocols work accordingly with presumably slightly different implementations. As long as there is an integrated request-response-mechanism in the reactive and an information-distribution-mechanism in the proactive routing protocol, the basic approaches discussed in this section can be applied.

5.2 Simulation Environment

For the simulation environment, various options are thinkable and available. We chose the open source and event-driven simulator *Network Simulator 3 (ns-3)* [63]. This simulator gives the user the possibility to create different scenarios for the same protocol stack to determine the behavior regarding node movement, traffic models, or other issues. All modules are written in C++, the scenario scripts can be written in Python or also in C++.

Ns-3 provides various protocols in all layers of the stack. Considering the Link Layer, implementations for Wireless Local Area Network (WLAN), Carrier Sense Multiple Access (CSMA), LTE, and Worldwide Interoperability for Microwave Access (WiMAX) are available. The Internet Layer comes with the usual implementations of IPv4 and IPv6. At the Transport Layer the TCP and the UDP is available. The Application Layer contains some dumb applications as the on-off-application, which sends senseless data just to simulate some random traffic. Furthermore, many routing protocols like AODV, OLSR, and Dynamic Destination-Sequenced Distance-Vector (DSDV) are ready to use. This gives the user a bunch of protocols to form a standard network with different scenarios. There is also a module dealing with node mobility and some analyzing tools.

A distinction must be made between the *module* and the simulation *script*. In a module, the implementation of a protocol including all functionality is described. Developing new protocols in any stack includes changes of the already implemented modules or writing completely new modules. The so called script defines the scenario, how many nodes participate, the movement of the nodes and which protocols are used on which stack (cf. Figure 5.6). Modules are loaded into the script via so called *helper*, where also all parameters implemented in the module can be set, e.g., the range of an antenna. A script can easily be changed to simulated different protocols with the same scenario. For instance, the same movement and traffic model can be tested with OLSR and AODV by just replacing the routing protocol in the script.

For the actual implementation of the protocols, we chose the *Click Modular Router (Click)* [64]. This software gives the developer a tool containing modules called *elements* to create a graph. Every element has its own task, whether this is a complex task like waiting for a route discovery or just a simple one like decrementing the Time to Live (TTL) of a packet.

By combining several elements, a full protocol can be built. Every element can be reused inside one graph and of course in different graphs. It is up to the developer to implement completely new elements or to create the same functionality by combining already existing ones. This combination is done by a script with an own easy scripting language. Click scripts usually run in the Network Layer and are therefore qualified to implement routing protocols. There are several elements available for core protocols like TCP, UDP, and IP as well as for often used functionality. It is of course possible to write new elements or to change already implemented elements to add functionality to the graph. Click elements are generally written in C++.

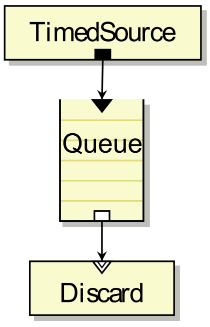

Figure 5.5: Example for a Click graph

Figure 5.5 shows an example for a Click graph. For reasons of simplicity, the code behind the graph is rather simple. The following script:

```
TimedSource
-> Queue
-> Discard;
```

contains of a timed source generating packets that flow into a queue and are discarded afterwards.

Every Click script can be run in either Kernel mode or in user space. The big advantage of this tool is that it can be run as a ns-3 module, too (cf. Figure 5.6). Therefore, a developer can validate the implemented protocol with various different simulations using ns-3 before running it on real hardware under Linux. The combination of a ns-3 simulation with a running Click script at the Network Layer is commonly know as *ns-3-Click*.

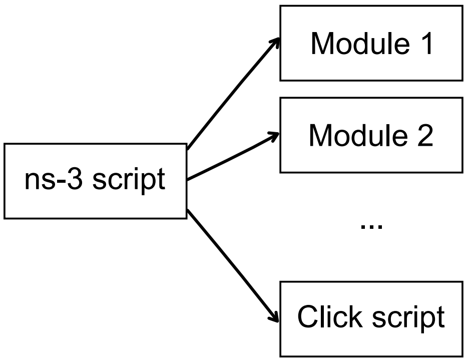

Figure 5.6: Relation between ns-3 script, ns-3 modules, and Click

Another program used in the simulations is *Bonnmotion* [65]. This tool can be used to generate movement files usable for ns-3, e.g., a random movement scenario where every node moves in a random direction with a random speed. The input for this program is the number of nodes, the maximum and minimum speed, and the range of the area where the nodes are located in. This program was chosen to simulate the MANET as randomly as possible and to include mobility in the scenarios. This proves the performance of our framework under unpredictable scenarios. Other movement scenarios like moving along streets in a Vehicular Ad Hoc Network (VANET) are generally practical, e.g., for moving fire trucks or police cars. However, we concentrated on the UAV and victim movement, which is more random and portrayed by Bonnmotion.

As ns-3 and Click both only run under a Linux-based operation system, we chose to use the commonly known Kubuntu Linux distribution [66]. Every other Linux would also be fine and should have no impact on the implementation and the simulation results.

5.3 Backbone

In this section, the backbone of the name resolution framework is presented. Our name resolution system is based on different interchangeable schemes represented by the several routing protocols. We just used two routing protocols to show the principle, but other modes are generally thinkable. If the protocol is changed, we want to save as much information as possible. Furthermore, some parts of the schemes are identical in the implementation. Therefore, we created a static backbone which is used for all routing schemes and is presented in this section.

Many papers in the state of the art focus on the address resolution process only. The protocol based on reactive routing presented by Engelstad et al. [41] for instance only covers the address finding process using AODV. Every node that wants to resolve a name has to initiate such a process again, as overheard knowledge is not stored inside intermediate nodes. We wanted to put more usage of already gathered distributed knowledge in our name resolution system. Therefore, we save every information into local tables and use this for future demands.

For the name mapping storage we invented a new table called the Hostname to Address Mapping (HAM) table. The structure of this table can be seen in Figure 5.7. The table itself is inspired from the routing backbone or more specifically the routing table. In routing tables, all discovered routes are stored and can be used for future demand. As we want to be as close as possible to the routing and the routing mechanisms, we designed the HAM table the same way.

Adaptive routing frameworks often keep the routing table or take over old routes to the new routing table of the new routing protocol if a switching decision was made. We use the same HAM table independently from the underlying routing protocol. This is represented by a Click element, which is independent from the rest of the graph.

data type	**field name**
string	hostname
uint8_t	hostname_length
bool	hostname_type
bool	address_type
uint8_t	address_length
IPAddress	local_address
bool	flag_multihomed
bool	flag_gateway
timestamp	time_to_live

Figure 5.7: Structure of a HAM table entry [67]

The main fields are of course the *hostname* itself and the *local address.* In our simulation we assume one version of IP on the Network Layer. That is why we set the data type to IP address. The Boolean field *address type* is set to either *0*, if it is an IPv4 network, or *1*, if IPv6 is used. But the structure is of course extensible if other Layer 3 protocols should be used. The field *address length* is correlatively set to either *32 bit* or *128 bit*, respectively. For future characterization of the hostname itself, we introduced the following additional fields for further description. If the Boolean *hostname type* is set to *1*, the hostname is hashed and not written in plain text. Further explanation to that can be seen in Section 7.1. The field *hostname length* gives the length of the string name in octets. As we assume a maximum length of 255 Bytes a 16 bit length field is sufficient. The maximum length is chosen because of the limitation of Fully Qualified Domain Name (FQDN) to that value [24]. The Boolean *flag multihomed* is set to *1* if the node hosting the name has more than one local addresses due to multiple interfaces. The *flag gateway* field is set to *1* if the node acts as a relay node to another network. A gateway node has to be multihomed by default while a multihomed node might not necessarily initiate a network coupling.

The table is implemented as a C++ map of the hostname as a string to the remainder of the HAM table entry as a structure of the mentioned data types. This design is similar to the implementation of the routing table and provides a fast search for entries to a given hostname.

The HAM table runs completely independent from the routing table. It would have been possible to just integrate the name mapping information into the routing table itself to save storage and information handling. However, we separated the storage to be independent from the routing protocol itself and to change the implementation of the routing protocol as little as possible. Especially by using adaptive routing and therefore switching from one protocol to another, an independent storage is essential (cf. Section 5.7).

The content of the HAM table can be used by Layer 3 of the node to answer name resolution requests from the Kernel (cf. Section 5.4). Every node has its own HAM table with possibly different content. The table itself is filled by the underlying routing protocol, either proactively if OLSR (cf. Section 5.6) is running or by initiating a request if AODV (cf. Section 5.5) is used.

Figure 5.8 shows an example HAM table entry for the node with the ID *Multicoper_3*.

field name	**value**
hostname	Multicoper_3
hostname_type	0
hostname_length	12
address_type	0
address_length	32
local_address	10.0.0.2
flag_multihomed	1
flag_gateway	1
time_to_live	2014-02-04 15:41:22

Figure 5.8: Example for a HAM table entry

Routing table entries are in most cases equipped with *timestamps* or an *expiration time*, respectively. As routes change frequently, old discovered routes are set invalid after a certain time. We included such an expiration time for our HAM table entry, too. If the *time_to_live* is set to *0*, the entry has no expiration time and is valid until it is deleted. The decision, whether to add an expiration time or not is given to the underlying routing protocol. The backbone itself checks the HAM table for expired entries with a defined period and deletes the entries accordingly. However, we included such a check also in the routing protocol during the entry search process.

5.4 Communication between the Kernel and the Routing Protocol

In Click implementations, two interfaces are used to communicate with the routing protocol that is implemented by the Click script. One interface sends and receives packets from the network interface on Layer 2, the other interface communicates with the Kernel. In our scheme, we assume some application to request for a name mapping. We assume applications trying to establish a connection based on identities. For instance mission planning applications to coordinate rescue teams could send mission requests to all UAVs represented by previously defined UAV identities. Therefore, this application has to resolve the UAV IDs first before sending out packets.

We realized that by a Click element, which captures such name requests from the Kernel and looks up the requested name in the cache represented by the HAM table (cf. Section 5.3). If it finds the regarding IP address, it immediately sends back a reply to the Kernel. Otherwise, the routing takes care of the name resolution. In OLSR, all name mappings should be available beforehand. If a mapping cannot be found in the own subnet, either in the proactive mode at the time of the Kernel request or in the reactive mode after the name resolution routine, the searching could be extended to other coupled networks. This routine is described in detail in Section 5.8. The following two sections describe the proactive and reactive mode in one subnet, respectively.

5.5 Reactive Scheme

In this section, we provide details on the implementation of the reactive scheme. The basic implementation for the reactive scheme was the AODV implementation in Click provided by Bart Bream [68]. We previously published some content of the following section in conference papers [62, 67].

5.5.1 Standard Compatibility

As written in Section 5.1, the reactive scheme is realized by adding extensions to the AODV packets. AODV uses the RREQ packet to send out a request via broadcast and the RREP packet to send the reply back to the originator via unicast (cf. Section 4.3). Referring to the AODV RFC 3561 standard Chapter 9 [47], it is allowed to add extensions to the usual AODV packets without being not standard compliant.

Consequently, an appropriate packet handling for the new extensions has to be included in the implementation of the routing protocol in all participating nodes. All other nodes could just withdraw packets with unknown extensions or forward the packet without any processing - depending on the actual implementation.

We assume all nodes to be either equipped with our system or forwarding the data without processing.

5.5.2 Name Request Message

The RREQ packet is extended with a NREQ message. This combined packet searches for a name to be resolved to an IP address and for the route to this found address. The format of the RREQ is described in Figure 4.2.

As every RREQ needs an IP address to find the regarding route, we need to add some address in the regarding entry. A necessary requirement is that this address is definitely not part of the network. Otherwise, a node could answer this reply before having the piggybacked NREQ resolved. Possible addresses are the broadcast address *255.255.255.255* or the zero address *0.0.0.0*. We chose the zero address for our system.

The remainder of the RREQ can be set as usual, including sequence numbers. This ensures that the packet is treated as a common AODV request. Nodes, not being aware of the name resolution system will just take the RREQ as it is. As the zero address is forbidden for node usage, those nodes will not have any routing information and would forward the packet to the next hop. That means that this approach is completely standard compatible to nodes using a standard AODV version or any derivate without the name resolution extension.

The construction of the NREQ, as shown in Figure 5.9, is straightforward. It has a Type Length Value (TLV) structure as it is defined by the AODV RFC [47].

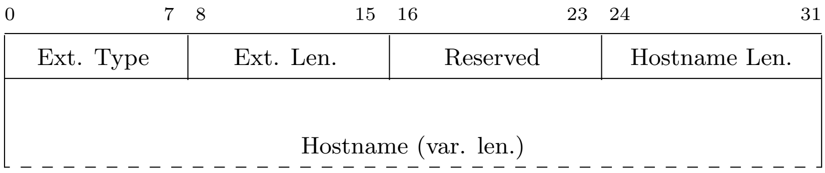

Figure 5.9: Structure of NREQ message extension

The *extension type* field gives the predefined type of the extension. The range of the extension numbers is from 1 to 256, while 1 to 127 is already reserved for the AODV protocol. For the NREQ extension, type *128* is set while for the NREP message, type *129* is used. The *Extension Length* field gives the length of the whole extension including the *Type* and *Length* field. The body of the message consists of a *Reserved* field for future extensions and the *Hostname Length* of the actual requested name. The following field *Hostname* contains the hostname string that needs to be resolved itself and has a variable length.

5.5.3 Name Reply Message

The NREP extension is added to the AODV RREP packet. The structure of the RREQ can be seen in Figure 4.3. The structure, as shown in Figure 5.10, is more complex than the one from the NREQ because some further information for the name mapping is transmitted. The *Extension Type* field is, as already mentioned, set to *129*. The *Extension Length* field is set to the length of the whole extension again. After that, all necessary information is inserted but the IP address, which is included in the surrounding RREQ or RREP, respectively. The flag *A* indicates the type of the local address, whether this is *IPv4* or *IPv6*. The two flags *M* and *G* indicate, whether the node is *Multihomed* and a *Gateway* to other networks. Finally, the *Reserved* bits are reserved for future extensions and set to *0* by default.

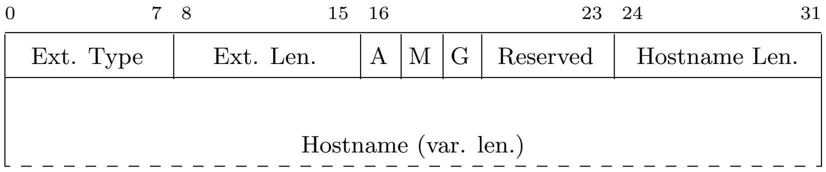

Figure 5.10: Structure of NREP message extension

The question could occur, why the hostname is transmitted in the NREP, too. It is of course not necessarily needed. The requesting node knows which name mapping it has requested. But as previously described, every intermediate node should learn from the NREPs to answer future requests or just to have the information for own future need. By sending the hostname along with the reply the generated traffic is increased slightly. But as the reply is sent unicast back to the requesting node the overall additional network load is negligible by the advantage of having more distributed knowledge. We want to prevent broadcast messages as good as possible and accept slightly more traffic in the replying phase to have a big benefit during the requesting phase.

5.5.4 Intermediate Step for Distributed Knowledge Usage

As proposed by Engelstad et al. [41], a node searching for a name mapping sends out the RREQ/NREQ packet and the regarding node hosting the name replies with the RREP/NREP packet.

We introduce a further step to improve the performance. All name mappings are not just consumed, they are stored in every node for future use. This could be a name mapping request from the Kernel of the same node or a request of another node. Furthermore, we let all intermediate nodes overhearing a NREP save the name mapping for the same reason. Saving all the mappings of course leads to a memory consumption. But having enough free memory space on the nodes is much more likely and resource-aware than sending a new request from the originator to the destination for every new name resolution needed. Memory is much cheaper than channel capacity or (transmission) energy in MANETs.

Another reason for the caching of all received name mappings is following the style of the routing protocol. In AODV, every overheard route reply is also stored by the protocol and used to answer future RREQs by intermediate nodes. Simulations comparing the non-caching mode with the caching mode are shown in Section 6.3.

Due to the distributed knowledge, intermediate nodes can answer requests even if they are not the node hosting the name or being aligned to the IP address. However, we assume that the name mapping is much more static than the route to an IP address. A node will not change its ID every millisecond but in a highly mobile network, routes could change frequently. That is why the HAM table entries have a much higher time to live until they expire. But this leads to the situation that one node could have a valid mapping stored to answer the NREQ but no valid routing information to answer the resulting RREQ. On the other hand, another node could have overheard the needed RREP, maybe from a usual AODV route discovery, but is not aware of the name mapping.

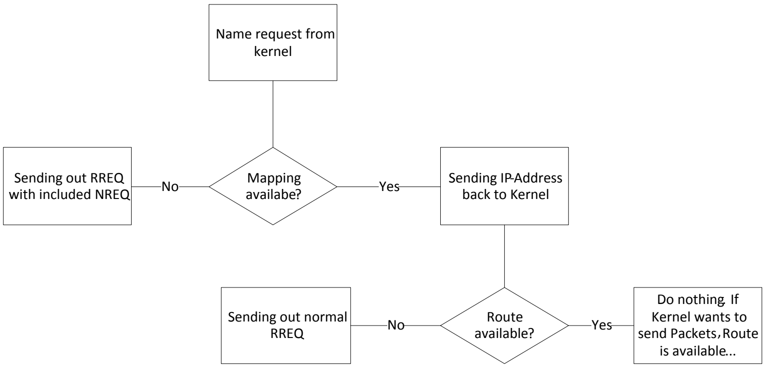
Name request from kernel
Mapping availabe?
No
Sending out RREQ with included NREQ
Yes
Sending IP-Address back to Kernel
Route available?
No
Sending out normal RREQ
Yes
Do nothing. If Kernel wants to send Packets, Route is available...

Figure 5.11: Flowchart for processing name requests coming from the Kernel in reactive mode

To use this split and distributed knowledge we introduced an intermediate step in the resolution procedure. First, every node answers the NREQ with a NREP if it has the information available. This results in a RREQ/NREP package. Therefore, we changed the implementation of the RREQ packet to be potentially extended with a NREQ and a NREP message, respectively. The dummy IP address in the RREQ part is then replaced with the newly found IP address belonging to the requested name. Therefore, the RREQ is no longer a dummy one but a usual one with a valid address to be found. If then another node answers this new RREQ with a RREP, the resulting RREP/NREP packet can finally be sent back to the requesting node via unicast. It is obvious that if a node has both information at the same time, it answers both requests sequentially. Otherwise, the RREQ/NREP packet is sent broadcast and treated as a usual RREQ of AODV. The intermediate step furthermore introduces the possibility to include nodes without the protocol extension installed in the resolution process, at least for the second phase discovery the route.

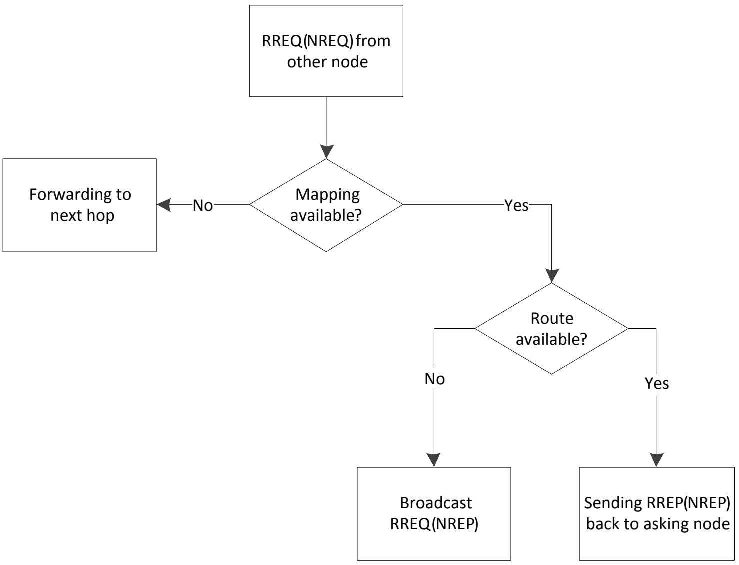

Figure 5.12: Flowchart for processing name requests from other nodes in reactive mode

The flow chart in Figure 5.12 shows the action a node takes if a name request from the Kernel arrives, while the flow chart shown in Figure 5.11 demonstrates the complete procedure, if a node receives a RREQ/NREQ packet.

5.5.5 Expiration Date Handling

The HAM table in the system's backbone gives every module the possibility to store an expiration date. If the expiration date is reached, the entry is invalid and deleted. However, if the expiration date is set to zero the entry has infinite validity.

In contrast to the routing information, we let the reactive mode set no expiration date for the name mappings, as we assume them to be more stable than routes. AODV has no refreshing time as it only reacts on-demand. Name mappings would be invalidated frequently and we would have no benefit from the distributed knowledge. In the case that a node gives up its identity and another node still tries to use the mapping, a Name Error (NERR) message is sent by the destination to inform the source about an expired name. Details about the NERR message and the message handling is shown in the next Subsection 5.5.6.

Setting an expiration date would lead to more traffic and delay for every node as most of the new Kernel requests would lead to new AODV name requests. Not setting such a validity time limits the additional traffic to those cases, where the name mapping meanwhile expired without notice of the network. However, this raises the delay even more as a requesting node needs to discover the expired binding and afterwards has to wait for the discovery. But it is worth to accept this with the assumption of almost stable name bindings. MANETs can not guarantee reliability anyhow.

5.5.6 Name Error Message

It can occur that one node has a previously discovered name mapping stored in the HAM table, which is no longer valid. As we assume name mappings being more or less static, we chose an infinite TTL using the reactive scheme if a mapping is once stored in the HAM table. If the name to address assignment expires, a previously discovering node does not necessarily get knowledge of this. If then the source node tries to establish a connection to the IP address being assigned to a given hostname in the HAM table, the destination node needs to have the possibility to react on it. That is why we introduce a NERR message in analogy to the RERR message in the AODV routing protocol.

With this packet, a node can inform other nodes about the expiration of a name mapping. If a node wants to establish a connection to a node assuming the name to be still hosted, the node generates the NERR and sends it unicast to the originator node. This can either be initiated in advance to inform former communication partners by any application or after a connection establishment request to a name that is no longer valid. If an intermediate node overhears such a unicast sent message, it simply deletes the regarding HAM table entry, too, if applicable.

Intuitively, the NERR message could be attached to the RERR pendant of AODV. However, this would make no sense as the RERR would erase a route that is possibly still valid. Therefore, we attach the NERR to a RREP packet providing the route to the originator of the NERR. This route is possibly not requested by anyone, but could be used for future need.

The structure of the NERR messages is shown in Figure 5.13.

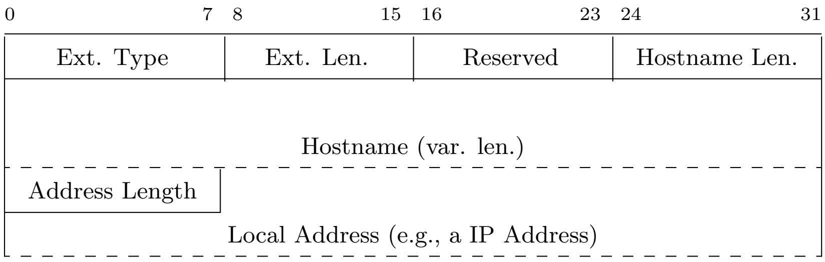

Figure 5.13: Structure of NERR message extension

The *Extension Type* is set to *130* (cf. Extension Type 128 for NREQ and 129 for NREP). The *Extension Length* field is set to the value of the whole extension. The *Reserved* field is reserved for future extensions such as flags and set to *0* by default. After that, the *hostname-address*-couple is appended together with the variable *length*, respectively.

5.6 Proactive Scheme

In this section, details on the implementation of the proactive scheme are provided. The basic implementation for the reactive scheme was the OLSR implementation in Click again provided by Bart Bream [69]. We previously published some content of the following section in conference papers [62, 67].

5.6.1 Standard Compatibility

The proactive OLSR distributes routing information in advance using Hello and TC messages. Hello messages are used to discover 2-hop neighbors and to select a subset of 1-hop neighbors to be MPRs. Every MPR then broadcasts TC messages containing all nodes that selected this node to be a MPR (cf. Section 4.2).

In our design, we want to integrate the name mapping as close to the basic routing mechanisms as possible. Therefore, we adapt the idea of periodical Hello and TC messages and we use the OLSR mechanisms to provide a network-wide name mapping distribution.

The OLSR RFC 3626 standard [52] gives the opportunity to introduce new control messages. This is slightly different from the AODV implementation. In the reactive mode, we extended already existing AODV messages with newly introduced extensions, which was standard compatible. In the proactive mode, we introduce new messages without extended existing ones. However, OLSR supports stacking of several messages, which we used to include our messages in the OLSR message stack.

5.6.2 Name Advertisement Message

As a new OLSR message, we introduce the so called Name Advertisements (NADVs). With this message, a node can inform the network about a name hosted on the node. This could be used for binding an identity to the current local address or to announce a service (cf. Section 7.2). The body of the NADV message is shown in Figure 5.14.

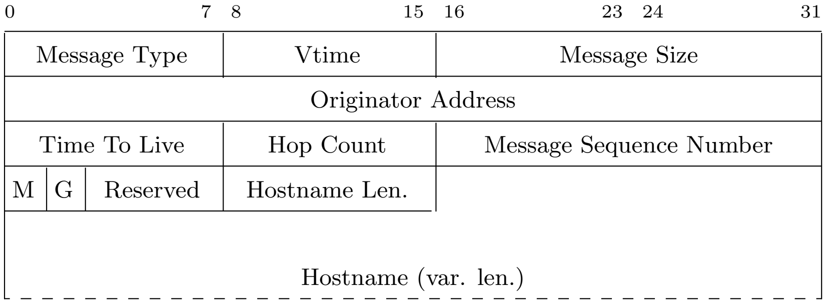

Figure 5.14: Structure of the NADV message

The structure of an OLSR message is shown in Figure 4.5 and described in Section 4.4. The header of the OLSR message is standardized, while the message body can be modified.

The header fields are set as follows. The *Message Type* field declares the type of the message. The range of the type is set by the OLSR standard to 1-255. The types 1-4 are used for the messages of the standard implementation. Types 4-127 are reserved for future OLSR core extensions. Types 128-255 are free to use (cf. OLSR RFC 3626 Chapter 22 [52]). We set the type for our newly introduced NADV to *128*. The *message size* is set to the entire length of the message, which is the standardized header size plus the size of the message depending on the length of the distributed hostname. The *Originator Address* includes the regarding local IP address. As this is the IP address where the hostname is mapped to, it is not necessary to include this information again in the message body. The *Hop Count* is set to zero and the *Time to Live* to the range in which the message should be distributed. The *Message Sequence Number* is set by OLSR.

The message body contains the remaining necessary information, which is not already included in the header. Firstly, we placed two flags, *M* and *G*. If *M* is set, the node is multihomed. The *G* flags indicates, whether the nodes couples several networks and acts as a relay. We reserved some bits for future need in the *Reserved* field, which is set to *0* by default. The *Hostname length* field gives the variable length of the hostname being distributed with that message. Finally, the *hostname* itself is attached.

5.6.3 Knowledge Distribution via Multipoint Relays

We want to inform the whole network about the name mapping. One way would be a normal flooding through the network. In this case, the *Time to Live* field is set to *255* and the NADV is broadcasted. By integrating the name resolution into OLSR, we can make use of the more effective OLSR flooding mechanism. That is why we flood the name mappings only over the previously selected MPRs. In a first step, every node sends its own mappings with a *Time to Live* of *1* to one MPR node previously selected by OLSR. If a node has more than one MPR neighbor, it selects the one with the most two hop neighbors.

We assume an identity to be unique for one node. But it can occur that one node has more than one identity, e.g., if it participates in more than one logical network or if it hosts more than one service. In that case, several NADV messages can be stacked into one OLSR packet. We assume that one node only hosts a single-digit number of names. However, by stacking the NADVs traffic can be minimized instead of sending every NADV as an extra packet.

A MPR node collects all received NADVs over a time period and stacks them again into one single packet. This packet includes all name mappings of the single hop neighbors and its own mapping(s).

In the second step of our proactive scheme, this packet is now broadcasted by the MPRs within a defined period. With this step, we prevent the network from being flooded by a, possibly stacked, NADV message from every node in the network. Instead, only a subset of nodes, i.e., the MPRs, distribute collected information into the whole network. As every node selects the MPR node with the most connections to 2-hop neighbors for broadcasting the NADVs, this node creates biggest possible stacked packets with a high probability. Therefore, we prevent the flooding of smaller packets by several nodes by instead sending bigger packets by only a few nodes.

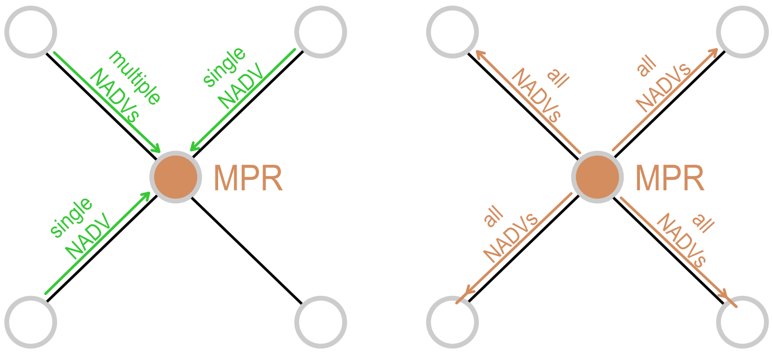

Figure 5.15: Two step process for NADV distribution

Figure 5.15 shows an example scenario for the two-step process. We chose a simple star topology to simplify the visualization. The node in the middle is obviously chosen to be the MPR by all other nodes. In the first step (left figure), every node informs the MPR about its own mappings. It should be noted that every of the four green marked messages is completely different from each other. Each of the three messages could contain multiple name mappings at once. Every of the four outer nodes sends its NADV message at a different time as they are not necessarily synchronized. The MPR is collecting all received one-hop NADVs until the time period for collecting is over. At that time, all received NADVs are stacked into one big packet and flooded through the network (right figure).

In the shown case in Figure 5.15, the packet only takes one hop in each direction. For bigger networks, this would be forwarded into the entire network. Please note that all orange marked packets are identical.

If a NADV message passes one node, whether it is forwarded afterwards or not, the information is stored in the HAM table or an already existing entry is updated. For the case that the proactive routing is enabled by the adaptive routing framework, we set the TTL of the HAM table entry to twice the sending period of the NADV message. This ensures that an entry automatically expires after two periods if no update was received.

5.6.4 Period Handling

One parameter that can be set by the user or the administrator, respectively, is the period of message sendings. This has an influence on both, the up-to-dateness of the distributed information and the generated traffic load for the network. The interval of the OLSR-internal Hello and TC message emissions can also be set depending on the network scenario.

We assume that the assignment of names to IP addresses is much more stable than routes between nodes. We can influence the generated traffic by selecting a relatively high NADV period in comparison to the OLSR message emission. In our example, the interval of the Hello messages was 2000 ms while the TC interval was 5000 ms. Both parameters were chosen by Bart Bream in the initial implementation [69]. We set the interval of the NADV-Hellos to 4000 ms, which is double the OLSR Hello period and the period of the NADV broadcasting to 10000 ms, which is double the OLSR TC period. This ensures that a MPR gets a NADV message twice before broadcasting, which increases the probability that this information is included. Furthermore, the periods of the name mapping distribution is a multiple of the OLSR periods, which provides the possibility to send both messages together in one packet.

However, it can certainly happen that a name mapping disappears within one period of NADV sending. If a node then wants to establish a connection to the IP address where it assumes the name is assigned to (cf. Figure 5.4), the destination node just responds with a NERR message. For simplification, we use the same NERR style as presented in the reactive scheme (cf. Subsection 5.5.6). We introduced the NERR message as a new OLSR message, which is sent unicast to the originator node. The OLSR message type for this message is set to *129* (cf. Message Type 128 for NADV messages).

A node, which wants to release a new hostname has to wait an undefined time until the information is distributed in the network. In the worst case, it lasts the own NADV emission period plus the NADV broadcasting period of the selected MPR node and an additional time until the flooding is finished. This could be important for the implementation of possible applications using the proactive name resolution. Furthermore, we have to take into account that messages could be lost due to the mobile communication. The NADV transmission to the MPR could fail as well as the distribution of the stacked NADVs packet. Therefore, the delay until the name is distributed could be increased even more. However, interferences can also disturb the reactive mode and therefore increase the delay therein, too.

Another idea to increase the OLSR performance is to send NADV messages along with normal OLSR messages. NADV emission to the MPR could be stacked with OLSR Hello messages to save traffic. The NADV broadcast of the MPR could be combined with OLSR-TC messages. As the emission period of the messages used by our proactive name resolution mechanism is a multiple of the OLSR messages, those messages are emitted at the same time anyway. With this approach, we can limit the additional (periodical) traffic in the proactive mode to just the length of the NADV messages without the necessary overhead caused by header of lower layers.

5.7 Adaptive Name Resolution

In the last two sections, we only contemplated the reactive and proactive scheme isolated. As we use adaptive routing and switch between different routing protocols, the next section deals with the switching process.

5.7.1 Independence from the Routing Protocol

The implementations of the reactive and proactive scheme are independent from each other. They can work alone. Every scheme has its advantages and disadvantages. While the OLSR-based module produces a periodical traffic even if no name resolution is needed at all, the delay to resolve names is zero if requested. In the reactive scheme, we have a delay depending on the network size but we can potentially save a lot of traffic. We introduce adaptive routing to our framework to change the routing protocol during runtime if the scenario is changing and the new situation can be better handled with a different protocol (cf. Section 4.5). In this section, we show the mechanisms to support such a routing protocol switching with integrated name resolution.

Even if the mechanism to resolve names to local addresses is integrated into the routing, the storage of the information is completely independent from the routing table. The storage of the name mappings is independent from the routing table and the routing protocol and the same backbone is used in both schemes. It would be possible to extend the several routing tables to just include naming information, but we decided not to use this because of two reasons. Firstly, we want to keep the routing protocols as untouched as possible and as standard compatible as feasible. This is moreover one of the design criteria of the adaptive routing framework of Finke et al. [61]. Secondly, we want to keep all previously gathered information, even if the routing protocol is changed, without copying the data. That is why we created the HAM table fully independent from the routing protocol. Every routing mechanism can access the HAM table and can add, delete, or edit entries. That means that even if the routing is changed the name mapping system does not need to start from scratch. Furthermore, this design is also essential for providing a degree of extendability.

5.7.2 Switching from Reactive to Proactive Routing

In the reactive mode, the distributed knowledge is incomplete. Only nodes which really need a name to be resolved initiate the name resolution routine. Other nodes overhearing the Name Reply save the knowledge to their own HAM table for possible future need. However, it is likely that not all of the mappings are distributed in the whole network.

Furthermore, the entries could be outdated. We omitted to store expiration dates for the entries in the reactive mode as they are not updated periodically but might be still valid. Therefore, outdated entries could occur.

If the network changes from reactive to proactive routing, the global knowledge is reestablished after the first complete NADV distribution process, assuming no critical packet loss. The maximum time until global knowledge is established depends on the set periods for NADV emission to the MPR and distribution by the MPR. Furthermore, the largeness of the network influences the flooding due to multihop sending. However, old entries from the reactive mode should be refreshed by proactive distributed ones relatively quick.

5.7.3 Switching from Proactive to Reactive Routing

In the proactive mode, every node should have global knowledge due to the proactively distributed NADV messages (cf. Section 5.6), at least if the the proactive mode is active long enough. If the adaptive routing changes the mode to a reactive protocol, in our case AODV, the network can use the gathered name mapping knowledge.

As every HAM table entry is stored with an expiration date, we introduced a function that sets all deadlines to zero if the routing is changed to any reactive mode. This ensures that no entry expires at all. This reduces the generation of NREQ packages. However, a non-expanded normal AODV RREQ could be necessary if the route changed meanwhile. Furthermore, nodes could change their local addressing or even release hosted names, which leads to NERR messages and an initialization of the name discovery process.

5.8 Coupling Several Networks

In the last sections, we dealt with our routing-based name resolution system inside one subnetwork. In this section, we present our inter-domain name resolution approach based on Border Nodes (BNs). We previously published some content of the following section in a conference paper [70].

5.8.1 Preliminary Considerations

The design of our framework so far is based on the idea that all participating nodes are within one network and that every node is able to reach the others. However, the disaster scenario that is the basic assumption for this thesis might contain several networks, presumably heterogeneous ones. In order to couple such subnets, an intelligent node's placement can be used [71], e.g., due to flying nodes like UAVs. If the nodes in the two coupled subnets use the same protocol stack, both MANETs could be merged using merging algorithms [72].

However, we assume heterogeneous subnets. Those must be coupled using multihomed nodes, i.e., a node that is connected to more than one networks at the same time independently. Multihomed nodes may have different addresses in each subnet and use different protocols, e.g., different routing. If a conversion of the technologies of the two networks is possible, we can furthermore assume that a communication between two nodes residing in two different subnets is possible, too. But before that, we have to deal with inter-domain routing and inter-domain name resolution. Hence, we enhanced our framework with the possibility to find nodes in foreign networks as well as the route to the nodes by using additional functionality in the multihomed border nodes. In the following subsections, we present different cases and how our system reacts.

5.8.2 Term Definitions

In this subsection, we define important terms which are used in the following sections.

A node, which couples two networks is called a *border node*. A border node has multiple interfaces and is connected to at least two different networks. Those networks could use the same stack or completely different technologies. However, a border node must be able to translate data from one network to another to ensure communication between the networks. The concept of a border node is equal to a multihomed node with several IP addresses. Other names in the literature are *relays* or *gateway nodes*.

In the following subsection, we entitle the network to which a node is connected to the *home network*. All other networks are *foreign networks*, especially those which are coupled by border nodes. A border node itself has at least two home networks. Networks between the home network and the network of the destination node, in case of cascaded networks, might be called *intermediate networks*.

5.8.3 Requesting Node Resides in a Reactive Home Network

Firstly, we want to discuss the behavior of the system if the requesting node resides in a network using the AODV-based scheme.

In the first consideration, we let the requesting node be in a **reactive home network** and the requested node in an also **reactive neighbored foreign network**. The initial process being started is the reactive AODV-based routine to resolve a name as described in the sections before. We let every requesting node first try to use the home network technology. In the reactive case, this would lead to a RREQ/NREQ broadcast. As the demanded node is not located in the home network, no participating node might answer.

For inter-network communication, we identified two possibilities to proceed. We can either initiate an additional RREQ/NREQ after the first one failed, which is then sent unicast to all border nodes. Or we let every border node automatically forward requests in the neighbor networks. The first method might save some traffic but causes a higher delay in the worst case. The requesting node has to wait a reasonable time for possible replies until sending the (second) unicast request to the known bordernodes. The second method delivers a faster discovery but might result in unnecessary traffic in the case that the requested node is in the home network, because the request is automatically forwarded. However, we need to introduce and implement more message types and message handling for the first method compared to the second one, as unicasted requests need different handling compared to the broadcasted one. Therefore, we decided to use the second method and let each border node automatically distribute requests in the coupled network.

In order to prevent flooding nevertheless, we make use of the AODV-mechanism called *Expanded Ring Search*. This prevents a request from being unnecessarily flooded through the whole network by setting a TTL. If a RREQ fails the now necessary retry is triggered with an increased TTL. On the one hand this automatically leads to more traffic if the destination is many hops away, as the request is flooded again and again with increasing range until we get a reply or the set number of maximum retries is reached. On the other hand, we can decrease the traffic significantly if we find the destination quickly. By using higher initial TTLs, we can at least prevent the network from flooding several coupled networks in a cascade with every initialized request.

In the second consideration, the requesting node is located in the **reactive home network** and wants to resolve a hostname, which is hosted on a node in a neighbored **proactive foreign network**. Let the system be without previous knowledge in the intermediate nodes. Again, the initial AODV process fails within the home network and the request reaches the border nodes eventually. However, the border node coupling the proactive foreign network has global knowledge about the proactive side and is therefore informed—by the foreign network's proactive scheme—about the mapping in advance. As the HAM table runs independent from the routing, the node can use the knowledge gathered in the foreign network to answer the reactive request coming from the home network of the requesting node. As the border node is the only one with a connection and therefore a route to the foreign network, it will also translate and forward the upcoming data transmission.

5.8.4 Requesting Node Resides in a Proactive Home Network

In the second case, we let the requesting node be in a **proactive home network** and the hostname to be mapped hosted on a node in a coupled reactive network.

In the initial process, the requesting node recognizes that it has no mapping information in its HAM table as the node is not located in the proactive home network. For a successful resolution, we have to transfer the knowledge in the reactive foreign network to the proactive home network. Two approaches are possible for this task.

One solution could be that the border node automatically floods the knowledge gathered in the reactive network periodically into the proactive network. If a node is in the border node mode, which is automatically the case if the node participates in more than one network, it automatically sets itself to a MPR (cf. Section 5.6), at least for the name resolution task. This means that the border node periodically distributes its HAM table into the proactive network. This obviously would produce more overhead and the gained knowledge might never be used. However, this approach is close to the general behavior of the proactive routing approach. On the other hand, this cannot secure that the border node, and therefore the coupled proactive network, knows the mapping in advance. If this name was not needed by the border node and therefore not requested on the reactive side, or the border node has not overheard the response by accident, the route might not be stored in the border node's HAM table and is therefore not available in the proactive home network, too.

The second solution includes a newly introduced reactive scheme in the proactive mode. Finke et al. [61] used this for their route discovery mechanism in their adaptive routing framework. Therein, a proactive node having no valid route to an IP address in its routing table switches into a reactive mode just to send a RREQ to all known nodes coupling different routing zones. This is done via an IP in IP tunneling. However, this mechanism is only applied to a single MANET using adaptive routing - not to several MANETs coupled by border nodes.

We adapted the basic idea for our system. If a proactive node wants to resolve a hostname not being stored in the HAM table, it switches to a *Passive Border Node* mode and distributes a generated RREQ/NREQ to all known border nodes. This is done via unicast transfer (cf. Figure 5.16). If one of the regarding border nodes couples a reactive network, it can simply forward the request via the usual broadcast. The resulting RREP/NREP that might be received is then tunneled back to the requesting node, again via unicast.

If two proactive networks are coupled, and the name is not known to a requesting node in the home network, we obviously have to use the *Passive Border Node* mode again, if the border node is not distributing foreign network knowledge via OLSR messages. If we use an additional reactive behavior in the proactive mode, the requesting node contacts the border node, which might be able to answer the request immediately because of the in advance distributed knowledge in the foreign network. Otherwise, the request dies at the border node.

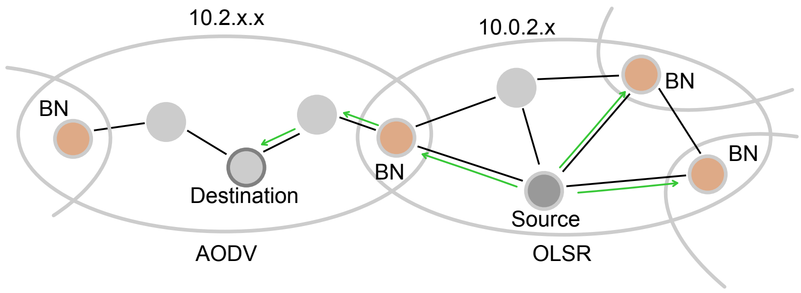

Figure 5.16: Example request from a node using the passive reactive mode [70]

For nodes being in the *Passive Border Node* mode, it is necessary that all current border nodes are known to all nodes in the network. This should generally be seen as a name resolution problem itself. For nodes in reactive networks, this problem does not apply as the request reaches the border node eventually. In the proactive mode, nodes coupling different networks have to inform the proactive side about that. We can see the provision of a gateway as a service and use any service discovery algorithm. In Section 7.2 we will show, how to realize service discovery based on our routing-based name resolution framework. However, as this functionality is somehow important and essential for inter-domain routing and name resolution, we decided to introduce a unique OLSR message instead.

The *Border Node Annotation (BANNO)* is periodically distributed by the border nodes using the intelligent flooding mechanism of OLSR. If the border node is a selected MPR itself, it just adds this message to the next bulk; otherwise, it sends the BANNO to self-selected MPR. The sending period is set to the same value as for the MPR messages and is treated in the same way in the intermediate nodes (cf. Subsection 4.4.3).

Figure 5.17 shows the structure of the new message. The *BNmode* field indicates the underlying routing protocol, if more than one proactive protocol is used. In our case, we only considered OLSR which is indicated with the value of *0*. Furthermore, the message contains a *Border Node Sequence Number*, which is incremented by the border nodes to allow the receiving node in the proactive network to recognize if the message should be processed or if it is out-dated. If the sequence number is lower than the one of an already processed BANNO, the packet is withdrawn.

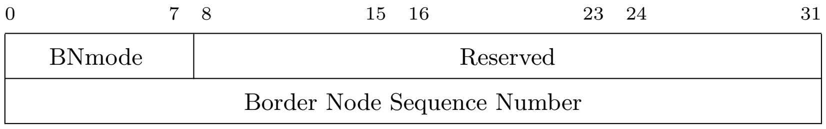

Figure 5.17: Border Node Annotation message structure [70]

The *Message Type* field in the OLSR message header which is shown in Figure 4.5 is set to *130* (cf. Message Type 128 for NADV messages and 129 for NERR).

5.8.5 Cascaded Networks

Additional to the already considered scenarios with only two coupled networks, we can beyond expect cascades of networks. Our system is designed to work in such situations, too. At each transition, one of the already mentioned mechanisms comes into force. Figure 5.18 shows an example scenario with five participating networks coupled by four border nodes.

Let us consider a requesting node residing in a reactive network. It distributes a reactive request. If this request reaches a border node there are three possible next steps.

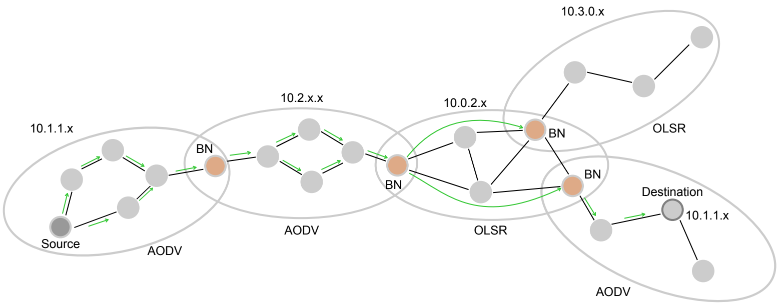

Figure 5.18: Example scenario for multiple coupled networks [70]

Firstly, the border node could answer the request if the neighbor network is proactive and provides the knowledge or if the neighbor network is reactive and the border node overheard the mapping by accident.

Secondly, the request could be broadcasted if the neighbor network is reactively. This broadcast eventually reaches the next border node, if existing, and is then handled the same way.

Thirdly, a unicast request could be sent to all known other border nodes if the neighbor network uses a proactive scheme. This is done via a unicast message as described in the subsection before.

The generated additional traffic is highly related to the scenario being present. However, we do not expect scenarios with a double-digit number of networks. Therefore, the impact on generated traffic and delay should not be significantly different from a usual single MANET.

5.8.6 IP Address Range Handling

An additional problem to be dealt with is the problem of same IP addresses in different subnetworks. If more than one subnetwork decides to use the same IP subnet mask and the same IP addresses for the participation nodes, our system so far could lead to false replies. If a nodes resolves a name, hosted in a foreign network, to an IP address that is also valid and used in the home network, it might be aware of the right local address but not of the right subnet or route. The underlying routing could overwrite the discovered route with an obviously better route to the home network node, which leads to a wrong route and a connection establishment to the wrong node. This could lead to a not erasable dead lock.

Figure 5.19 shows an example scenario, which leads to an address collision. The requesting node with the hostname *ID1* requests a route to the node with the hosted identity *ID3*. Due to the border node functionality, the hostname can be resolved to the address *10.1.1.5* over the next hop node having some other ID (in this case *ID2*). If the requesting node now initiates a connection establishment to *10.1.1.5*, the next hop node, having the same IP address, would try to answer, which breaks the whole system. Even if no connection establishment is initiated, the routing table entry is overridden by the next Hello message from the underlying OLSR protocol.

Assuming that most private networks automatically use one of the three private address ranges (e.g., *10.x.x.x*), such a situation could occur frequently, leading to over and over repeated name resolutions and making communication impossible.

A possible solution for this problem is to send along the network ID with every reply. An additional field in the HAM table could indicate, whether the regarding local IP address is located in the home network or in any foreign network. If the node is located in any foreign network, whether this is a neighbored network or one at some point in a cascade, the border node IP address must also be saved. As the border node address is definitely unique in the home network, the communication

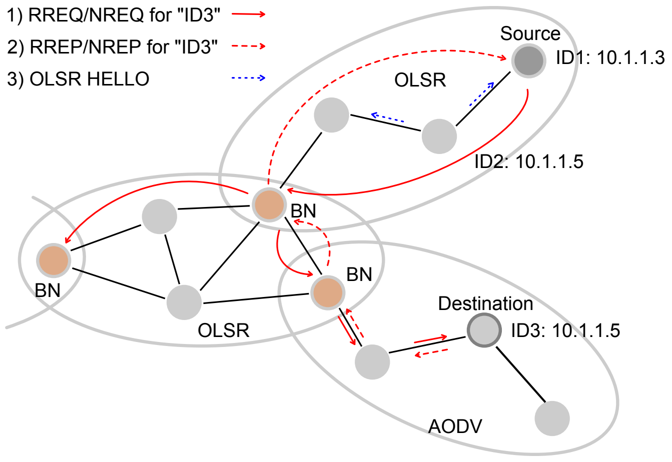

Figure 5.19: Example scenario leading to an address collision [70]

could be tunneled to the border node, while the gateway then forwards the packets to the right destination. Even in cascaded networks, this solution would work. If a border node receives a tunneled data stream which is not for a node within the neighbor network, it just tunnels the packets again to the next regarding border node until the destination network is reached. However, network IDs could also be not unique or simply not set.

In our previous publication [70], we proposed another solution using so called *Pseudo Addresses*. We let border nodes re-map the actual address of the destination node, if residing in a foreign network, to an address that is definitely not in use in the home network. This is done during the process that builds the reverse routing path. The border node stores this mapping in a table called *Pseudo Address Table (PAT)*. A name resolution initiated by the requesting node would therefore result in a mapping—and a route—to the pseudo address. If the requesting node then initiates a data connection, it sets the destination address to the pseudo address. The border node, over which the routing path goes, replaces the pseudo address with the real address according to the PAT table. In every packet coming from the node in the foreign network, the source address of the IP packet is changed to the pseudo address.

This scheme can also be used in a cascaded system with multiple networks connected over multiple border nodes. Every intermediate border node then allocates pseudo addresses as described above. This could lead to multiple pseudo addresses during one routing path and therefore multiple address replacements. However, until the path is disturbed due to the source and/or destination node switching the network or due to disconnected border nodes, this approach solves the address collision problem. If the path is disturbed, the routing process has to be restarted or, in case one of the communication partners changes the local address, the whole name resolution process itself.

A validation of the inter-domain name resolution framework is presented in Section 6.5.

5.8.7 Transition to a Network Using DNS

The framework proposed in this chapter is generally routing-based. We showed, how we can build name resolution schemes on top of different routing schemes, by the example of AODV and OLSR, and how heterogeneous networks are coupled.

However, we have to assume that the participating networks must support our name resolution if they want to participate in our name mapping framework. Networks built by arbitrary nodes without our software, however, might not be able to use the mechanisms properly because standard DNS is probably configured by default on their devices.

To overcome this problem, our border nodes should be able to couple foreign networks which use standard DNS. Engelstad et al. proposed, how their AODV-based works if interconnected to a network using standard DNS [42]. The question how to couple foreign name resolution protocols, no matter whether this is DNS or any other protocol like mDNS, is out of the scope of this thesis. Basically, at the edge of the network, border nodes or gateways have to convert any request to the format of the neighboring scheme. This leaves the realization of such mechanisms to the programmer of the border nodes, primarily some mobile ferries like UAVs.

5.8.8 Extending the Border Node Concept with Delay Tolerant Networking Technology

Our border node concept so far works if all subnets are connected. We basically assume that subnets run independent from each other but are coupled by at least one multihomed gateway. However, it is not realistic that all networks are bridged by border nodes all the time even if we assume intelligent node placement [71]. As rescue teams might get new missions in other areas, entire MANETs could move out of range. A name resolution, and therefore also the connection establishment, might fail again and again and leads to numerous retries until the source node gives up. In this subsection, we briefly discuss a concept to overcome such problems. We previously published some content of the following section in a conference paper [33].

One way to enable an inter-domain name resolution nevertheless is using Delay Tolerant Networking (DTN) [73, 74]. This technology allows successful data transmissions and communication even with a connection is not up and running all the time. By using ferries traveling from one network to another and uploading data if applicable, communication can even be established with permanently having no direct connection [75] by downloading the data in one network and uploading it in the other one.

In Figure 5.20, we show an example scenario with three networks, two directly coupled by an (ordinary) border node and two coupled by a DTN-enabled one.

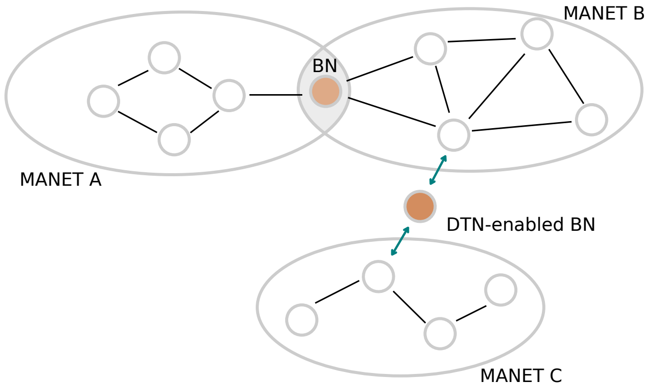

Figure 5.20: Example scenario with a DTN-enabled border node [33]

As ferries need seconds or minutes to travel from one subnet to another—depending on the physical distance—name resolutions over DTN backbones adds a huge delay some orders of magnitude higher than the usual basic delay which lays in a range of milliseconds. Furthermore, introducing DTN functionality let new questions and problems arise. However, for emergency communication and mission managements this is the only way to deal with spatially separated networks.

Additionally, in DTN unique End Point Identifiers (EIDs) are required, which fits to our naming scheme (cf. Section 2.5). The EIDs should consist of a prefix (*dtn:* for DTNs [76]) and a scheme specific part containing a unique logical subnet identification in addition to the actual node identifier. A full EID could be $dtn : rescue\_team_3 : node_2$. Due to the combination of network and node identifier in the EID, a border node can easily identify whether a node is within the same subnet and forward the message, if applicable.

DTNs allow late binding, which means that EIDs do not have to be resolved at the source node. This node tries to resolve only parts of the EID to forward the bundle accordingly. The final resolution is then done by the DTN-enabled border nodes forming a DTN backbone. It is obvious that source nodes do not use the DTN-backbone to resolve EIDs as long as they can find nodes within the same subnet or coupled by border nodes as described in the subsections before. If this initial process fails—i.e., after a timeout—the source node can forward the data packets to the DTN-enabled border nodes. The gateways then translate the packets to the DTN bundle and keep them until a forwarding opportunity occurs, e.g., after it comes in contact with another subnet or other gateways with higher delivery probability calculated by the gateways based on the contact history. However, this is up to the used DTN protocol and not in the scope of this thesis.

If the DTN-enabled border nodes moves to another subnet and gets connected, it contacts a local gateway and offloads all incoming messages while collecting outgoing ones corresponding to the used DTN protocol. The ferry must resolve the destination's EID to find a possible route. It is able to resolve the subnet part based on a distributed address table containing the EID as well as the corresponding subnet and the last location information. The ferries collect IDs and locations of all subnets they visited and include information collected by other ferries during the message exchange.

If the destination node is a gateway, its IP address is obviously resolved in this step. Otherwise, only the destination subnet ID can be resolved and the message is forwarded to the corresponding gateway. If the message finally reaches the regarding border node, it decapsulates the DTN bundle and resolves the local IP address according to our routing-based name resolution mechanism.

6 Simulation Results and Evaluation

In this chapter, we show simulation results to evaluate the framework we presented in the last chapter. We give an insight into the performance of the two presented schemes and compare key metrics to two selected approaches of the state of the art.

6.1 Simulation Metrics and Parameter

To evaluate the proposed scheme and to compare it with selected approaches from the state of the art, we chose to simulate the protocols with three different metrics. As simulation framework, we chose the Network Simulator 3 (ns-3) combined with the Click Modular Router (Click) as described in Section 5.2.

Additionally to the proposed reactive and proactive routing-based solutions, we implemented two reference protocols, i.e., DNS and mDNS (cf. Sections 3.1 and 3.2). For reason of simplicity, we implemented just the key features of the protocols to get the necessary data for the metrics of the simulation. Further functionality, like security issues, were neglected.

We chose to determine the following three metrics, i.e., the **generated traffic**, the **delay** until the name is resolved, and the **success rate**.

Generated traffic is important to discuss the network load and therefore the energy consumption of the participating nodes. Energy is a major issue in MANETs, especially while used in disaster scenarios. The less traffic we generate, the less energy for data processing and transmission is consumed. Furthermore, we prevent interferences which could cause multiple retransmissions. For this metric, every packet transmission is counted as traffic as long as this packet is caused by the name resolution scheme. Packets that would be generated even without the used scheme are not considered. For instance, if one unicast packet carrying name mapping information (e.g., a RREP) is sent from Node A to Node B over Node C, it has to be transmitted two times and is counted twice.

The measured delay can be used to evaluate how fast our system works. Some applications are time-critical, e.g., mission planning algorithms. If nodes are found to late, mission requests could be sent to another node with worse properties, e.g., a higher distance to the destination or less available energy. As starting time of the measurement, we take the appearance of an unresolved hostname needed by the Kernel. We stop the time at the point where the name is successfully resolved to an address and a route to the address is found.

Delay and traffic often have an opposed behavior. The more traffic we spend, the less delay we can achieve and vice versa. In the following sections, we show how this behavior occurs due to the usage of our two routing-based schemes.

Measured traffic and delay have no meaningfulness without the success rate. If the delay is measured for successful requests only, we can achieve very low values but maybe just for a few percent of the cases while the others failed. However, assuming an infinite delay for unsuccessful requests is a non-plottable value. A success rate of 100 % might not always be achievable as nodes might not be present in the network or interferences disturb the message transmissions.

For our simulations, we always used the same parameter setting, independent from the name resolution protocol. We chose IEEE 802.11 (named Wireless Local Area Network (WLAN) or Wi-Fi, in this thesis WLAN) to run in the Physical and Link Layer, respectively. 802.11 is currently the most used technology for local wireless communication and provides an ad hoc mode. We used the YANS wifi channel implementation [77] available for ns-3 in the current version.

A list of the (fixed) parameters that were the same for all name resolution schemes is shown in Table 6.1.

Parameter	**Value**
Area	1000 m x 1000 m (max)
Number of nodes	10 to 100 by 10
Node speed	1-20 m/s (random)
Simulation time	80 s
Tx range	150 m

Figure 6.1: Simulation parameters

The simulation area was adapted to the number of nodes in order to realize a good connectivity, with a minimum size of 350 m x 350 m for 10 nodes and a maximum at 1000 m x 1000 m. By adapting the network size, we prevent the scenario from both: all nodes being on one place and therefore forming a fully-meshed network and all nodes being too far from each other and therefore having no connectivity at all. Even though we realized an (almost) static connectivity level for our networks, the resulting curves might show a small break at the points of adjustment.

The number of nodes was changed from 10 to 100 nodes. Studies on disaster scenarios showed that those are realistic numbers [78]. The movement speed of the nodes is randomly chosen between 1 and 20 m/s, which reflects the range of walking persons up to driving vehicles or flying multicopters. Furthermore, the direction of the moves are also generated randomly. We determined other more realistic movement patterns following some structure for dense and sparse scenario [78]. We showed that using realistic movement and traffic pattern increases the overall performance of the network. However, for the evaluation of our name resolution approaches we assume movement following the the Random Waypoint (RWP) model implemented in Bonnmotion to proof that our protocols work even with unusual movement considered.

The number of name resolution requests was increased in tiers with increasing number of nodes to reflect the increased demand in increasing network size and base the success rate metric on more data. Table 6.2 gives an overview of the configuration.

Number of nodes	**Number of requests**
10 to 20	2
20 to 50	5
60 to 100	10

Figure 6.2: Number of name resolution requests

For every setting, we did ten independent runs to be able to show an averaged value with a standard derivation.

The performance of the routing-based schemes is highly dependent on the routing parameter setting. We used a standard adjustment here. The AODV Hello period was fixed to 1 s. The OLSR Hello period was set to 2 s, while TC messages are distributed every 5 s. The distribution periods of the name mapping in the proactive scheme are set to the double of the routing periods, which results in a NADV message every 4 s and extended TC messages every 10 s.

6.2 Performance Evaluation

In this section, we want to evaluate the overall performance of our routing-based schemes as described in Chapter 5. The evaluation shows the impact of the routing-based schemes on the routing traffic in the network. Due to our integrated name resolution, the actual traffic caused by the routing protocols increases, which results in a higher load for the network. In this section, we compare the name resolution overhead with the routing traffic, which occurs independent from our extension. We previously published some content of the following section in a conference paper [67].

Figure 6.3 shows the generated traffic of the AODV-based scheme in comparison to the general routing traffic. We changed the number of nodes from 10 to 100 with a step size of 10 nodes, which is assumed to be a usual size for a network in a disaster scenario [78].

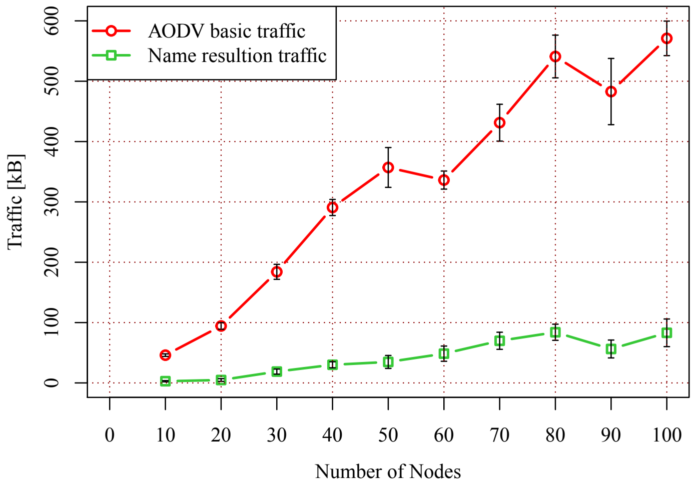

Figure 6.3: Evaluation of the traffic in the reactive scheme

The figure shows that the additional traffic is less than the basic AODV traffic caused by AODV Hello packets. The name resolution traffic ascends with increasing number of nodes, as the number of requests increases, too. Furthermore, the Hello traffic rises linearly as more nodes result in more Hello packets. Hello messages are only sent to the direct neighbors for the neighbor discovery.

We have not set any data traffic in this simulations. This would automatically lead to more routing traffic, as the two communication partners have to establish a route beforehand. Furthermore, the system is without previous knowledge. Previously performed name resolutions or route discoveries would potentially further decrease the traffic.

The generated traffic is obviously highly depending on the number of requests. If all nodes have frequent requests, the additional traffic might explode. The traffic should not scale with the number of nodes, as the broadcast leads to an exponential growth (as can be seen in the figure).

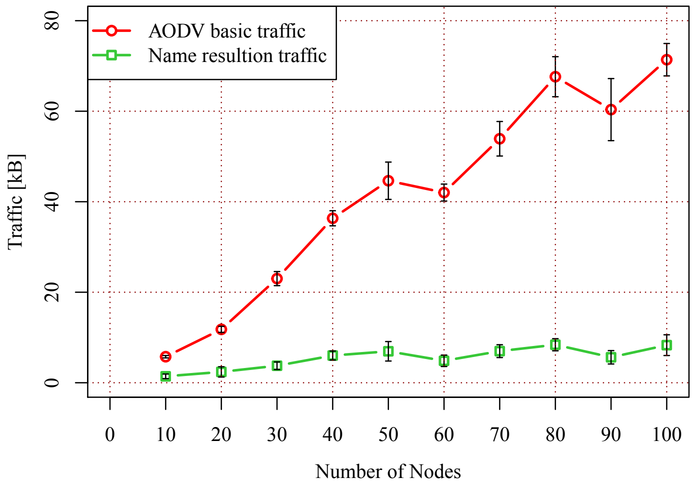

Figure 6.4: Evaluation of the normalized traffic in the reactive scheme

Furthermore, the simulation time was set to 80 sec to give the reactive scheme enough time for performing the request-response-process considering that the nodes initiate the requests at different (random) times of the simulation greater than zero. The Hello interval is by default [68] set to 1 sec, which results in eight observed Hello's within the simulation time.

To give a general statement about the traffic impact, we normalized the traffic in Figure 6.4. The AODV Hello traffic was normalized to one period, while the additional name resolution traffic was normalized to the number of requests (cf. Table 6.2).

The figure shows that one name request procedure generates significantly less traffic than the routing within one Hello period. Furthermore, the name resolution traffic is almost constant even though the number of nodes increases the request broadcast traffic.

An evaluation of the delay and the success rate metric is shown in Section 6.4 in comparison to other approaches of the state of the art.

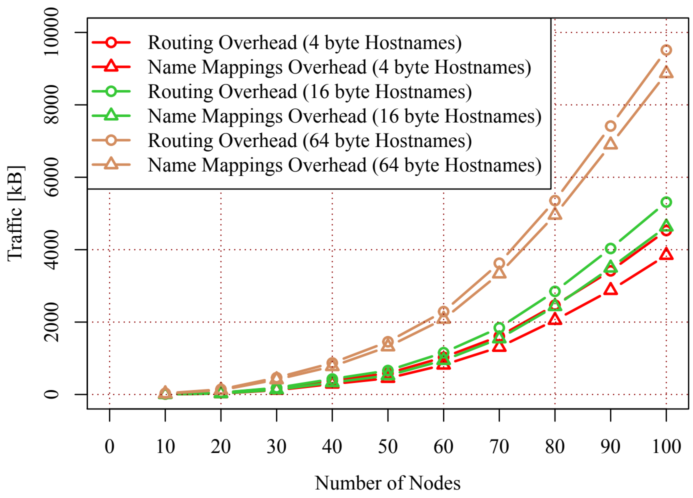

Figure 6.5: Evaluation of the traffic in the proactive scheme

In Figure 6.5, we show the traffic impact of the proactive name resolution scheme in comparison to the basic OLSR traffic.

The figure shows the pure OLSR traffic generated due to Hello and TC packets and the pure name mapping distribution due to NADV and enhanced TC messages. One can see, that the additional name resolution traffic is less than the usual OLSR traffic. On the other hand, that means that we almost double the basic traffic with our name mapping distribution. This is caused by the length of the NADV messages.

In the simulation, we let each node have one hostname acting as an identity. Each hostname had the same length, which is subsequently increased. We chose three scenarios with hostnames of 4 Byte, which is the same length as an IPv4 address, 16 Byte, which is the same length as an IPv6 address, and 64 Byte.

6.3 Comparison with other Routing-based Schemes

In this section, we want to compare the performance of our reactive scheme with the model proposed by Engelstad et al. [41]. As described in Section 3.3, Engelstads approach waives caching of overheard replies to answer future requests by intermediate nodes.

As therefore only the destination node is able to answer, we expect a higher traffic and a higher delay using this implementation. To proof that we can increase the performance with caching, we simulated the same scenario with and without saving knowledge in intermediate nodes activated. We previously published some content of the following section in a conference paper [79].

We used the same parameters and metrics as described in Section 6.1 and did ten independent runs in order to plot a standard derivation. We let random nodes ask for random hostnames. The several requests were initiated at different simulation times during the simulation run. However, for both schemes the same node couples and request times were used in order to compare the impact of the different implementations probably.

Figure 6.6 shows the comparison of the generated additional traffic. Without caching, intermediate nodes are without any knowledge all the time. This means that every (broadcast) request must be forwarded until the destination is reached and responds. Due to the caching, intermediate nodes can answer requests before the NREQ reaches the actual destination. The broadcast is then skipped, which decreases the traffic significantly.

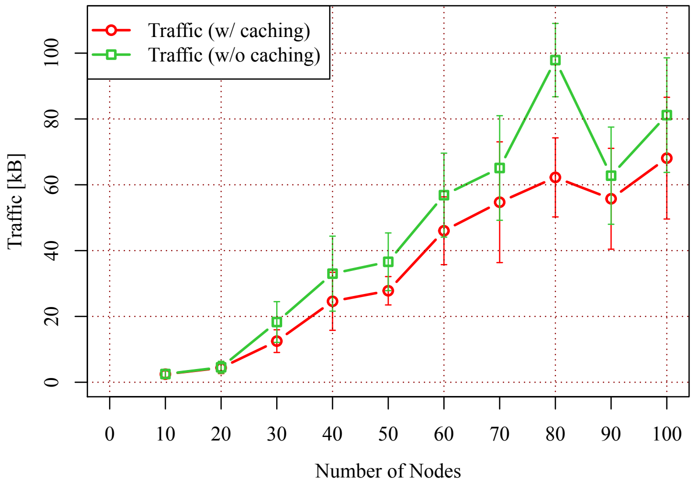

Figure 6.6: Comparison of generated traffic for reactive name resolution w/ and w/o caching [79]

Furthermore, the unicast reply might need more hops until it is back at the requesting node. Every transmission from hop to hop is counted as traffic here. The figure shows that we can save around 10 kB traffic during the simulation run.

Figure 6.7 shows the comparison of the achievable delay for the same configuration as described above. Due to the knowledge distribution, intermediate nodes might answer requests faster than the actual destination. Therefore, we can stop the broadcast earlier. AODV come with the Expanded Ring Search mechanism, which increases the range of the broadcast if the request was unsuccessful. This mechanism is processed in the source node, which stops after a successful reply, too.

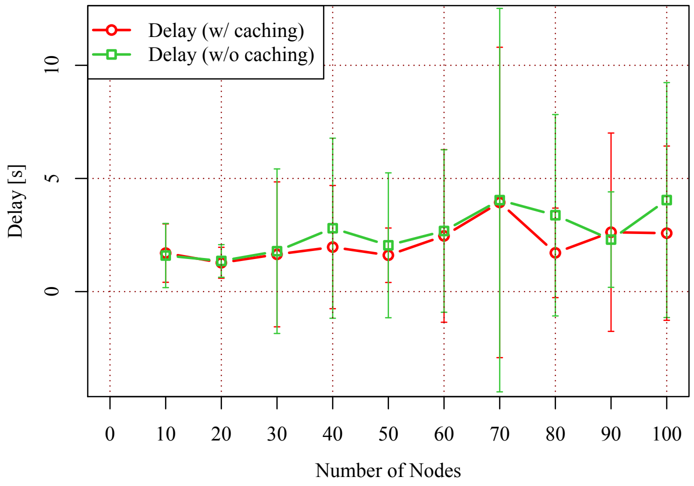

Figure 6.7: Comparison of delay for reactive name resolution w/ and w/o caching [79]

Furthermore, we also have less number of hops for the backward route in that case. The figure shows that the delay for the mode without caching is permanently higher for up to 1-2 s. It should be noted, that the delay is highly depending on the number of AODV resends. As we used the same configuration and the same node couples for both implementation, this effect should have no effect on the curves. One can see, that the traffic impact is much higher than the impact on the delay. That is because the broadcast request has more impact on the traffic than on the delay.

In Figure 6.8, we show the comparison of the success rates. A name is successfully resolved, if any reply reaches the originator. Due to the distributed knowledge for the name mapping and the routing, respectively, intermediate nodes might answer even if the destination is currently not reachable. This could either occur due to network partitioning or because of failed packets because of interferences.

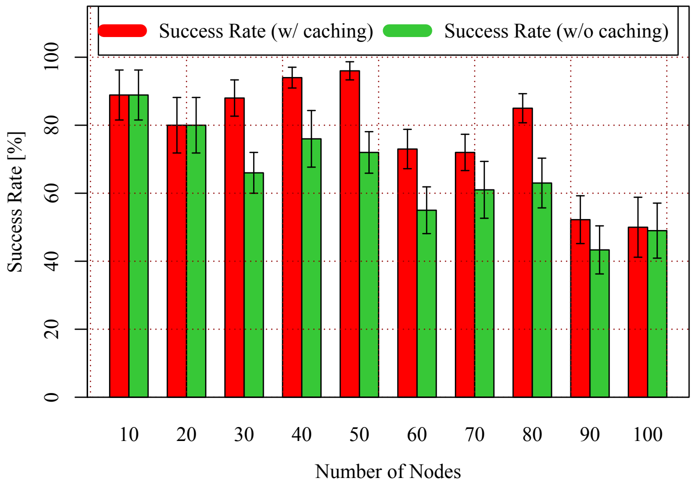

Figure 6.8: Comparison of success rate for reactive name resolution w/ and w/o caching [79]

The plot shows that the success rate is comparable for small and big networks. For networks between 30 and 90 nodes, we can achieve significantly higher success rates with caching activated, with a 20 % peak for the scenario with 50 nodes.

6.4 Comparison with Application Layer Approach

In this section, we want to compare the two routing-based schemes to name resolution protocols located in the Application Layer. As discussed in Section 3.4, we see the routing-based solutions as the most promising solution for disaster scenarios. This section proves that we can outperform the two selected Layer 7 approaches, i.e., DNS and mDNS. We previously published some content of the following section in a conference paper [79]. The mDNS implementation was supported by the Advanced Research Project written by Usama Sallakh [80] during his master studies.

In the first plot (cf. Figure 6.9), we compared the (additional) traffic caused by the several name resolution approaches.

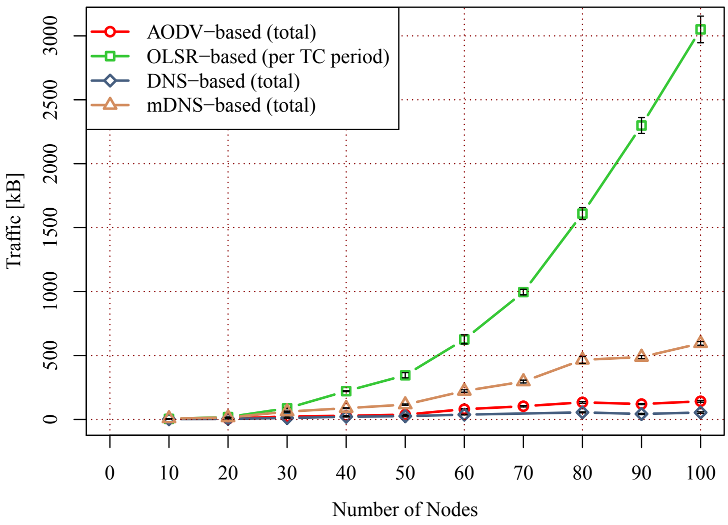

Figure 6.9: Comparison of generated traffic [79]

We considered only the traffic that was caused additionally to the basic traffic. In the reactive routing-based scheme this is the request and reply communication, in the proactive scheme the NADV Hellos and the additional overhead of the extended TC messages. In the DNS simulation, we counted the unicast communication between source and server as well as the AODV packages needed to find the route to the server and to the destination, respectively. For the mDNS protocol, both broadcasts, i.e., the request and the reply process, are taken into account.

One can see that the traffic caused by DNS is the lowest. This is due to the fact that the DNS request and the reply are both sent by unicast, while only the route discovery for finding the server and the destination is broadcasted. However, the reactive routing-based scheme lies only slightly higher, even though the whole name request is carried in the RREQ packets. Furthermore, we have to take into account that the discovery process using DNS is stopped if the DNS server is not found during the AODV route discovery phase. This reflects in significantly lower success rates but decreases the traffic, too.

The OLSR traffic is plotted normalized to one period of TC distribution. As OLSR traffic is periodic, we could otherwise change the heights of the curve by adjusting the simulation time. The generated traffic of the proactive scheme is the highest in the observed cases. However, this drawback comes with a benefit for the delay. The curves shows an exponential growth, which fits to the general assumptions. For scenarios with 10 to 40 nodes, the additional overhead is decent. For higher number of nodes, a switch to reactive routing is advisable, assuming an adaptive routing framework.

The mDNS traffic lies between the one for the reactive and proactive scheme. As the request and the reply is broadcasted to all nodes in the network, the generated traffic is more than double as big as for the AODV-based scheme. However, due to the distributed name mapping knowledge, we prevent potential future requests and also decrease the delay as future requests are answered mit zero delay.

In Figure 6.10, we plotted the delay until a name was successfully resolved for all protocols against changing number of nodes. Failed requests are not counted in this plot. For the DNS protocol, we measured the delay until the name mapping is found and the route to the destination is established. In the routing-based schemes, name mapping and route information are always found together.

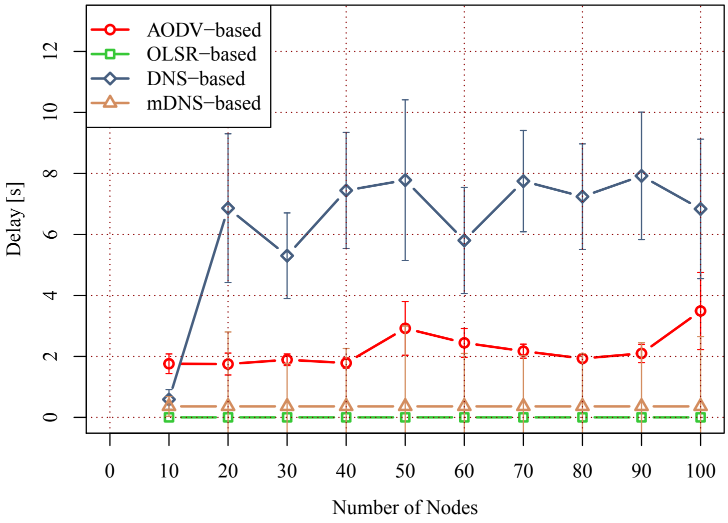

Figure 6.10: Comparison of delay until name is resolved [79]

The delay for the OLSR-based scheme is zero by default. Either the hostname is already in the HAM table of the requesting node or the request was unsuccessful. Such requests are reflected in lower success rates.

The delay for the DNS system—which lies in the range of around 8 s—is around double as high as the AODV-based delay, being around 3 s. Here, we have to find a route to the DNS server first, which can itself lead to almost the same delay as the total delay in the reactive scheme is. Secondly, we need to wait for the unicast traffic to proceed and, thirdly, we need to find the route to the destination. The observed behavior fits to our assumption made in Subsection 5.1.2 (cf. Figure 5.1).

The averaged delay in the mDNS is lower than in the reactive scheme but higher than the DNS delay. This is due to less request resends. If the first request is unsuccessful, we let mDNS and the reactive scheme send another request after 1 s. This is repeated five times until the node gives up. In mDNS we observed less resends resulting in less (average) delay. This could be caused by the more distributed knowledge due to the broadcasted replies or because the reply transmission is prone against interferences, as replies can reach the destination over different paths and resends are prohibited.

Finally, we compared the success rates of the several protocols in Figure 6.11.

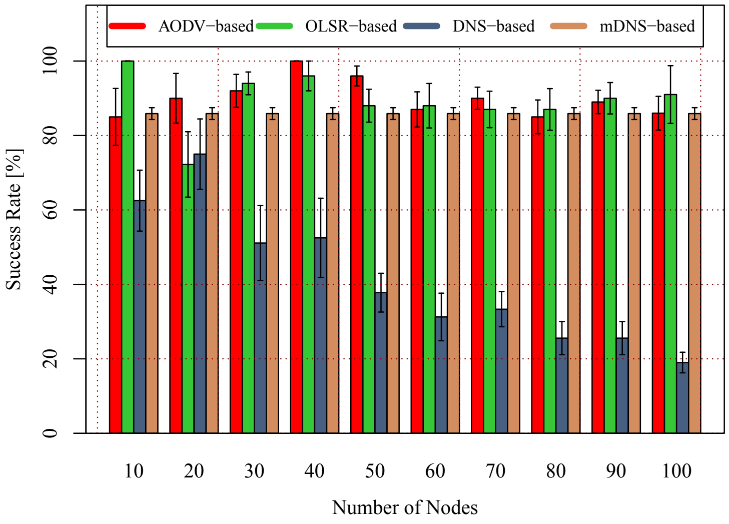

Figure 6.11: Comparison of success rate [79]

A name request is successful resolved, if the node found an IP address to a given name and a route to the found local address. In case of AODV used, this is achieved if a RREP/NREP packet arrives; in case of OLSR if there is an entry in the HAM and routing table, respectively. For DNS, we counted a success if the server is found, the name is resolved by the server, and a route to the destination was found.

The figure shows that we can achieve between 80 and 100 % in the reactive scheme for all observed numbers of nodes. The proactive scheme and mDNS deliver comparable success rates. This is due to the fact that all three protocols are generally broadcast-based. In case of the reactive scheme and mDNS, the request is broadcasted; in case of the proactive scheme, the name mappings are distributed via broadcast. If there is a connection between the nodes, we can resolve the name.

For very small networks of just 10 nodes, the value for proactive routing is even higher than in the reactive mode. This fits to the general routing recommendation, which says that proactive routing should be used for small networks and reactive routing for big size ones. The success rates for the client-server-based structure using DNS shows the lowest success rates with down to less than 20 %. This is because all three steps of the resolution could fail. The requesting node could either fail to find the server, or packets could be lost during the resolution process, or the destination could be unavailable for route establishment.

Generally spoken, we advise the proactive routing scheme for scenarios with a small amount of participating nodes, low requirements on battery consumption or network load, and time critical applications because of the low delays. For big size networks with a low mobility, the reactive scheme performs best. This advice can be used as input in the formula to calculate routing decisions in the switching decision maker of the adaptive routing framework. Furthermore, the overall performance of the name resolution or the actual requirement can be used as an input to make decisions, too.

6.5 Performance Evaluation of the Border Node Concept

In Section 5.8, we described our border node concept to couple different heterogeneous networks for inter-domain routing and name resolution. In this section, we want to deliver a brief evaluation of this mechanism. The simulation results presented here were previously published in a conference paper [70].

The parameters are the same as described in Section 6.1, though we simulated an area of 1500 m x 1000 m. The larger area was needed, because we did not just simulated one MANET but several networks and needed a bigger area to create spatial segregation between the several independent subnets.

The simulated scenario is shown in Figure 5.18 containing five subnets in a cascade coupled by four border nodes. Within every single MANET, nodes move with a random speed between 10 and 20 m/s, while the border nodes stay at a fixed position to create stable connectivity. We did 20 independent runs to calculate an average value and the standard derivation.

The first determined value is the delay. In Figure 6.12, we calculated the delay between source and destination against the number of hops or number of nodes in one single network, respectively. In this simulation, the location of the nodes were fixed while we added additional nodes between source and destination. The curves show the time until a reply for a sent request was received.

The figure shows a linear dependency if low packet drop rates are assumed. With increasing number of hops the standard deviation increases, too. However, the results show only the intra-domain delay within one MANET which only depends on the number of hops between two nodes, e.g., two border nodes. Additional nodes within the network might disturb packet transmissions which leads to more retransmissions and therefore a higher delay.

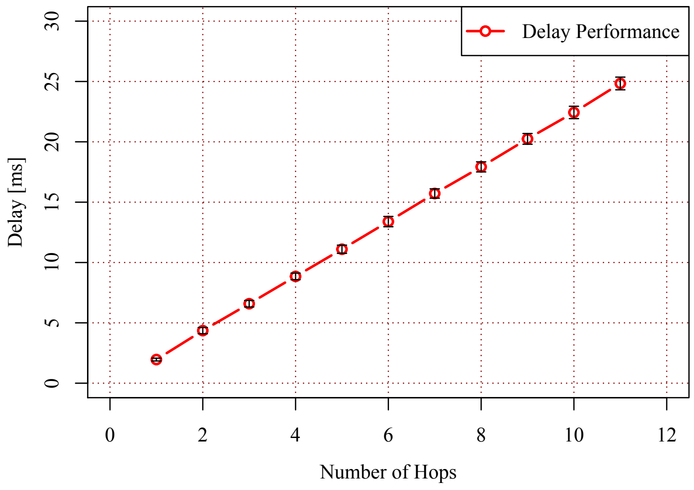

Figure 6.12: Delay performance for fixed nodes and changing number of hops [70]

In Figure 6.13, we plotted the delay—again within one subnet—and increased the number of additional nodes moving with a random waypoint model.

The observed delay seems to be almost constant even for high number of nodes. That proves that the average delay only depends on the actual number of hops and not on the number of nodes in the network. However, during the simulation we observed increasing number of retransmissions caused by the occurring interferences of the additional nodes and multipaths due to the node density. The red curve shows the intermediate delay if we forbid retransmissions and ignore the resulting failed resolutions. The green curve shows the intermediate delay if resends are allowed, of course based on a higher set of measurements as the red curve.

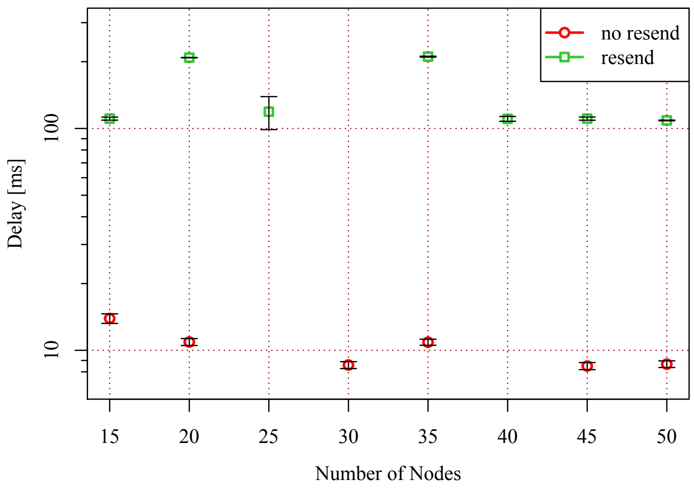

Figure 6.13: Delay performance for moving nodes [70]

It can be seen that the delay is increased by ten times (y-axis has logarithmic scale). This is because we set the time until a resend is triggered to 100 ms in our protocol. Re-transmissions occurred in 43 % of the simulated scenarios.

We also identified an increasing number of answered requests by intermediate node's HAM tables because previous requests might have reached the destination and the reply failed on its way to the originator. All intermediate nodes, however, had already stored the information in their HAM tables and are, therefore, able to answer the resent request immediately. This has has create influence if we assume a cascade of several networks that has to be traveled by both request and reply.

Missing points in the green curves mean that no resend was needed for all requests, missing points in the red curve correspond to cases where all requests failed in the first attempt.

In the next step, we simulated not only one MANET but the whole scenario showed in Figure 5.18. We decided to simulated two extreme cases with all subnets either running AODV or OLSR. In Figure 6.14, the red data points are for the resulting delay for the proactive-only scenario and the green ones for reactive-only routing. Given the presented protocol design for our border node concept, the AODV-only curve should be the same as for the example configuration. The overall delay is only depending on the routing protocol running in the last subnet. The delay for the home and the intermediate networks does not depend on the routing but on the number of hops and the interferences as validated in Figures 6.12 and 6.13 because it does not matter whether the request is broadcasted in the neighbored network, if reactive, or whether it is sent unicast from border node to border node, in the proactive case.

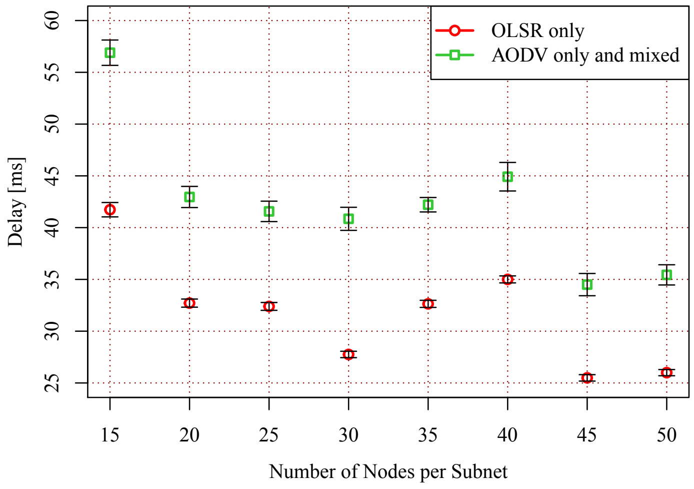

Figure 6.14: Overall delay performance depending on employed routing protocol [70]

The figure shows that we can achieve lower delays if the last network is proactive because the corresponding border node can answer the request immediately because of the previously learned mappings while in the reactive case the request-reply-mechanism has to be processed. However, the delay for the case that AODV is running in the last network is just slightly higher but with a bigger variance.

In the last plot, we measured again the total delay for our example scenario but this time depending on retransmissions. The green data points in Figure 6.15 show the worst case delay we measured if multiple retransmissions occurred, while the red ones represents the best case.

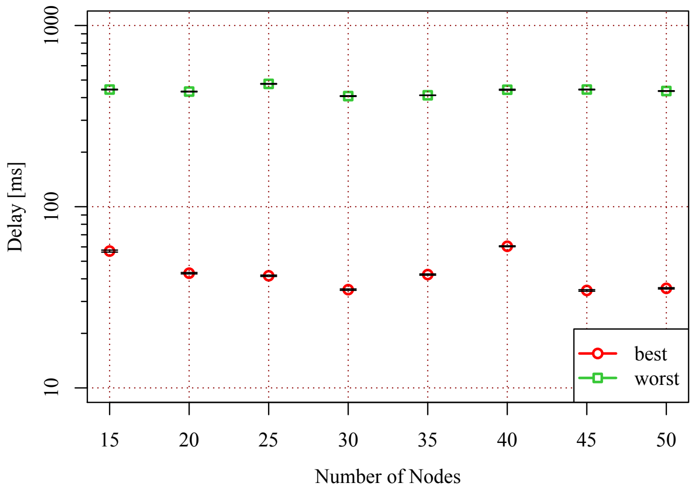

Figure 6.15: Overall best and worst case delay performance [70]

The worst case delay is a tenth of the best case one (notice the logarithmic scale). As showed in the figures above, one retransmission causes an additional 100 ms, which adds up to this high number if occurring frequently. However, if we allow retransmissions we can increase the success rate significantly. Especially in disaster scenarios with highly mobile nodes it is important to find other nodes at all.

In our tests, we did not consider failed routes due to high mobility or node failure. If intermediate nodes or even border nodes fail or disappear, the discovery mechanism has to be restarted again. Furthermore, it is obvious that our mechanism does not result in a reply if there is no connection at all.

For the evaluation of DTN-enabled border nodes, we enhanced the simulation scenario shown in Figure 5.18 by spatially disconnecting some subnets as shown in Figure 6.16.

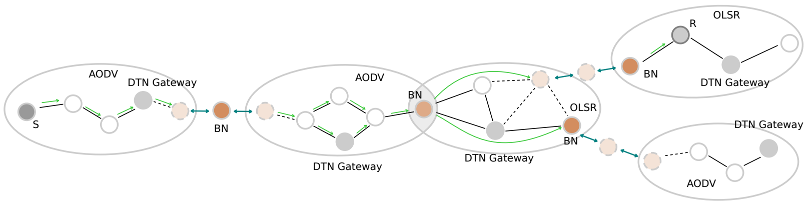

Figure 6.16: Enhanced scenario with DTN-enabled border nodes [33]

The border nodes now act as ferries oscillating between two subnets and forwarding messages as described in Subsection 5.8.8. The distance between the MANETs lay between 300 and 600 m, the distance between the several nodes between 80 and 100 m. The ferry speed was randomly chosen between 0 and 6 m/s, the speed of the non-ferry nodes between 0 and 1.5 m/s. Mobility files were again generated with BonnMotion.

To simulate MANETs and DTNs at the same time, we coupled our ns-3-Click simulation framework with the Opportunistic Network Environment (ONE) [81] simulator, which is designed for DTN simulation without considering the underlay. The movement files used in ns-3 were converted to the ONE format in order to have the same scenario simulated in both simulators. While all nodes are part of the underlay simulated in ns-3, only the DTN-backbone formed by ferries and gateways are part of the ONE simulation. The ferries moved between the MANETs they coupled with controlled movement including a random period of staying at each subnet for uploading and downloading data such as name requests and replies.

The first parameter we measured was the delay. The overall delay mainly depends on three sub-delays, i.e., the intra-domain delay which was determined above, the time the ferry needs to move from one subnet to another, and the mentioned waiting time in each subnet.

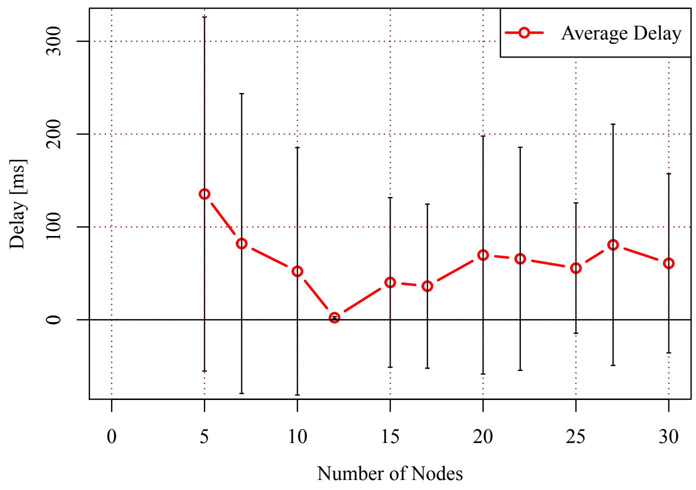

Figure 6.17: Worst case average intra-domain delay [33]

First, we simulated the intra-domain delay caused by the routing-based name resolution mechanisms. Figure 6.17 shows the averaged delay for the given scenario assuming that all subnets are fully connected without DTN nodes for different numbers of nodes and for the worst case scenario where source and destination lie on the subtend sides of the scenario.

The figure shows a delay in the range of 0 to 300 ms. With increasing number of nodes, the curves drop as new nodes build better paths through the network. After the breaking point, the delay increases again due to caused congestion.

Second, we measured the delay introduced due to physical ferry movements between MANETs for the given scenario (cf. Figure 6.18). The resulting delay consists of two components, the travel time of the ferry and the transmission time needed to offload and upload data. The movement speed was varied between almost 0 and 25 m/s.

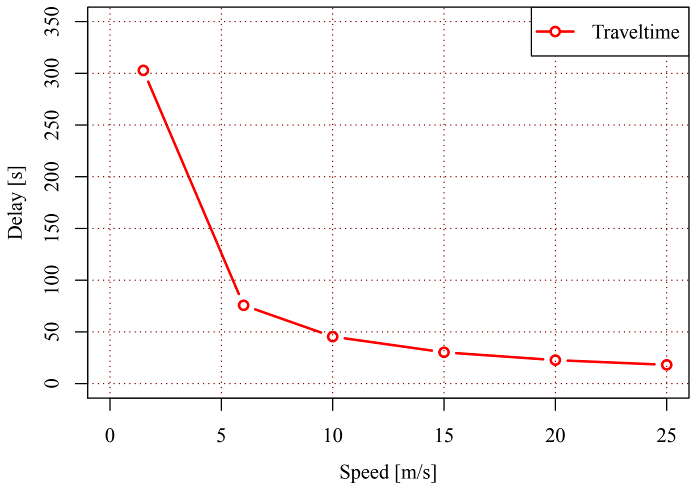

Figure 6.18: Delay caused by ferry movement [33]

An decreasing delay while the movement speed is increased is obvious for the given configuration. However, the figure shows an exponential behavior. This is caused by the additional effect, the movement time has on the transmission time. If the ferries move faster, they potentially reach the several subnets more often. That means that the ferries can exchange data more often while freeing their buffer. Therefore, the needed transmission time per ferry meeting decreases according to the increases movement speed.

Comparing the delay caused by ferry movement shown in Figure 6.18 to the intra-domain delay presented in Figures 6.17, the major proportion of the overall delay is caused by the ferry movement here. While the intra-domain delay is in the range of milliseconds, the movement delay is some seconds or even minutes high. Hence, the used routing protocol in the several subnets does not significantly affect the overall delay and is therefore negligible for DTN communication.

7 Extensions to the Basic Mechanism

In this chapter, we show some extensions to the basic framework explained in the preceding chapters. We show, how the hashing of the transmitted host names can reduce the traffic consumption. Furthermore, a service discovery mechanism based on the name resolution framework is described. Finally, the introduction of location information to improve the resolution of names is shown.

7.1 Host Name Hashing

In this section, we deal with the idea of sending hashed hostnames instead of full-text ones. This was invented to reduce the name resolution traffic during the discovery process. We previously published some content of the following section in a conference paper [82].

7.1.1 Basic Idea

In the current system, the hostname that needs to be mapped to the local IP address is sent as it is: as a standard string. We call such hostnames being sent **full-text**, **plaintext**, or **as a whole** interchangeable in this section. In the reactive mode, we send the hostname in full-text in both the NREQ and the NREP. It is sent in the NREQ to request a certain name mapping and in the NREP to distribute the mapping in the network. We could skip the name in the reply because the originator knows the name it asked for. But the drawback would be a loss of distributed knowledge to answer future request by intermediate nodes. In the proactive mode only one message type, the NADV, is broadcasted also with the plaintext name.

We consider hostnames as being not greater than a Fully Qualified Domain Name (FQDN), which has a maximum length of 255 Byte. Names greater than 255 Byte do not fit to the requirement that the name should be human readable. However, reaching the maximum leads to huge packets and massive generated traffic. Especially in the proactive mode, where name mappings are distributed periodically, this could lead to a high (periodical) network load.

An idea to reduce the load is using hashing algorithms to map the name to a defined length. Hashing is used to compute a hash key out of any input. The hash key has the same length no matter what the input is while the input could be of any length. Prominent examples are the Message-Digest Algorithm 5 (MD5) [83] and the Secure Hash Algorithm (SHA) [84]. We chose MD5, which maps names to a hash key of 128 bit or 16 Byte, respectively. This is exactly the length of an IPv6 address. The protocol proposed by Ge et al. [85] uses indirect-addressing for their service discovery protocol. This is the same basic idea of name hashing that is used in our framework, too. However, our approachs aims at the reduction of generated traffic—an effect that was not evaluated at all in this paper.

The biggest part of a name resolution message in our model is the name itself, because it is stored as a string where every letter takes a byte. The other fields, like lengths or flags, only consume limited storage. Therefore, limiting the name field could save much traffic. In the best case, a 255 Byte FQDN is downsized to an only 16 Byte long hash key, if MD5 is used. This results in a saving of up to 93.75 %. It is obvious that the hashing of names smaller than 16 Byte would result in a bigger packet than sending along the plaintext. So it makes no sense to use hashing for such small names.

The question is how communication is possible based on hashed names in the requests or advertisements, respectively. Computing a hash key is irreversible. That means that receiving nodes cannot compute the full text name out of the received hash. But one fact helps to communicate only based on hashes, i.e., that the same input always results in the same hash key no matter on which node the MD5 hash is computed. Every node taking part in the network and being aware of the name resolution framework proposed in this thesis can compute the unique hash key for every plaintext name it knows. This could be done either if a name resolution message containing a hashed name is received or in advance. As storage is cheap, we decide to extend the HAM table and include an extra field for the hashed hostname, which is added to every HAM table entry.

If one node receives a NREQ in the reactive mode or a NADV in the proactive mode, where it is indicated that the hashed name is sent along, it just compares the received hash key with the hash keys stored in its HAM table to create a NREP or store the new information, respectively.

Even if the full-text name is unknown, nodes can store a received HAM table entry in the HAM table without the regarding full name, by just leaving this field empty. If afterwards further requests reach the node, it can compare the stored hash key with the received hash key directly or compute the hash key of a received full-text name and compare the result with the stored one. Therefore, it can answer requests even without knowing for which full-text hostname the request was sent.

However, it has to be ensured that every node is able to identify, whether a hashed name or a full name is stored or sent, repectively. This has an influence on the structure of the three messages, i.e., NREQ, NREP, and NADV, as well as the HAM table. In the following subsection, the new structures and procedure for the protocol mechanisms are presented.

7.1.2 Implementation

As described in the last subsection, we assume that nodes can choose between sending the hashed name or the full-text one. We showed that both variations basically work. It is also possible that both are used at the same time, e.g., by sending the full-text name in the NREQ and the hashed name in the NREP. However, some changes in the implementation presented in Chapter 5 have to be made.

The backbone of our framework consists of a table where all name mapping information is stored, called HAM table. In this table, all gathered information is stored for own future use or to answer upcoming requests from other nodes. However, if we let every node the choice to either send the full-name or the hashed one along with the request, the table structure has to be accommodated accordingly.

Figure 7.1 shows the new structure of the HAM table to store name mapping information.

data type	**field name**
string	hostname
uint8_t	hostname_length
bool	hostname_type
string	hashed_hostname
bool	address_type
uint8_t	address_length
IPAddress	local_address
bool	flag_multihomed
bool	flag_gateway
timestamp	time_to_live

Figure 7.1: Structure of accommodated HAM table entry for hashing system [82]

Compared to the old implementation of the HAM table as shown in Figure 5.7, we added the new field *hashed hostname* which can be used to store the hashed version of the full-text name, which is still stored in the *hostname* field.

As mentioned above, every node having the full-text name, automatically computes the MD5 hash and stores it in the HAM table. This ensures that every node can quickly searches its HAM table for an entry matching to a currently received hashed name request or advertisement.

A special case is, if the node receives a NREP or NADV, respectively, with included hashed name and without having any knowledge of the plaintext name. In this case, the full-text *hostname* field is set to the *0*. To indicate that this is no full-text version of the name, we further introduced the *hostname type* flag. If this is set to *1*, no plaintext name is available. If later the nodes receives information about the full name it replaces the value in the *hostname* field and sets the *hostname type* flag to *0*.

The downstream completion of an incomplete entry can either be done by NREPs or NADVs, respectively, but also if a NREQ is received. The only necessary information needed is the the full-text name. By hashing the hostname and comparing the hash to all those hash keys stored in incomplete entries—indicated by the *hostname type* flag—the missing entry can be eliminated, if applicable. We implemented such a procedure in the regarding Click elements.

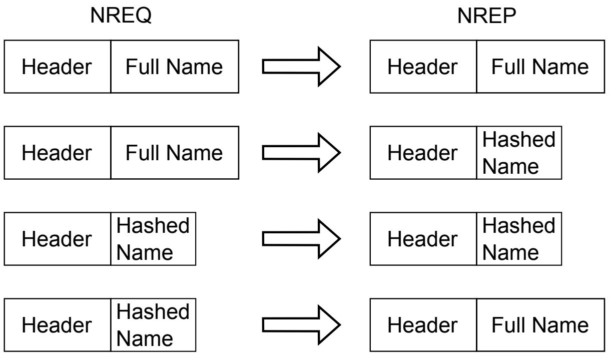

Figure 7.2: NREQ and NREP w/ and w/o hashed hostnames

Beside the storage in the HAM table, it is also necessary to change the structure of the three messages used by the reactive (NREQ and NREP) and proactive (NADV) mode. We changed the implementation of all three messages by adding flags indicating hashed names optionally. In the reactive mode, this results in four possible variations shown in Figure 7.2.

The structure of the new NREQ message is shown in Figure 7.3.

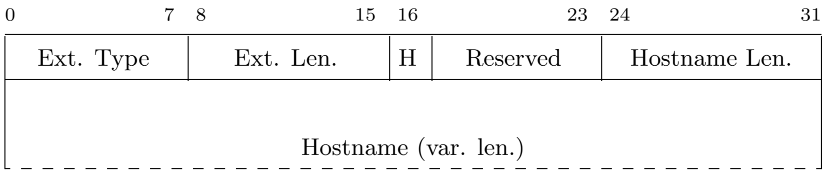

Figure 7.3: Structure of NREQ message extension with hashing option

For indication, whether the full-text name or the hashed one is included, we use one bit of the *Reserved* field. The new *H* flag—standing for *Hashed Name*—is set to *1* if the name is hashed, otherwise it is set to *0*. This keeps it compatible with the standard system described in Section 5.5. The *Hostname Length* field is per default set to *16* if the *H* flag is set to *1*, as this is the default length for an MD5 hash.

The structure of the new NREP message is shown in Figure 7.4.

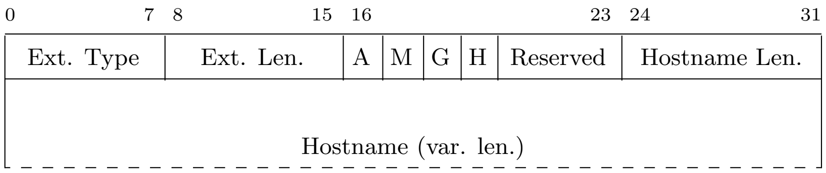

Figure 7.4: Structure of NREP message extension with hashing option

Here, we also used one bit of the *Reserved* field as the new *H* flag, indicating whether the included hostname is hashed or not.

The structure of the new NADV message is shown in Figure 7.5.

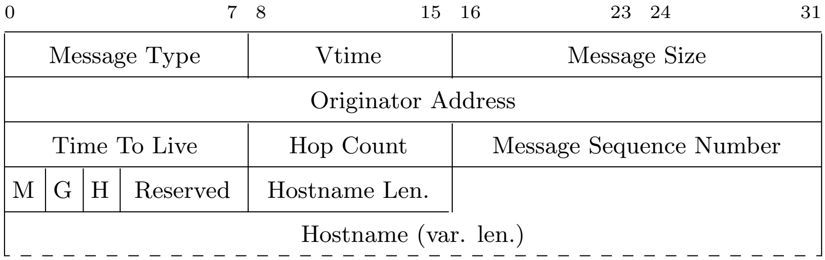

Figure 7.5: Structure of NADV message with hashing option

The traffic saving effect of hashed hostnames in the NERR packet is very low, as such messages occur rarely because of the relatively static hostname to IP address mapping. However, if a node has only knowledge about a mapping of a hashed name to a local address, it has no other change but informing the other nodes using the hashed key. Figure 7.6 shows the new NERR design according to the other packets.

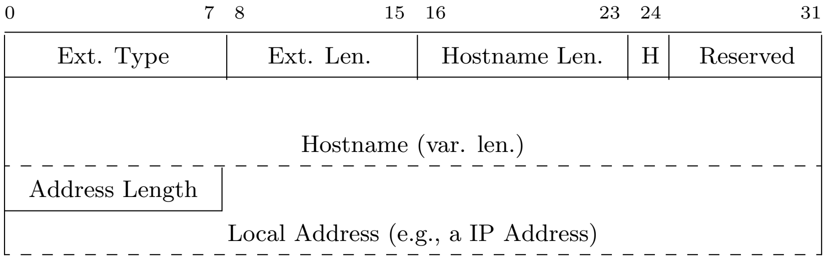

Figure 7.6: Structure of NERR message extension with hashing option

The newly introduced *H* flag indicated, whether the hostname is hashed. In this case, the *Hostname Length* is automatically set to *16*.

Hash functions map variable data to a fixed data size. In our case, we only assume the hashing of names greater the size of the resulting hash. However, in this case the hashing of two complete different hostnames could lead to the same hash key. This eventually leads to a hash collision and therefore a false mapping. If a node recognizes such a wrong hash to local address relation, it deletes the entry from its HAM table and distributes a Collision Error (COLERR) message inside the next OLSR packet. Nodes receiving the COLERR packet delete the false entry in their HAM table, too, if applicable. If the alarm reaches the hosting node eventually it stops distributing the hashed version of the conflicted hostname and instead starts sending the full name.

The structure of the COLERR message is presented in Figure 7.7. It only consists of a number of MD5 hash keys. The OLSR message type for this message is set to *131* (cf. Message Type 128 for NADV messages, 129 for NERR, and 130 for BANNO).

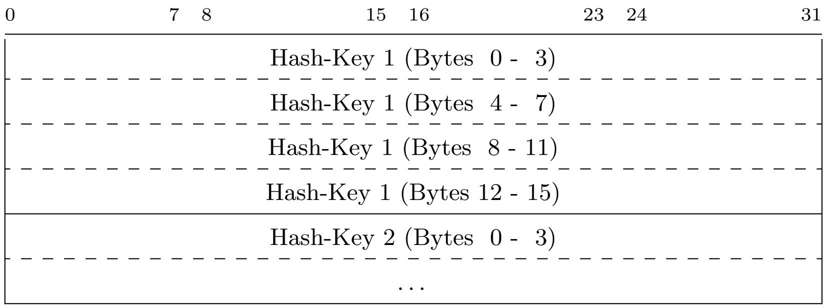

Figure 7.7: Structure of COLERR message [62]

In the next section, we show the simulation results evaluating the extension.

7.1.3 Evaluation

To evaluate the behavior of the described extension, we simulate the new system with the several settings. We used the same scenario and parameters as described in Section 6.1. The delay and success rate lie in the range of the results presented in Section 6.2, which is obvious as we only changed the transmitted hostname. Therefore, we only observe the behavior of the traffic in this subsection.

Figure 7.8 shows the generated traffic in the reactive mode with and without hashed host names. Every node hosts a hostname of 255 Byte, which is the worst case scenario for the generated traffic.

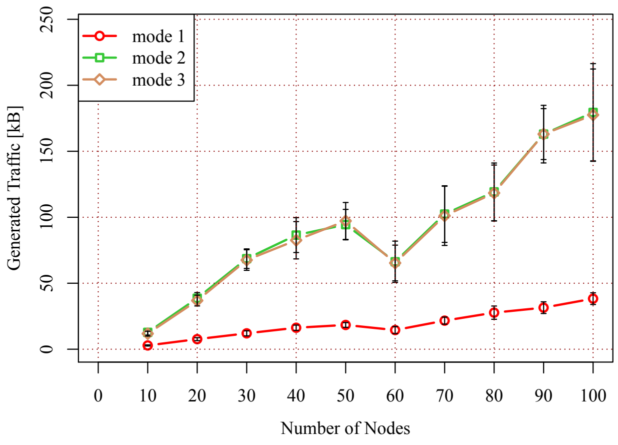

Figure 7.8: Comparison of traffic in AODV w/ and w/o host name hashing [82]

In *Mode 1*, the name is included hashed in the NREQ and the NREP. In *Mode 2*, the NREQ is sent with plaintext names included, while the reply contains the hashed version. Finally, *Mode 3* represents the case, where both messages contain the full name.

The generated node movement is always the same for all three modes to have a meaningful comparison. It results in different traffic for the same amount of sent request and reply messages. In every run, 10 nodes were chosen randomly to request a name mapping from a randomly chosen other node. We have done at least 20 runs per number of nodes and plotted the averaged over the results, including the standard deviation.

As can be seen in the figure, Mode 1 where the hashed name was included in both messages generates the least traffic. The curves for the traffic generated in Mode 2 and 3 are almost equal, despite Mode 2 uses hashed names in the reply. The reason for this behavior is that the NREQ is sent broadcast while the NREP is sent back unicast. It can be seen that having a bigger reply does not add considerably higher load to the network. On the other hand, reducing the length of the request can increase the overall performance of the system significantly.

Even tough our implementation gives the application the possibility to include hashed names in the NREQ and the NREP, our recommendation for action is to use a hashed name in the NREQ, if possible, and to send out the full name in the reply, if applicable.

As the proactive mode consists of distributing data in the entire network in advance, there are only two modes considering hashed hostnames. Either the full name is distributed in the NADV messages or the hashed one. To evaluate the performance of both modes, we simulated a scenario where all nodes host unique hostnames with a length of 4, 8, 16, 32, 64, 128, and 256 Byte. The 4 Byte scenario could be interpreted as having a logical overlay network using IPv4 addresses, while the 16 Byte case could be an overlay with IPv6 addresses. The 256 Byte scenario is the worst case scenario where every node has a hostname with a maximum length.

Again we used the same parameters as shown in Section 6.1. We changed the number of nodes in the range of 60, 70, 80, 90, and 100 nodes. Every set was ran 20 times and the shown graph is the average over the results. The deviation is also computed and indicated. The time period for sending NADV messages was set to 2 s, which is the double period of the OLSR Hello messages. Figure 7.9 shows the generated name resolution traffic for each scenario including the standard deviation of the values.

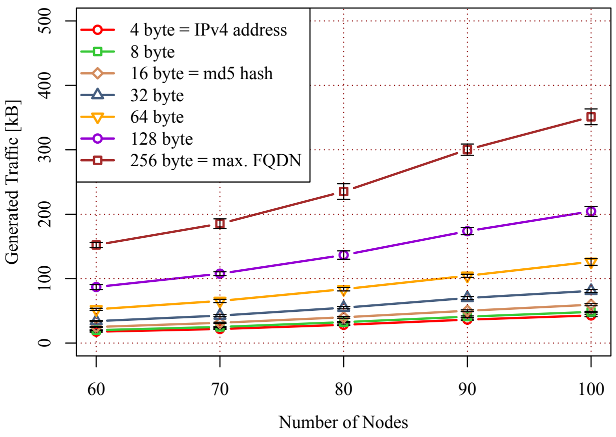

Figure 7.9: Comparison of traffic in OLSR w/ and w/o host name hashing [82]

The diagram shows that the generated traffic of 4, 8, and 16 Bytes differs not as much as in comparison to the longer names. For hostnames greater than 16 Bytes, the traffic increases dramatically. To evaluate the impact of hashing hostnames, it should be noted that by hashing names the traffic can be limited to the 16 Byte case. Especially if this case is compared to the worst case of having 256 Byte names, it can be seen that around 80 % of traffic can be saved by consequently selecting the hashed name mode.

7.2 Service Discovery

This sections deals with a service discovery mechanism based on the introduced name resolution mechanism. The basic idea and the state of the art is shown as well as details on the implementation and simulation of the approach. We previously published some content of the following section in a conference paper [86]. Furthermore, the student Z. Ansar worked on a literature survey during his Advanced Research Project [87] and A. Saliminia implemented and tested the following approach in his master thesis [88]. All students were supervised by the author of this thesis.

7.2.1 Preliminary Considerations

Services are essential for communication networks. Most communication is done between a client and a service provider, while the client utilizes the service of the provider. Those services could be usual technical services like the name resolution itself or more abstract ones like medical aid or fire distinguishing. As we work on a communication system for disaster scenarios, we want to design a service discovery mechanism that is applicable for both.

Firstly, we want to define the term service. In a layered communication system, a lower layer always provides a service for the next higher layer. The Network Layer for instance provides the service of packet delivery for the Transport Layer, in case of the Internet Protocol unreliably. In this section, we just consider services in the Application Layer of the Internet reference model.

Nodes joining a new network might be interested in special services or might be interested in all services provided by any node in the network. We want to design our framework to provide both. A further design criteria is decentralization and self-organization, just like in the already proposed name resolution scheme.

Service discovery in infrastructure networks is usually provided by centralized entities. Service providers register their services at one or more central points and clients can discover them the same way. Examples are the Service Location Protocol (SLP) [89]—proposed by the IETF—and industry-made protocols such as Universal Plug and Play (UPnP) [90] or Jini [91]. As this thesis is focused on MANET-adapted approaches, we skip any explanation of those protocols as they are generally not applicable for our scenario.

The service discovery problem is generally similar to the name mapping problem. Communication between the client and the server is done at the Network Layer and a node needs to know the IP address of the server beforehand. Furthermore, the route to the found IP address is needed, too. This leads to the same requirements and problems we already dealt with in the past chapters. A service is identified by a service ID or any URL. That means that we can use the already proposed name resolution framework to realize service discovery, too.

Selecting a service comes with some differences compared to just mapping node identities to local addresses. Several nodes could provide the same service, which leads to the resolution of one service ID to several IP addresses. On the other hand, we need some further information in the service discovery process like service descriptions.

In the following section, we show some approaches for service discovery mechanisms for MANETs. Afterwards, we present our own mechanism based on the already introduced name resolution over adaptive routing framework.

7.2.2 State of the Art

This section shows selected related work regarding service discovery approaches in ad hoc networks. Ververidis et al. [17] gave a good survey on general techniques. In this section, we present some selected approaches.

Koodli et al. [92] proposed a service discovery protocol based on an on-demand routing. The basic service discovery process uses the same operations and message types as the route discovery process, but extends them with a Service Request (SREQ) or a Service Reply (SREP) message. As SREQs, two kinds of extensions can be used. The Service Port Request asks for a services bound to a defined port while the Service URL Request asks for a Service URL.

Serhani et al. [93] showed a decentralized service discovery mechanism for MANETs, which is used for the management of provided physical services having specific attributes. Physical services are rather unique because service providers have to move physically to the location of need. It is furthermore assumed that service providers can only provide the service for one requesting node over a period of time. The service can be discovered by sending a SREQ message broadcast and specifying the distance with the number of hops via the time to live field in the IP packet.

Heni et al. [94] presented a service discovery protocol based on OLSR. The new Service Discovery Message (SDM) is defined for the routing protocol. SDM is added to Topology Control messages (TC-SD) and broadcasted periodically. This announcement is therefore received by all nodes in the network. A node receiving such a message checks its service table for the corresponding key to the sources network address. If the list associated with this key already contains the service, the node compares the lifetime of the two service descriptions and saves the one with the greatest lifetime. If the node does not find the address of the source of the message, it simply adds another column in the table corresponding to the new provider.

All presented papers were based on routing protocols. Further approaches not presented here try to use a hybrid approach with reactive and proactive phases. However, they are all limited to only one underlying routing mechanism and therefore not adaptive to different network scenarios. In the framework presented in this thesis, we include several routing-based proposals combined with adaptive routing.

Furthermore, the reactive schemes only deal with the discovery process but not with an efficient decentralized information storage used by the discovery process. None of the papers had location information involved in the service discovery. In the following three subsections, we show the extensions we made to enable service discovery using our name resolution framework. In Section 7.3, we will furthermore show how to extend our scheme to enable location-aware service discovery, too.

7.2.3 Changes in the Backbone

Regarding changes in the backbone, two aspects are important. First, the storage of the gathered mapping information has to be adjusted to the new requirements and, second, the identification of the services in the naming scheme has to be defined.

A first idea to integrate service discovery into the name resolution mechanism would be to extend the HAM table with the necessary further fields. However, we decided to introduce a new table that is designed to just deal with the service discovery task named the *Service Table*. Figure 7.10 shows an example entry in this new table.

The table consists of the *service name*, i.e., the service ID, including the regarding *length*. Furthermore, we included a string field to store a *service description*. Descriptions could be important to the requesting nodes, e.g., to transport attributes like "available". If no description is available, the *description length* field is set to *0*. The following field is the *local address*, according to the HAM table shown in Table 5.7. Furthermore, we included a flag indicating that this service might be *multihomed*, i.e., available at more than one node, and a *time_to_live* field if this service is available just for a limited time.

In case of multihomed services, every mapping is stored in one entry. In the state of the art, the identification of services is often done by using port numbers or URLs. In this thesis, we assume that services are identified by abstract *service IDs*. Furthermore, we assume some

data type	**field name**
uint8_t	service_name_length
string	service_name
uint8_t	service_description_length
string	service_description
bool	address_type
uint8_t	address_length
IPAddress	local_address
bool	flag_multihomed
timestamp	time_to_live

Figure 7.10: Structure of a Service Table entry

important and often used services to have predefined identities in our system, e.g., *ID=firefighter* or *ID=medicalaid*. The several (special) rescue teams are assumed to be informed in advance of such predefined important IDs and set their equipment according to the guidelines.

7.2.4 Extensions of the Reactive Mode

The reactive scheme of our name resolution approach presented in Section 5.5 is based on the AODV protocol. We use name requests and name replies transported via AODV RREQs and RREPs to implement a request-response-algorithm.

The same approach is used in the service discovery case. As the processing of a Service Request (SREQ) and a Service Reply (SREP) is different from the lone name resolution ones, we decided to introduce new messages for this task. Figure 7.11 shows the structure of the new SREQ message.

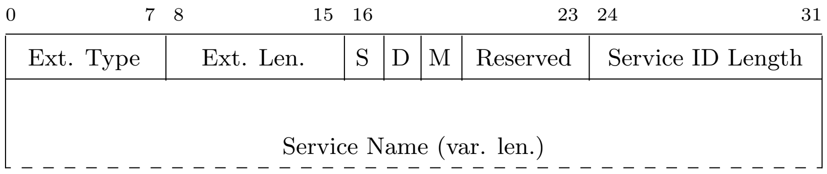

Figure 7.11: Structure of Service Request (SREQ) message

As the SREQ is a new message independent from the implementation of the name resolution messages, we have to introduce a new *Extension Type*, which is set to *131* for the SREQ (cf. Extension Type 128 for NREQ, 129 for NREP, and 130 for NERR). In comparison to the NREQ format presented in Figure 5.9, we replaced three bits from the reserved field with new flags.

If the *S* flag is set to *1*, a node requests all services available in the subnet. This method is implemented for all nodes entering a network for the first time to get an overview about the services currently provided by any node. However, assuming large networks with hundreds of nodes this flag could lead to a huge overhead especially when used by many nodes frequently. This eventually leads to a knowledge of all nodes proving one service and to a possible call for all nodes, e.g., for a rescue mission, or to the selective call for the probably best provider, e.g., depending on the service attributes.

The *D* flag indicates, whether the requesting node wants to have a service description in the reply. If set to *0*, the replying node leaves out the description field in the reply, if set to *1* a description is included.

With the *M* flag, a requesting node indicates whether it wants to have only one or all service provider in the network. If the flag is set to *0*, the usual algorithm is used. That means that the first node replying to this request stops the request and answers with a reply. If the *M* flag is set to *1*, the request is still broadcasted by nodes being able to reply, in addition to the generated SREP.

In Figure 7.12, we show the design of the newly introduced SREP. The SREP *Extension Type* is set to *132* (cf. Extension Type 128 for NREQ, 129 for NREP, 130 for NERR, and 131 for SREQ). The body structure is similar to the NREP presented in Figure 5.10. We kept the TLV style and the ***A****ddress Type*, ***M****ultihomed*, and ***G****ateway* flags but added the *D* flag.

If this flag is set to *1*, a service description is included in the message. In that case, the *optional* fields *Service Description Length* and *Service Description* are appended. The style guidelines of the *description* field is not in the scope of this thesis.

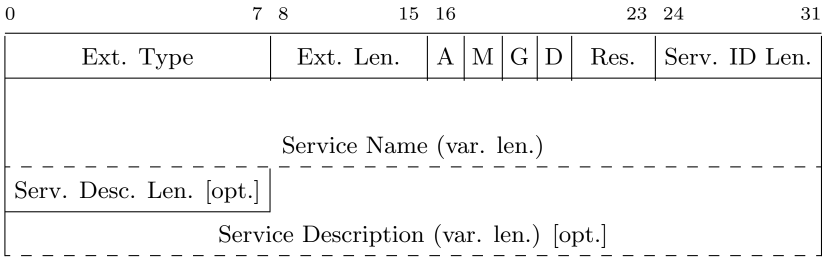

Figure 7.12: Structure of SREP message

Figure 7.13 shows the adapted flow chart for service requests coming from a node's Kernel. The procedure is almost the same as for the blank name resolution process, as shown in Figure 5.11.

If higher layers want to have all services discovered, the node immediately sends out a RREQ/SREQ with the *S* flag set to *1*. If only a particular service is needed, the node first looks up its own service table for regarding entries. If information about the requested service ID is not available, the combined request is broadcasted. If an IP address for the service is available, the route to the found IP address is checked. If available, the connection establishment can take place, if unavailable a lone RREQ regarding the AODV standard is broadcasted.

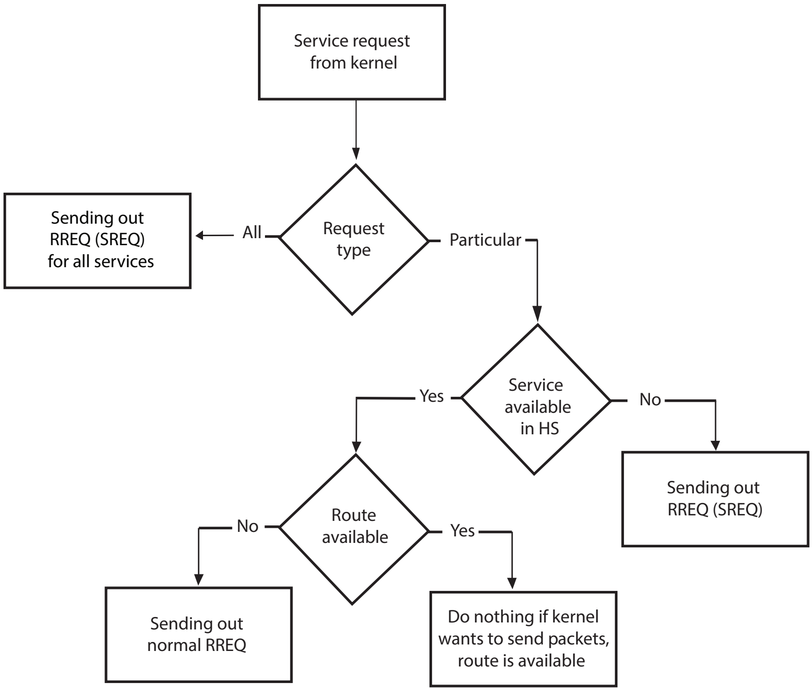
Service request from kernel
Request type
All
Sending out RREQ (SREQ) for all services
Particular
Service available in HS
Yes
No
Sending out RREQ (SREQ)
Route available
No
Yes
Sending out normal RREQ
Do nothing if kernel wants to send packets, route is available

Figure 7.13: Flowchart for processing service requests coming from the Kernel in reactive mode

The procedure to handle service requests is shown in Figure 7.14, which is an enhancement of the name resolution flow chart shown in Figure 5.11. If a node receives a SREQ with the *S* flag set to *1*, it replies with all hosted services and forwards the message. If only a particular service is requested, it replies if it has a fitting entry in the service table or if the node itself hosts the service. Otherwise, the request is forwarded. Furthermore, if the *M* flag is set, the request is replied, if applicable, and forwarded at the same time. The route finding process starts after the RREQ/SREP is generates as described in Section 5.5.

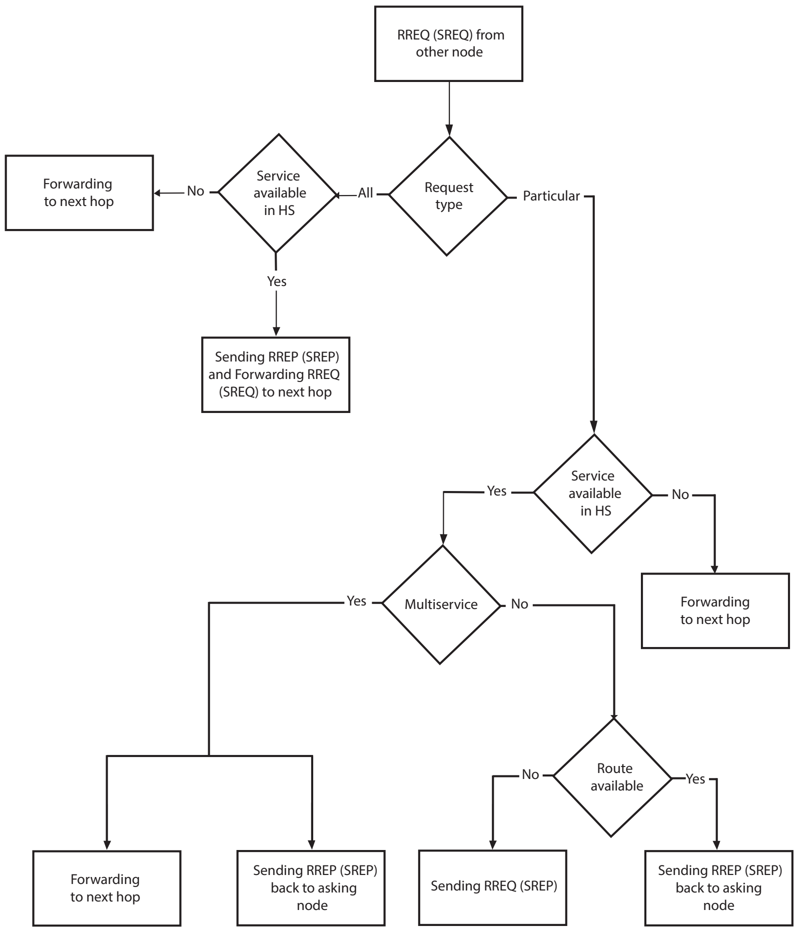

Figure 7.14: Flowchart for processing service requests coming from other nodes in reactive mode

7.2.5 Extensions of the Proactive Mode

The design of the proactive service discovery scheme is straightforward. Nodes that want to share the provision of a service, distribute a newly introduced OLSR message. The Service Advertisement (SADV) message is distributed in the same way as the NADV messages introduced in Section 5.6. Every node informs one MPR about the hosted services via periodical unicast messages. The MPR then distributes the collected SADVs along with the NADVs and the routing information periodically. The period of the SADV distribution can be set by the user independently or just to the value of the NADV period. The structure of the SADV is slightly different and shown in Figure 7.15.

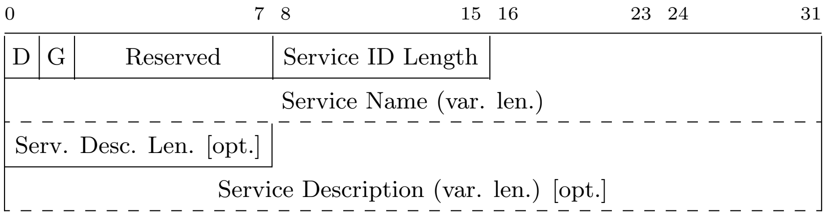

Figure 7.15: Structure of the SADV message

If the *D* flag is set to *1*, a description is appended to the message. The OLSR message type for this message is set to *132* (cf. Message Type 128 for NADV messages, 129 for NERR, 130 for BANNO, and 131 for COLERR).

To dismiss a mapping, we could introduce a Service Error (SERR) according to the NERR introduced in Subsection 5.5.6. However, for the necessary task only the ID is needed to delete the entry. Therefore, the NERR packet gives all the functionality needed, too. We therefore decided to use the NERR packet but changed the processing in order to not only search the HAM table but also the Service Table for expired entries.

7.2.6 Simulation Results

To evaluate our service discovery scheme, we extended our name resolution scheme and implemented the new messages and the processing in new Click elements. The performance of the three key parameters—traffic, delay, and success rate—is comparable to the name resolution scheme already prospected in Chapter 6. However, we determined other issues directly related to the service discovery task.

Figure 7.16 shows the service availability against an increasing number of participating nodes. We used a fix area of 1000 m x 1000 m and placed the nodes randomly. One can see that the curve quickly increases to a high level of around 85 % and stays at that level. This is because of the increasing number of connections between the nodes due to the more dense scenario. We do not reach 100 % eventually as more nodes result in more congestion and interference.

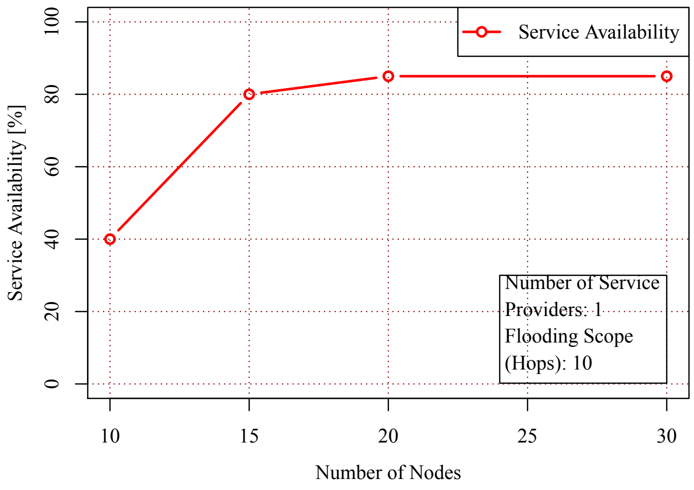

Figure 7.16: Service availability in the reactive mode [86]

In the second plot shown in Figure 7.17, we measured the delay in the reactive mode until the service is found. We set the number of service providers to two, which means that the node has to find only one of the providing nodes in order to resolve a service request.

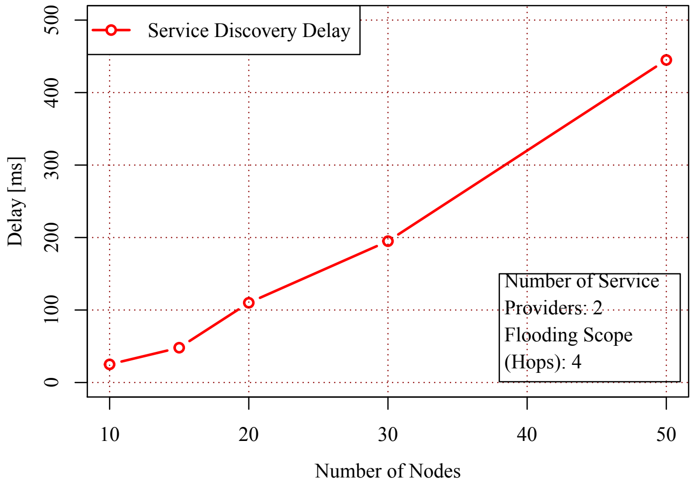

Figure 7.17: Delay until a service is discovered in the reactive mode [86]

The plot shows an (almost) linear growth with increasing number of nodes. The more nodes participating in the network the longer we have to wait until we find a service provider. By setting more nodes to serve as a service provider the gradient of the curve can further be decrease. However, the linear behavior stays considering only one reactive network.

In Figure 7.18, we figured out the relationship between the generated traffic against the number of requesting nodes. We fixed the total number of nodes to 10 and increased the number of requesting nodes from 1 to 5.

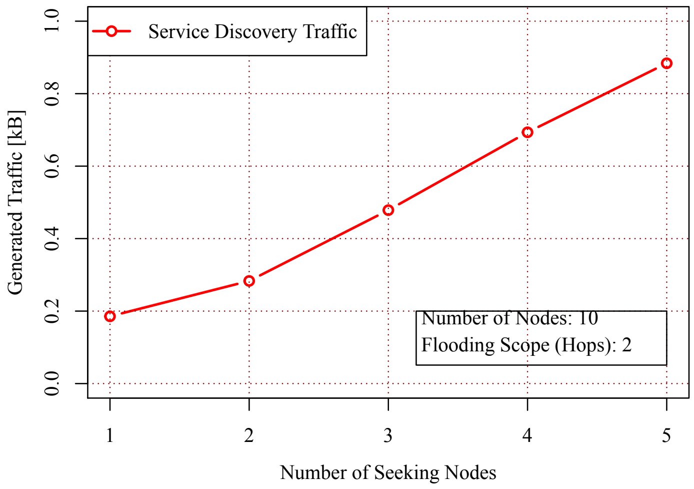

Figure 7.18: Generated traffic against number of requesting nodes [86]

We observed an almost linear relationship, which means that the generated traffic scales with increasing number of requesting nodes.

7.3 Location Resolution

This section deals with the extension of the previously presented framework to enable a location-aware name resolution. We previously published some content of the following section in a conference paper [86]. Furthermore, the student F. Zhang contributed a literature survey during his seminar paper (Hauptseminar) [95] and the students R. Satannavar and V. S. Raghavendra worked on this topic during their Research Project [96]. All students were supervised by the author of this thesis.

7.3.1 Basic Idea and Use Cases

In this thesis so far, we considered name resolution only as a mechanism to resolve addresses in the Application Layer to addresses in the Network Layer by using routing mechanisms. In Section 7.2, we showed how this basic mechanism can be used to find or locate services, too. However, a possible use case could demand further information and functionality from the basic mechanism.

The current system is able to find IP addresses for given host names including the route. This could either be the nearest one, or the fastest, or simply all possible founds (cf. Subsection 7.2.3). Including location information in the discovery process could be worth it, considering abstract, non-technical services provided by humans.

In disaster scenarios, nodes could search for abstract services like *medial aid* or the *fire brigade*. A victim could be interested in an ambulance car, e.g., because of an injury, but might only need a single one. The nearest service provider to the nodes' position would be the best choice, presumed the service is available. Or a victim could be interested in more than one service provider but not more than a defined number, e.g., considering a fire were only two fire engines are needed.

The current system does not provide such functionality for an optimal selection. A node with a better communication link could be selected as the "best" node, while its geographical position is further away than the one of a node with a worse link.

Therefore, we introduce location-aware name resolution to our existing system. We assume that every node is either equipped with a Global Positioning System (GPS) device or is located by other nodes. Some effort has be spent in the Graduate School [1] on localization of nodes, especially using UAVs [97].

7.3.2 State of the Art

Several protocols were proposed, to enable a spatial addressing or spatial delivery, respectively. The key goal is to send packets just to a defined zone, either to all nodes in this zone or just a subset of nodes. The approaches can be grouped to flooding-based and routing-based.

Ko et al. presented the Location Based Multicast (LBM) approach, where messages are flooded - but only in defined spatial zones [98]. Those so called Forwarding Zones are usually rectangular and all zones combined cover the whole area. Nodes within the zone broadcast messages, while nodes outside the zone withdraw those packets. The other presented option is based on distances. The sender defines a distance to which the packet should travel. If a receiving node lies within the distance, it forwards the packet, otherwise it withdraws.

Stojmenovic et al. proposed an enhancement of this scheme by replacing the Forwarding Zones with Voronoi Diagrams [99]. If a sender has N neighbors, it divides the network into N pieces—every piece building a Forwarding Zone. Every Zone is then consisting of a direct neighbor and a number of nodes lying closest to this neighbor. The packets are then forwarded to the neighbor whose Voronoi area fits to the target region. This algorithm is repeated until the packet is successfully delivered.

Summarizing, we can say that all flooding-based approaches lead to a high traffic. In comparison, the routing-based algorithms establish a connection from the sender to the target area in advance before sending packets. Examples are the Geocast Adaptive Mesh Environment for Routing (GAMER) approach [100], Geocast based Temporally Ordered Routing Algorithm (GeoTORA) protocol [101], and GeoGRID [102].

In GAMER, the source node broadcasts so called Join-Demands into the network. Nodes lying in the target area reply with a unicasted Join-Table packet. The so established backwards route is then used to send the geocast data to the target area, where it is flooded to all corresponding nodes.

In GeoTORA, the nodes lying on the way from the source to the destination area assign so called *heights* to destination-oriented Directed Acyclic Graphs (DAGs) during the route discovery process. If the height is zero, the destination region is reached. After the route discovery, the source nodes send the data unicast from a high height to height zero, where it is flooded to all nodes having that height.

The basic idea behind the routing-based approaches is to use the routing mechanisms, which was reactive routing in the presented protocols, to establish location-aware routes. In the following sections, we present our extension to our name resolution framework to resolve names based on location information.

7.3.3 Changes in the Backbone

For reason of simplicity, we can use the backbone of our name resolution or service discovery schemes almost untouched. In our framework, we assume nodes to find name mappings and corresponding routes based on location requirements. After this initial process, the connection is established. Therefore, we need to store the collected location information in our backbone to be available for connection establishment decisions.

The first decision to make is the way of storage of the node's position. As we assume that every node is equipped with a GPS device, we chose the prominent position representation in *Longitude* and *Latitude*. With both values, a node can specify the position of other nodes or compare it to the own values. Both values are stored in a 32 bit field.

Consequently, we extended the HAM and the service table, respectively, with the corresponding two fields. If no value is available, because the destination node is unable to calculate its own position or the location-awareness is simply not used, both values are set to *0*.

In the following three subsections, we describe how to get the location information based on the routing schemes.

7.3.4 Reactive Mode with Location Information in the Service Request

The name resolution in the reactive mode is related to the reactive behavior of the route discovery. This contains a request to ask for a name resolution and a reply for responding (cf. Section 5.5). Therefore, two possible ways of including location information into the process are thinkable: either by changing the structure and processing of the request or of the reply.

As the implementation for the name resolution and service discovery process is similar and somehow interchangeable, we just talk about the location-aware service discovery process in the following two sections. The showed extensions and mechanisms for the SREQ and SREP are also applicable for the NREQ and NREP, respectively. This section deals with the first one, the integration of location information in the request.

Using this approach, the questioner defines further location requirements on demand in the requests. The current NREQ (cf. Figure 5.9) just asks for the name and nothing more. In the current SREQ (cf. Figure 7.11), we add requirements by introducing further flags. With it, a node can for instance request having a service descriptions included in the upcoming SREP.

For the location-aware approach, we give the questioner the possibility to add the own position and a range. If the requested service lies within the range around the position, it responses to the request. Otherwise, the NREQ is withdrawn and not answered.

If a victim requests all fire brigades in the circumcircle of 5 km, it includes its position and sets the range to 5000 m. Figure 7.19 shows the new structure of the SREQ with included location information.

The GPS position is given by the *Longitude* and *Latitude* couple in a 32 bit field each. There are other possibilities for giving a position but we chose this one. The 8-bit long *Range* field gives the circumcircle

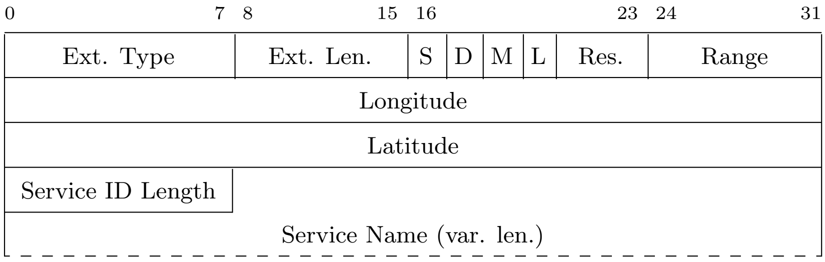

Figure 7.19: Structure of SREQ message with included location information

around the position in hundreds of meters in which service providers are allowed to answer the request. If set to *50*, the provider must be within a range of 5 km. The maximum range is therefore 25.6 km, which should fit to a common disaster area. If a bigger range is needed, the field has to be extended, e.g., to 16 bit. The three new fields are added before the name itself without an extra length field as they have a fixed size.

The regarding Click elements only process the new fields if the *L* flag, standing for Location Information, is set. We used one bit of the *Reserved* field for this tasks. In total, 72 bit are added to the SREQ and therefore to the whole AODV packet. The RREQ/SREQ couple is broadcasted into the network the same way as already described.

For the reply side, nothing changes in this mode. The only difference is that only nodes within the range might answer the request. Figure 7.20 shows a possible scenario with one client (Node C) and two possible service provider (Nodes S), where only one service provider lies within the range and therefore is allowed to answer.

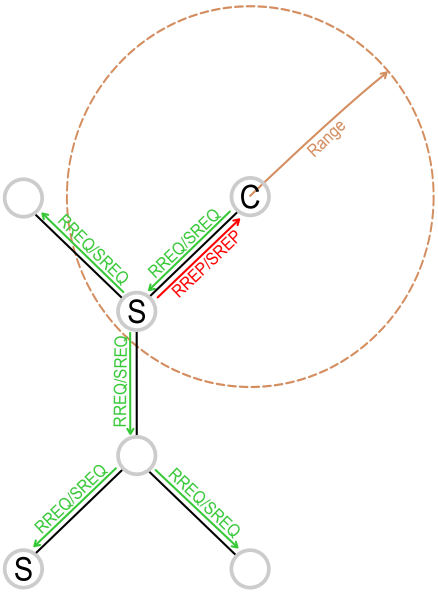

Figure 7.20: Possible scenario for location-awareness in the request

7.3.5 Reactive Mode with Location Information in the Service Reply

The second realization of location-aware name resolution and service discovery is based on sending location information along with the reply. Generally, requests consume more traffic than replies because AODV broadcasts the RREQ while the RREP is unicasted. This effects was shown in the evaluation of the hostname hashing usage in the reactive mode in Subsection 7.1.3. A resource saving alternative would be to include the location information in the reply. This approach is comparable with the service discovery approach, where service providers include service description only in the reply and the requester is just demanding them in the request.

We extended the SREP message (cf. Figure 7.12) with similar fields as in the other mode. The new structure can be seen in Figure 7.21.

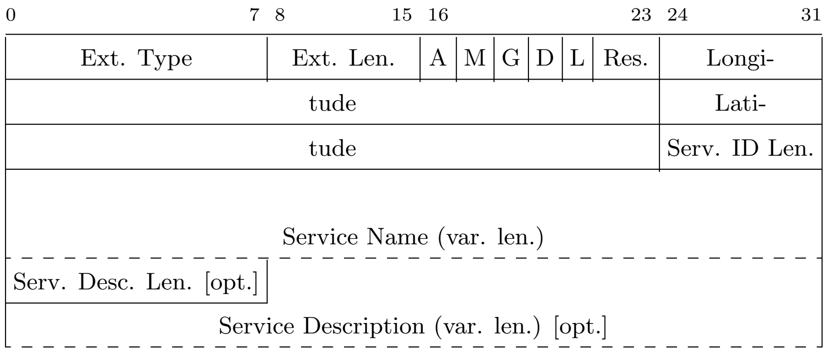

Figure 7.21: Structure of SREP message with included location information

Again, the position is given by *Longitude* and *Latitude*, now included in the SREP. In the second mode, the requester sends a conventional SREQ in the normal way as presented in the previous section and chapters. We keep the new *L* flag from the first approach included in the SREQ, but this time no location information is added in there. The *L* flag now indicates, whether the reply should include the longitude and latitude of the responding node or not.

A requesting node could broadcast a request for a certain service and set the two flags for requesting all service provider in the network and for requesting location information at the same time. This would lead to a certain number of replies. If none or only one reply is received there is no choice for the requesting node. If more than one reply is received the requesting node can chose whether it wants to establish a connection to only the nearest node or to a set of node with a favorable position range.

The second advantage beside the lesser traffic is that the decision is made after the discovery process. If a node uses the first approach by setting a range and receives no answer, it has to increase the range and ask for the service again. This causes not only even more traffic load for the network but also a much higher delay. By requesting all services with location information, the node just initiates the discovery process once and has a complete picture of available services including the location of the service providers in less time.

It should be noted that the L flag can be set by each node depending on the scenario. If a node searches for a non-physical service, e.g., for an Internet gateway, location information might not be useful and the flag can be set to *0*.

7.3.6 Proactive Mode

In the reactive mode, the requesting node can influence the service discovery process as it is based on the request-response-principle. The last two sections showed, how the basic process is extended by adding new fields to the packets. In the proactive mode, the name resolution information is distributed in advance. A node, which wants to connect to a service provider, knows the provided service and the route to the service provider due to the periodically sent SADV and routing messages.

The introduction of location information is therefore primarily in the scope of the SADV distributing service provider. Including the location of an Internet access for instance might not be useful as nodes just forward packet to the gateway no matter how far the node is spatially away. For a physical service like medial aid, it might be interesting for the provider to send along the location. We assume that the mission planning of the rescue teams and the other nodes decide to use the location-awareness independently. With the following implementation, we just provide the functionality.

To provide the functionality, we extended the SADV structure shown in Figure 7.15 to the new structure presented in Figure 7.22.

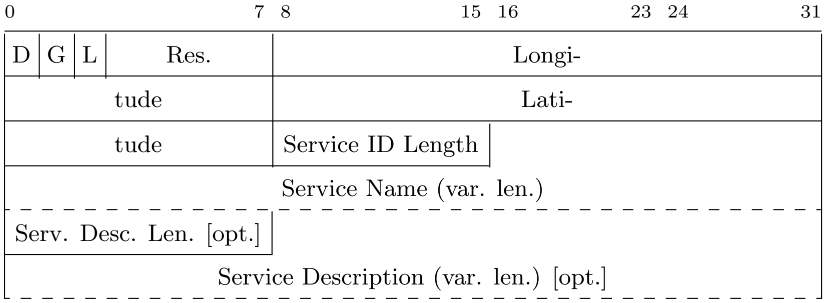

Figure 7.22: Structure of the SADV message with location-awareness included

Similar to the reactive mode, the location of the service provider is given with the *Longitude* and *Latitude* couple. If the two additional fields are included, the *L* flag, previously a reserved flag, is set to *1*. If set, the receiving node processes the additional fields. By including location information in all emitted SADV and NADV messages, the periodical load for the network is increased, depending on the number of distributed messages.

8 Conclusion and Future Work

In the last chapter, we want to conclude the main aspects of the thesis. Furthermore, we indicate possible future work on not contemplated issues.

8.1 Wrap Up and Conclusion

In this thesis, we presented our framework to solve two major things: providing an addressing scheme and an address resolution mechanism applicable for disaster scenarios.

We identified the main challenges in such scenarios and the need for ad hoc networking technology if the communication infrastructure fails. Our proposed framework consists of an addressing scheme that splits the task of identification and localization of the common local IP addresses by introducing a new identifying address layer. While this scheme solves the problem of changing local addresses, we showed that this approach causes the new problem of a proper name resolution mechanism that works in MANETs. We presented related work to name resolution in ad hoc networks and indicated a new solution based on different routing protocols, included into an adaptive routing framework.

Our routing-based name resolution framework consists of two different routing protocols, the reactive AODV and the proactive OLSR. We showed, how to extend those two protocols to include mechanisms to resolve names on-demand or in advance, while keeping the protocols compatible to the standard. Our framework is completely decentralized and works self-organized in the sense of automatic. We presented, how to switch between the schemes without loosing knowledge. Furthermore, we show how our framework is able to provide inter-domain routing and to find names in coupled networks.

We showed three different extensions to the basic mechanisms providing hashed host names to reduce the traffic, a service discovery mechanism on top of the name resolution schemes, and location-awareness for better usability in disaster scenarios.

Our simulation results validated the correctness of the concept and showed that we can outperform the approaches of the state of the art.

8.2 Future Work

Though we solved important problems with the concept presented in this thesis, further work can be done to push the performance and functionality further.

A further investigation should be done on consistency issues of names. The current system provides a mechanism to tell other nodes about name conflicts if more than one node shares the same identity. However, a deeper look into consistent naming schemes could prevent such conflicts in advance. We discussed the relationship between names and addresses for different use cases but the process of name assignment was not in the scope of this work. Identities or service IDs could be assigned to the nodes in advance, e.g., before a rescue mission starts, or by each node individually.

During the thesis, we showed some extensions to the basic mechanism to enable service discovery and location-awareness. But there are further use cases thinkable, where we can use our framework with limited changes to solve further problems. E.g., a working name resolution is essential to build a realistic underlay for Delay Tolerant Networking (DTN) [33]. Due to high mobility, DTNs are used to enable a successful message transmission if connections are sparse and unsteady. Furthermore, our naming scheme can be used to map multicast group messages to static identities in reliable multicast communication [31, 32]. Both aspects are subject of ongoing research in the Graduate School [1] that founded the work on this thesis.

A very important issue that was out of focus in this thesis is security. We assumed identities to be unique and assigned to the right node. However, if malicious nodes take wrong identities, they could easily disturb mission requests or send nodes to wrong destinations. In the Host Identity Protocol (HIP), a mechanism was introduced to encrypt the node identity with private keys so that every node reading another node's hostname can validate the correctness of the sender. However, as we have not included such mechanisms in our framework, this issue is up to the future work.

Appendix

List of Figures

Acronyms

ANS	Ad-hoc Name Service.
AODV	Ad-hoc On-demand Distance Vector.
AOMDV	Ad hoc On-Demand Multipath Distance Vector.
ARP	Address Resolution Protocol.
BANNO	Border Node Annotation.
BN	Border Node.
Click	Click Modular Router.
CML	Chameleon Routing Protocol.
COLERR	Collision Error.
CSMA	Carrier Sense Multiple Access.
DAG	Directed Acyclic Graph.
DHCP	Dynamic Host Configuration Protocol.
DNS	Domain Name System.
DNS-SD	DNS Service Discovery.
DSDV	Dynamic Destination-Sequenced Distance-Vector.
DTN	Delay Tolerant Networking.
EID	End Point Identifier.
FOG	Forwarding on Gates.
FQDN	Fully Qualified Domain Name.
GAMER	Geocast Adaptive Mesh Environment for Routing.

GeoTORA	Geocast based Temporally Ordered Routing Algorithm.
GPS	Global Positioning System.
HAM	Hostname to Address Mapping.
HIP	Host Identity Protocol.
HIT	Host Identity Tag.
HTTP	Hypertext Transfer Protocol.
IANA	Internet Assigned Numbers Authority.
ID	Identification.
IEEE	Institute of Electrical and Electronics Engineers.
IETF	Internet Engineering Task Force.
IP	Internet Protocol.
IPv4	Internet Protocol version 4.
IPv6	Internet Protocol version 6.
ISO	International Organization for Standardization.
LBM	Location Based Multicast.
LISP	Locator/Identifier Separation Protocol.
LTE	Long Term Evolution.
LUNAR	Lightweight Underlay Network Ad hoc Routing.
MAC	Media Access Control.
MANET	Mobile Ad hoc Network.
MAODV	Multicast Ad hoc On-Demand Distance Vector.
MD5	Message-Digest Algorithm 5.
mDNS	Multicast Domain Name System.

MOBICOM	International Graduate School on Mobile Communications.
MPR	Multipoint Relay.
NADV	Name Advertisement.
NDP	Neighbor Discovery Protocol.
NERR	Name Error.
NREP	Name Reply.
NREQ	Name Request.
ns-3	Network Simulator 3.
OLSR	Optimized Link State Routing.
ONE	Opportunistic Network Environment.
OSI	Open Systems Interconnection.
P2P	Peer-to-Peer.
PAT	Pseudo Address Table.
PDA	Personal Digital Assistant.
RERR	Route Error.
RFC	Request For Comments.
RREP	Route Reply.
RREQ	Route Request.
RWP	Random Waypoint.
SADV	Service Advertisement.
SAP	Service Access Point.
SDM	Service Discovery Message.
SERR	Service Error.
SHA	Secure Hash Algorithm.
SLP	Service Location Protocol.
SMTP	Simple Mail Transfer Protocol.
SREP	Service Reply.
SREQ	Service Request.

TC	Topology Control.
TCP	Transmission Control Protocol.
TLV	Type Length Value.
TRILL	Transparent Interconnection of Lots of Links.
TTL	Time to Live.
UAV	Unmanned Airborne Vehicle.
UDP	User Datagram Protocol.
UPnP	Universal Plug and Play.
URI	Uniform Resource Identifier.
URL	Uniform Resource Locator.
VANET	Vehicular Ad Hoc Network.
WiMAX	Worldwide Interoperability for Microwave Access.
WLAN	Wireless Local Area Network.
ZRP	Zone Routing Protocol.

Bibliography

[1] "International Graduate School on Mobile Communications," Homepage, TU Ilmenau, 2009 - 2014. [Online]. Available: http://www.tu-ilmenau.de/gs-mobicom

[2] IETF Mobile Ad-hoc Networks Group. [Online]. Available: http://datatracker.ietf.org/wg/manet/charter/

[3] J. Postel, "Internet Protocol," RFC 791 (Standard), Internet Engineering Task Force, Sep. 1981, updated by RFC 1349. [Online]. Available: http://www.ietf.org/rfc/rfc791.txt

[4] T. Simon and A. Mitschele-Thiel, "Micro Aerial Disaster Communication Systems," in *GMDS 2012/INFORMATIK 2012, Workshop: IT-Unterstützung im Emergency Management & Response*, Braunschweig, Germany, Sep. 2012.

[5] S. Schellenberg, "Namensgebung und Adressauflösung in heterogenen Netzen," in *12. Ilmenauer TK-Manager Workshop.* Ilmenau, Germany: Telekommunikations-Manager (TKM) e.V., Sep. 2012, pp. 99–108, - in German -.

[6] ITU, *Information technology - Open Systems Interconnection - Basic Reference Model: The basic model (ITU-T Recommendation X.200)*, International Telecommunications Union, Jul. 1994.

[7] R. Braden, "Requirements for Internet Hosts - Communication Layers," RFC 1122 (INTERNET STANDARD), Internet Engineering Task Force, Oct. 1989, updated by RFCs 1349, 4379, 5884, 6093, 6298, 6633, 6864. [Online]. Available: http://www.ietf.org/rfc/rfc1122.txt

[8] *Guidelines for 64-bit Global Identifier (EUI-64)*, IEEE Std., Mar. 1997. [Online]. Available: http://standards.ieee.org/develop/regauth/tut/eui64.pdf

[9] S. Deering and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” RFC 2460 (Draft Standard), Internet Engineering Task Force, Dec. 1998, updated by RFCs 5095, 5722, 5871. [Online]. Available: http://www.ietf.org/rfc/rfc2460.txt

[10] T. Berners-Lee, R. Fielding, and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” RFC 3986 (INTERNET STANDARD), Internet Engineering Task Force, Jan. 2005, updated by RFC 6874. [Online]. Available: http://www.ietf.org/rfc/rfc3986.txt

[11] P. Mockapetris, “Domain names - concepts and facilities,” RFC 1034 (Standard), Internet Engineering Task Force, Nov. 1987, updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936. [Online]. Available: http://www.ietf.org/rfc/rfc1034.txt

[12] D. Plummer, “Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware,” RFC 826 (Standard), Internet Engineering Task Force, Nov. 1982, updated by RFCs 5227, 5494. [Online]. Available: http://www.ietf.org/rfc/rfc826.txt

[13] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, “Neighbor Discovery for IP version 6 (IPv6),” RFC 4861 (Draft Standard), Internet Engineering Task Force, Sep. 2007, updated by RFC 5942. [Online]. Available: http://www.ietf.org/rfc/rfc4861.txt

[14] Internet Assigned Numbers Authority. [Online]. Available: www.iana.org

[15] R. Droms, “Dynamic Host Configuration Protocol,” RFC 2131 (Draft Standard), Internet Engineering Task Force, Mar. 1997, updated by RFCs 3396, 4361, 5494, 6842. [Online]. Available: http://www.ietf.org/rfc/rfc2131.txt

[16] P. Roensch, “Vergleich von Verfahren zur Adressvergabe in Mo-

bilen Ad-Hoc Netzwerken (MANETs)," 2013, Hauptseminar, Ilmenau University of Technology, - in German -.

[17] C. Bernardos, M. Calderon, and H. Moustafa, "Survey of IP address autoconfiguration mechanisms for MANETs," Internet Engineering Task Force (IETF), Internet-Draft, November 2008.

[18] S. Cheshire, B. Aboba, and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses," RFC 3927 (Proposed Standard), Internet Engineering Task Force, May 2005. [Online]. Available: http://www.ietf.org/rfc/rfc3927.txt

[19] S. Nesargi and R. Prakash, "MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network," in *21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 2. New York City, NY, USA: IEEE, Jun. 2002, pp. 1059 – 1068.

[20] M. Mohsin and R. Prakash, "IP Address Assignment in a Mobile Ad Hoc Network," in *Military Communications Conference (MILCOM)*, vol. 2. Anaheim, CA, USA: IEEE, Oct. 2002, pp. 856 – 861.

[21] H. Zhou, L. Ni, and M. Mutka, "Prophet Address Allocation for Large Scale MANETs," in *22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 2. San Francisco, CA, USA: IEEE, Mar. 2003, pp. 1304 – 1311.

[22] V. C. Giruka and M. Singhal, "A Localized IP-address Autoconfiguration Protocol for Wireless Ad-hoc Networks," in *4th ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots (WMASH) in conjunction with ACM MobiCom*. Los Angeles, CA, USA: ACM, Sep. 2006, pp. 101 – 108.

[23] T. Berners-Lee, L. Masinter, and M. McCahill, "Uniform Resource Locators (URL)," RFC 1738 (Proposed Standard), Internet Engineering Task Force, Dec. 1994, obsoleted by RFCs

4248, 4266, updated by RFCs 1808, 2368, 2396, 3986, 6196. [Online]. Available: http://www.ietf.org/rfc/rfc1738.txt

[24] P. Mockapetris, "Domain names - implementation and specification," RFC 1035 (Standard), Internet Engineering Task Force, Nov. 1987, updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966. [Online]. Available: http://www.ietf.org/rfc/rfc1035.txt

[25] F. Liers, M. Hager, S. Schellenberg, and J. Seitz, "Recursive Layering of Forwarding on Gates and Traffic Engineering Middleware for Ethernet," in *27th International Conference on Information Networking (ICOIN)*. Bangkok, Thailand: IEEE, Jan. 2013, pp. 600 – 605.

[26] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "The Locator/ID Separation Protocol (LISP)," RFC 6830 (Experimental), Internet Engineering Task Force, Jan. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc6830.txt

[27] C. Perkins, "IP Mobility Support for IPv4," RFC 3344 (Proposed Standard), Internet Engineering Task Force, Aug. 2002, obsoleted by RFC 5944, updated by RFCs 4636, 4721. [Online]. Available: http://www.ietf.org/rfc/rfc3344.txt

[28] C. Perkins, D. Johnson, and J. Arkko, "Mobility Support in IPv6," RFC 6275 (Proposed Standard), Internet Engineering Task Force, Jul. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6275.txt

[29] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Host Identity Protocol," RFC 5201 (Experimental), Internet Engineering Task Force, April 2008.

[30] O. Ponomarev and A. Gurtov, "Using DNS as an Access Protocol for Mapping Identifiers to Locators," in *Workshop on Routing in Next Generation (RiNG)*, Madrid, Spain, Dec. 2007.

[31] P. Begerow, S. Schellenberg, J. Seitz, T. Finke, and J. Schroeder, "Reliable Multicast in Heterogeneous Mobile Ad Hoc Networks," in *Workshop on Self-Organized Communication in Disaster Scenarios (SoCoDiS) in conjunction with Networked Systems 2013*, Stuttgart, Germany, Mar. 2013.

[32] P. Begerow, S. Krug, S. Schellenberg, and J. Seitz, "Buffer Management for Reliable Multicast over Delay Tolerant Networks," in *10th International Conference on Mobile Ad-hoc and Sensor Networks (IEEE MSN 2014)*. Maui, HI, USA: IEEE, Dec. 2014, pp. 171–178.

[33] S. Krug, P. Begerow, A. A. Rubaye, S. Schellenberg, and J. Seitz, "A Realistic Underlay Concept for Delay Tolerant Networks in Disaster Scenarios," in *10th International Conference on Mobile Ad-hoc and Sensor Networks (IEEE MSN 2014)*. Maui, HI, USA: IEEE, Dec. 2014, pp. 163–170.

[34] K. Wang and B. Li, "Efficient and Guaranteed Service Coverage in Partitionable Mobile Ad-hoc Networks," in *IEEE 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 2. New York City, NY, USA: IEEE, Jun. 2002, pp. 1089–109.

[35] A. Derhab, N. Badache, and A. Bouabdallah, "A Partition Prediction Algorithm for Service Replication in Mobile Ad Hoc Networks," in *2nd Annual Conference on Wireless On demand Network Systems and Services (WONS)*. St. Moritz, Switzerland: IEEE, Jan. 2005, pp. 236–245.

[36] S. Ahn and Y. Lim, "A Modified Centralized DNS Approach for the Dynamic MANET Environment," in *9th International Symposium on Communications and Information Technology*. Incheon, Korea: IEEE, Sep. 2009, pp. 1506 – 1510.

[37] S. Cheshire and M. Krochmal. (2011, Feb) Multicast DNS (IETF Internet-Draft). Expires: 18 August 2011.

[38] J. Jeong, J. Park, and H. Kim, “Name Service in IPv6 Mobile Ad-hoc Network connected to the Internet,” in *14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communication (PIMRC)*, vol. 2. Beijing, China: IEEE, Sep. 2003, pp. 1351–1355.

[39] P. Begerow, S. Krug, S. Schellenberg, and J. Seitz, “Robust Reliability-aware Buffer Management for DTN Multicast in Disaster Scenarios,” in *7th International Workshop on Reliable Networks Design and Modeling (RNDM)*. Munich, Germany: IEEE, October 2015, pp. 274–280.

[40] S. Cheshire and M. Krochmal. (2011, Feb) DNS-Based Service Discovery (IETF Internet-Draft). Expires: 31 August 2011.

[41] P. Engelstad, D. Van Thanh, and T. Jonvik, “Name Resolution in Mobile Ad-Hoc Networks,” in *10th International Conference on Telecommunications (ICT 2003)*, vol. 1. Anchorage, AK, USA: IEEE, Mar. 2003, pp. 388 – 392.

[42] P. Engelstad, D. Thanh, and G. Egeland, “Name Resolution in On-Demand MANETs and over External IP Networks,” in *International Conference on Communications (ICC)*, vol. 2. Anchorange, AK, USA: IEEE, May 2003, pp. 1024 – 1032.

[43] C. Jelger and C. Tschudin, “Underlay Fusion of DNS, ARP/ND, and Path Resolution in MANETs,” in *5th Scandinavian Workshop on Wireless Ad-hoc Networks (ADHOC)*, Stockholm, Sweden, May 2005.

[44] H. Zimmermann, “OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection,” *IEEE Transactions on Communications*, vol. 28, no. 4, Apr. 1980, pp. 425–432.

[45] R. Perlman, D. E. 3rd, D. Dutt, S. Gai, and A. Ghanwani, “Routing Bridges (RBridges): Base Protocol Specification,” RFC 6325 (Proposed Standard), Internet Engineering Task Force, Jul. 2011, updated by RFCs 6327, 6439. [Online]. Available: http://www.ietf.org/rfc/rfc6325.txt

[46] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys Tutorials*, vol. 7, no. 2, Feb. 2005, pp. 72–93.

[47] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561 (Experimental), Internet Engineering Task Force, Jul. 2003.

[48] M. K. Marina and S. R. Das, "Ad hoc On-demand Multipath Distance Vector Routing," in *Wireless Communications and Mobile Computing*, vol. 6, Nov. 2006, pp. 969 – 988.

[49] S. R. Biradar, K. Majumder, S. K. Sarkar, and P. C, "Performance Evaluation and Comparison of AODV and AOMDV," *International Journal on Computer Science and Engineering (IJCSE)*, vol. 02, no. 02, Jan. 2010, pp. 373 – 377.

[50] E. M. Royer and C. E. Perkins, "Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing," Internet Engineering Task Force (IETF), Tech. Rep., Jul. 2000.

[51] T. Finke, K. Klaric, J. Schroeder, S. Schellenberg, M. Hager, and J. Seitz, "Latency Avoidance by Route Assumption for Reactive Routing Protocols," in *5th International Conference on Ubiquitous and Future Networks (ICUFN)*. Da Nang, Vietnam: IEEE, Jul. 2013, pp. 707 – 711.

[52] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626 (Experimental), Internet Engineering Task Force, Oct. 2003.

[53] C. Mbarushimana and A. Shahrabi, "Comparative Study of Reactive and Proactive Routing Protocols Performance in Mobile Ad Hoc Networks," in *IEEE 21st International Conference on Advanced Information Networking and Applications (AINA)*, vol. 2. Niagara Falls, ON, Canada: IEEE, May 2007, pp. 679–684.

[54] P. Kuppusamy, K. Thirunavukkarasu, and B. Kalaavathi, "A Study and Comparison of OLSR, AODV and TORA Routing

Protocols in Ad Hoc Networks," in *International Conference on Electronics Computer Technology (ICECT)*, vol. 5. Kanyakumari, India: IEEE, Apr. 2011, pp. 143–147.

[55] W. Li, "Umschaltung von Routingprotokollen zur Laufzeit - Adaptives Routing," 2012, Hauptseminar, Ilmenau University of Technology, - in German -.

[56] S. Nanda, Z. Jiang, and D. Kotz, "A Combined Routing Method for Ad Hoc Wireless Networks," Dept. of Computer Science, Dartmouth College, Tech. Rep. TR2009-641, Feb. 2009.

[57] J. Hoebeke, "Adaptive ad hoc routing and its application in Virtual Private Ad Hoc Networks," Ph.D. dissertation, Department of Information Technology, Ghent University, Belgium, 2007.

[58] J. Hoebeke, I. Moerman, and P. Demeester, "Adaptive routing for mobile ad hoc networks," *EURASIP Journal on Wireless Communications and Networking*, no. 2012:216, Mar. 2012.

[59] N. Beijar, "Zone Routing Protocol (ZRP)," *Networking Laboratory Helsinki University of Technology Finland*, vol. 9, no. 4, 2001, pp. 427–438.

[60] T. Ramrekha and C. Politis, "A Hybrid Adaptive Routing Protocol for Extreme Emergency Ad Hoc Communication," in *19th International Conference on Computer Communications and Networks (ICCCN).* Zurich, Switzerland: IEEE, Aug. 2010, pp. 1–6.

[61] T. Finke, "Adaptives Routing in mobilen Ad-hoc-Netzwerken," in *12. Ilmenauer TK-Manager Workshop.* Ilmenau, Germany: Telekommunikations-Manager (TKM) e.V., Sep. 2012, pp. 39–42, - in German -.

[62] T. Finke, J. Schroeder, S. Schellenberg, M. Hager, and J. Seitz, "Address Resolution in Mobile Ad Hoc Networks using Adaptive Routing," in *7th International Conference on Systems and Networks Communications (ICSNC).* Lisbon, Portugal: IARIA, Nov. 2012, pp. 7–12.

[63] Network Simulator 3 (ns-3). [Online]. Available: http://www.nsnam.org/

[64] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek, "The Click Modular Router," in *ACM Transactions on Computer Systems*, Aug. 2000, pp. 263 – 297.

[65] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn, "BonnMotion - a Mobility Scenario Generation and Analysis Tool," in *3rd International ICTS Conference on Simulation Tools and Techniques (SIMUTools).* Torremolinos, Malaga, Spain: ICST, Mar. 2010, Article No. 51.

[66] Kubuntu Operating System. [Online]. Available: http://www.kubuntu.org

[67] S. Schellenberg, P. Begerow, M. Hager, J. Seitz, T. Finke, and J. Schroeder, "Implementation and Validation of an Address Resolution Mechanism using Adaptive Routing," in *27th International Conference on Information Networking (ICOIN).* Bangkok, Thailand: IEEE, Jan. 2013, pp. 95–100.

[68] B. Braem. An implementation of AODV in Click. [Online]. Available: http://www.pats.ua.ac.be/software/aodv/

[69] B. Braem. An implementation of OLSR in Click. [Online]. Available: http://www.pats.ua.ac.be/software/olsr/

[70] S. Schellenberg, S. Krug, T. Finke, P. Begerow, and J. Seitz, "Inter-Domain Routing and Name Resolution Using Border Nodes," in *International Conference on Computing, Networking and Communications (ICNC 2015).* Anaheim, CA, USA: IEEE, Feb. 2015, pp. 950–956.

[71] M. Tarasov, J. Seitz, and O. Artemenko, "A Network Partitioning Recovery Process in Mobile Ad-Hoc Networks," in *7th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob).* Sanghai, China: IEEE, Oct. 2011, pp. 32–36.

[72] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan, "A Cluster-based Approach for Routing in Dynamic Networks," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 2, Apr. 1997, pp. 49–64.

[73] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-Tolerant Networking Architecture," RFC 4838 (Informational), Internet Engineering Task Force, Apr. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4838.txt

[74] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," in *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM).* Karlsruhe, Germany: ACM, Aug. 2003, pp. 27–34.

[75] T. Simon and A. Mitschele-Thiel, "A Self-Organized Message Ferrying Algorithm," in *14th International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM).* Madrid, Spain: IEEE, Jun. 2013, pp. 1–6.

[76] K. Scott and S. Burleigh, "Bundle Protocol Specification," RFC 5050 (Experimental), Internet Engineering Task Force, Nov. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc5050.txt

[77] M. Lacage and T. R. Henderson, "Yet Another Network Simulator," in *Workshop on ns-2: the IP Network Simulator (WNS2) in conjunction with 1st International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS).* Pisa, Italy: ACM, Oct. 2006, Article No. 12.

[78] S. Krug, M. F. Siracusa, S. Schellenberg, P. Begerow, J. Seitz, T. Finke, and J. Schroeder, "Movement Patterns for Mobile Networks in Disaster Scenarios," in *8th IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC).* Sydney, Australia: IEEE, Jun. 2014, pp. 1 – 6.

[79] S. Schellenberg, S. Krug, J. Eckert, and J. Seitz, "Comparison of Application and Network Layer Name Resolution in Mobile Ad

hoc Networks," in *12th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN) in conjunction with the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM).* Cancun, Mexico: ACM, Nov. 2015, pp. 71–76.

[80] U. Sallakh, "Distributed Name Resolution Mechanism in Heterogeneous MANETs," 2012, Research Project, Ilmenau University of Technology.

[81] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *2nd International Conference on Simulation Tools and Techniques (SIMUTools).* Rome, Italy: ICST, Mar. 2009, Article No. 55.

[82] S. Schellenberg, A. Saliminia, J. Seitz, T. Finke, and J. Schroeder, "The Impact of Host Name Hashing on Name Resolution Traffic," in *5th International Conference on Ubiquitous and Future Networks (ICUFN).* Da Nang, Vietnam: IEEE, Jul. 2013, pp. 729 – 734.

[83] R. Rivest, "The MD5 Message-Digest Algorithm," RFC 1321 (Informational), Internet Engineering Task Force, Apr. 1992, updated by RFC 6151. [Online]. Available: http://www.ietf.org/rfc/rfc1321.txt

[84] D. Eastlake 3rd and P. Jones, "US Secure Hash Algorithm 1 (SHA1)," RFC 3174 (Informational), Internet Engineering Task Force, Sep. 2001, updated by RFC 4634. [Online]. Available: http://www.ietf.org/rfc/rfc3174.txt

[85] R. Ge, G. Di Crescenzo, M. Fecko, and S. Samtani, "Efficient and secure indirect-address service discovery in manet," in *IEEE Military Communications Conference (MILCOM)*, vol. 3. Atlantic City, NJ, USA: IEEE, Oct. 2005, pp. 1514 – 1520.

[86] S. Schellenberg, A. Saliminia, S. Krug, J. Seitz, T. Finke, and J. Schroeder, "Routing-based and Location-aware Service Discov-

ery in Mobile Ad-hoc Networks," in *28th International Conference on Information Networking (ICOIN).* Phuket, Thailand: IEEE, Feb. 2014, pp. 7–12.

[87] Z. Ansar, "Service Discovery Mechanism in Mobile Ad Hoc Networks (MANETs)," 2012, Research Project, Ilmenau University of Technology.

[88] A. Saliminia, "Implementierung und Simulation eines Diensterkennungsmechanismus basierend auf Namensaufloesung ueber Routing in Mobilen Ad-hoc-Netzwerken," Master's thesis, Ilmenau University of Technology, 2013, - in German -.

[89] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol, Version 2," RFC 2608 (Proposed Standard), Internet Engineering Task Force, Jun. 1999, updated by RFC 3224. [Online]. Available: http://www.ietf.org/rfc/rfc2608.txt

[90] "Universal Plug and Play," Microsoft Corporation, http://upnp.org/, Oct. 1999.

[91] "Jini Architecture Specification Version 2.0," Sun Microsystems, Jun. 2003.

[92] R. Koodli and C. E. Perkins, "Service Discovery in On-Demand Ad Hoc Networks," IETF, Oct. 2002, internet Draft.

[93] M. Serhani and Y. Gadallah, "A Service Discovery Protocol for Emergency Response Operations Using Mobile Ad Hoc Networks," in *6th Advanced International Conference on Telecommunications (AICT).* Barcelona, Spain: IARIA, May 2010, pp. 280 – 285.

[94] M. Heni and R. Bouallegue, "Adaptive Service Discovery and Proactive Routing Protocol for Mobile Ad Hoc Network ," in *11th Mediterranean Microwave Symposium (MMS).* Hammamet, Tunisia: IEEE, Sep. 2011, pp. 193 – 196.

[95] F. Zhang, "Vergleich von Verfahren zur Geocast-Adressierung," 2013, Hauptseminar, Ilmenau University of Technology, - in German -.

[96] R. Satannavar and V. S. Raghavendra, “Concept of Geocast over Name Resolution in MANETs,” 2013, Research Project, Ilmenau University of Technology.

[97] O. Artemenko, T. Simon, A. Mitschele-Thiel, D. Schulz, and M. R. S. Ta, “Comparison of Anchor Selection Algorithms for Improvement of Position Estimation During the Wi-Fi Localization Process in Disaster Scenario,” in *37th IEEE Conference on Local Computer Networks (LCN).* Clearwater, Florida, USA: IEEE, Oct. 2012, pp. 44 – 49.

[98] Y.-B. Ko and N. H. Vaidya, “Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms,” in *2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA).* New Orleans, LA, USA: IEEE, Feb 1999, pp. 101–110.

[99] I. Stojmenovic, A. P. Ruhil, and D. Lobiyal, “Voronoi diagram and convex hull based geocasting and routing in wireless networks,” in *8th IEEE International Symposium on Computers and Communication (ISCC)*, vol. 1. Kemer, Turkey: IEEE, Jun. 2003, pp. 51–56.

[100] T. Camp and Y. Liu, “An Adaptive Mesh-based Protocol for Geocast Routing,” *Journal of Parallel and Distributed Computing*, vol. 63, no. 2, Feb. 2003, pp. 196–213.

[101] Y.-B. Ko and N. H. Vaidya, “GeoTORA: A Protocol for Geocasting in Mobile Ad Hoc Networks,” in *8th International Conference on Network Protocols (ICNP).* Osaka, Japan: IEEE, Nov. 2000, pp. 240 – 250.

[102] W.-H. Liao, Y.-C. Tseng, K.-L. Lo, and J.-P. Sheu, “GeoGRID: A Geocasting Protocol for Mobile Ad Hoc Networks Based on GRID,” *Journal of Internet Technology*, vol. 1, no. 2, Dec. 2000, pp. 23–32.