Fachhochschule Köln

Cologne University of Applied Sciences

Fakultät für Informations-, Medien- und Elektrotechnik

Studiengang Bachelor Of Information Engineering

Institut für Nachrichtentechnik

Labor für Datennetze

Bachelor-Thesis

Thema: **Automatische Optimierung von Routing-Einträgen mit Prioritätssteuerung**

Student:	**Alexander Kniwel**
Matr. Nr.:	**11045840**
Referent:	**Prof. Dr. Andreas Grebe**
Korreferent:	**Dipl. Ing. Stefan Abu Salah**

Abgabedatum: **31.08.2007**

Hiermit versichere ich, dass ich die vorliegende Bachelor-Thesis selbstständig und nur unter Benutzung der angegebenen und bei Zitaten kenntlich gemachten Quellen und Hilfsmittel angefertigt habe.

Alexander Kniwel

Danksagung:
An dieser Stelle möchte ich mich bei meinen Eltern bedanken, die mich während der Studienzeit und besonders während der Bachelor-Thesis unterstützt haben. Ebenso möchte ich mich bei allen meinen Freunden und Kommilitonen bedanken, im Besonderen bei B. Sc. Karin Schuster und Frank Schuster für das Korrekturlesen und für die zahlreich eingebrachten Tipps.
Mein weiterer Dank gilt meinem Referenten Herrn Prof. Dr. Andreas Grebe und meinem Korreferenten Herrn Dipl.-Ing. Stefan Abu Salah, sowie allen Mitarbeitern des Bereichs Datennetze.

Inhaltsverzeichnis

1 Einleitung und Ziel dieser Arbeit 6
2 Linux 7
2.1 Ubuntu 7
2.2 Shell 8
2.2.1 Shell-Programmierung 8
2.3 Zenity 11
3 Lokale Netzwerke 13
3.1 Drahtgebundene Netze 13
3.2 Drahtlose Netze 14
3.2.1 WLAN-Standards 14
3.2.2 Verschlüsselungsverfahren 15
3.2.3 Verschlüsselung unter Ubuntu 17
3.3 Adressierung 18
3.3.1 statische Adressvergabe 20
3.3.2 dynamische Adressvergabe 21
3.3.3 Interfacekonfiguration 23
4 Routing 25
4.1 Die Routingtabelle 25
4.2 Routing vs. Forwarding 27
4.3 Routenmanipulation unter Ubuntu 27
4.4 Routing-Protokolle 28
5 Entwicklung 29
5.1 Lösungsansatz 29
5.2 Implementierung NetRoam 30
5.3 Notwendige Modifikationen 50
5.4 Schnittstellen zum Webfrontend 53
6 Messung 58
6.1 Software für die Messungen 58
6.1.1 Wireshark 58
6.1.2 Ping 58
6.1.3 Programm utime 59
6.2 Aufbau der Messumgebung 59
6.3 Messergebnisse 60
6.3.1 Wechsel von LAN auf WLAN 61
6.3.2 Wechsel von WLAN zu WLAN 62
6.3.3 Wechsel von WLAN auf LAN 63
6.3.4 Messungen des Programms utime 64
6.4 Leistungsbewertung 66
7 Installation 67
7.1 Voraussetzungen für NetRoam 67
7.1.1 Hardware 67
7.1.2 Software 67
7.2 Voraussetzungen für das Webfrontend 68
7.2.1 Software 68
7.3 Installation von NetRoam 69
8 Bedienung 70
8.1 NetRoam 70

8.2 Webfrontend 71
9 Schlussbetrachtung 78
10 Abbildungsverzeichnis 79
11 Tabellenverzeichnis 80
12 Literaturverzeichnis 81
12.1 Printmedien 81
12.2 Internetquellen 81
13 Anhang 82
13.1 Script install 82
13.2 Script netroam 83
13.3 Script netroamstart 84
13.4 Script start 84
13.5 Script deinstall 84
13.6 Programm utime 85
13.7 Beispielkonfiguration der wpa_supplicant.conf für WPA 85
13.8 Beispielkonfiguration der wpa_supplicant.conf für WPA2 85
13.9 Beispielkonfiguration der wpa_supplicant.conf für WPA2-RADIUS 86

1 Einleitung und Ziel dieser Arbeit

Heutzutage ist das Verlangen nach dem Internet größer denn je. Noch beschränkt sich der Internetzugang meist auf einen privaten Anschluss. In einigen Jahren wird jedoch die Anzahl, der bereits jetzt schon vorhandenen öffentlichen drahtlosen Zugangspunkten, so genannten Hot Spots, weiter gestiegen sein. Aber auch im privaten Bereich werden die Internetzugänge, drahtlose (WLAN[1]) wie drahtgebundene (LAN[2]), weiter steigen. Dann wird eine nahezu flächendeckende Internetverfügbarkeit bestehen.

Um die neuen Techniken wie Voice over IP (kurz VoIP) oder das normale Internetsurfen über mobile Endgeräte, wie beispielsweise Notebooks, kostengünstig und überall nutzen zu können, besteht die Möglichkeit, sich per WLAN mit diesen Hot Spots oder mit drahtgebundenen Zugangspunkten zu verbinden. Dabei entsteht jedoch ein Problem. Führt man beispielsweise mit einem Notebook, welches mit einem Hot Spot verbunden ist, ein VoIP-Gespräch, so ist man während des Gespräches an diesen Hot Spot gebunden. Da die meisten Hot Spots nicht miteinander verbunden sind, funktioniert auch das klassische WDS[3] nicht, was das Wechseln von mobilen Zugangspunkten ermöglichen würde. Dasselbe Problem würde auch beim Wechsel von einem drahtlosen Netzwerk zu einem drahtgebundenen Netzwerk auftreten, da auch dort keine Möglichkeit besteht, beim Wechsel die Verbindung aufrecht zu erhalten.

Ziel dieser Arbeit ist es nun, eine Software zu entwickeln, die die Möglichkeit bietet, beim Internetsurfen bequem zwischen bekannten Netzen, ohne Verbindungsabbrüche, zu wechseln. Dabei soll man sowohl zwischen kabelgebundenen Netzen und drahtlosen Netzen, sowie zwischen zwei drahtlosen Netzen ohne Einschränkung wechseln können. Hierfür soll eine Prioritätssteuerung eingeführt werden, die es ermöglicht die Priorität entweder auf LAN oder auf WLAN zu setzen. Damit eine möglichst große Anzahl an Access Points genutzt werden können, sollen die Verschlüsselungsmethoden WEP[4], WPA[5], WPA2 und WPA2-RADIUS[6] unterstützt werden.

Des Weiteren soll das Programm eine Basis für SIP[7]-Programme schaffen, mit Hilfe derer das Wechseln der Netzwerke auch während eines Gespräches über VoIP möglich ist.

1 Wireless Local Area Network
2 Local Area Network
3 Wireless Distribution System
4 Wired Equivalent Privacy
5 Wi-Fi Protected Access
6 Remote Authentication Dial-In User Service
7 Session Initiation Protocol

2 Linux

Linux ist ein frei verfügbares Betriebssystem, welches den Multitasking[8]- und Multiuserbetrieb unterstützt. Frei verfügbar bedeutet unter anderem, dass der Quellcode für jedermann frei zugänglich ist. Dies garantiert die GPL[9], unter welches Linux von Beginn an gestellt wurde. Linux wurde im Jahre 1991 von Linus Torvalds entwickelt und besitzt heutzutage eine Vielzahl weiterer Entwickler, welche sowohl Unternehmen als auch Einzelpersonen sind. Mit dem Begriff Linux ist im Allgemeinen der Kernel gemeint, der beispielsweise die Ressourcen und die Anwender verwaltet. Der Kernel liegt aktuell in der Version 2.6.22.6 vor. Um den Kernel, der einen wesentlichen Teil des Systems ausmacht, wirklich nutzen zu können, ist jedoch weitere Software nötig.

Die so genannten Linux-Distributionen sind Pakete, in denen verschiedene Software zusammengestellt ist. Es gibt eine Vielzahl von Distributionen, die alle aus einem Linux-Kernel bestehen. Die bekanntesten Distributionen sind unter anderem Debian, SuSE und Ubuntu mit seinen verschiedenen Derivaten, um nur ein paar zu nennen.

Im nächsten Kapitel wird auf die Distribution Ubuntu mit seinem Derivat Kubuntu eingegangen, welches in dieser Bachelorarbeit verwendet wurde.

2.1 Ubuntu

Die Distribution Ubuntu startete im Jahre 2004 mit der grafischen Benutzeroberfläche GNOME[10]. Im Laufe der Zeit erschienen jedoch noch weitere Derivate wie Xubuntu, welches die Arbeitsumgebung Xfce einsetzt. Außerdem kam noch das mittlerweile weit verbreitete Derivat Kubuntu hinzu, welches KDE[11] als grafische Benutzeroberfläche einsetzt.

Kubuntu basiert auf der Distribution Ubuntu und unterscheidet sich im Großen und Ganzen nur in der grafischen Oberfläche. Kubuntu gibt es seit der Ubuntu Version 5.04 und besitzt, wie schon beschrieben, die Desktopumgebung KDE, welche auch für den Namen von Kubuntu verantwortlich ist. Die aktuellste Kubuntu Version ist die Version 7.04 „Feisty Fawn“. Kubuntu wird in regelmäßigen Abständen von 6 Monaten veröffentlicht.

8 Unter dem Multitaskingbetrieb versteht man die gemeinsame Nutzung von Systemressourcen, bei dem der Eindruck entsteht, dass mehrere Prozesse gleichzeitig laufen.
9 Die GPL (General Public License) wird für frei verfügbare Software von der Free Software Foundation vergeben.
10 GNU Network Object Model Environment
11 K Desktop Environment

2.2 Shell

Die Shell ist der Kommandointerpreter eines Linux-Systems. Ein Linux-System verfügt nicht nur über eine Shell. Der Benutzer hat die Möglichkeit aus einer Vielzahl von Shells zu wählen.

Die Standardshell unter Kubuntu ist die Bourne-Again-Shell (Bash), die auch in dieser Arbeit verwendet wurde. Die Bourne-Shell (sh), die Mutter aller Shells, wurde im Jahre 1978 von S.R. Bourne entwickelt und ist die Standard-Shell in UNIX-Systemen. Des Weiteren gibt es noch die Korn-Shell (ksh), die eine Erweiterung der Bourne-Shell ist, die Z-Shell (zsh), die A-Shell (ash), die Restricted-Shell (rbash, rzsh) und die TC-Shell (tcsh).

In dieser Arbeit wurde, wie schon beschrieben, die Bourne-Again-Shell verwendet. Der Grund dafür ist, dass es die Standardshell unter Kubuntu ist und es eine große Anzahl an Literatur für diese Shell gibt, welches das Einarbeiten im Hinblick auf die Shell-Programmierung in Scripten einfacher macht.

2.2.1 Shell-Programmierung

Linux bietet die Möglichkeit mit so genannten Shell-Scripten mehrere Kommandos aneinander zu hängen und sie dann mit dem Aufruf des Scripts zusammen auszuführen. Dafür werden die Kommandos untereinander in eine Textdatei geschrieben und abgespeichert, wobei die Textdateien bzw. Scripte keine besondere Dateiendung benötigen. Einzig beim Ausführen des Scripts muss darauf geachtet werden, dass das Script mit dem Befehl

```
sh meinscript
```

gestartet wird. Eine weitere Möglichkeit ein Script zu starten besteht darin, vor dem ersten Starten die execute-Rechte für das jeweilige Script zu setzen. Dies geschieht mit dem Befehl

```
chmod +x meinscript
```

Das Script kann nun gestartet werden, indem man den Namen des Scripts aufruft:

```
./meinscript
```

Das nachfolgende Beispiel soll noch einmal die Vorgehensweise verdeutlichen. Dafür wird eine Datei mit dem Namen meinscript erzeugt, in welche folgendes geschrieben wird:

```
#!/bin/bash
echo "Hallo Welt!"
ps | grep "bash"
```

Die erste Zeile bestimmt mit welcher Shell das Script ausgewertet werden soll. In diesem Fall wird die Bash-Shell verwendet. In der zweiten Zeile wird eine normale Ausgabe auf der Konsole mit Hilfe des *echo* Befehls gemacht. Als Letztes wird noch ein Befehl ausgeführt, der die gerade laufenden Prozesse auflistet. Mit Hilfe des *grep* Befehls werden hier nur die Prozesse aufgelistet, die das Wort bash enthalten.

Die komplette Ausgabe des Scripts sieht wie folgt aus:

```
Hallo Welt!
5147 pts/3    00:00:00 bash
```

Wie schon im Beispiel gezeigt, gibt es in der Shell-Programmierung einige helfende Befehle. Es soll nun auf zwei für diese Arbeit wichtigen Kommandos eingegangen werden, der grep- und der awk-Befehl.

grep

Der Name *grep* steht für „**G**lobal search for a **R**egular **E**xpression and **P**rint out matched lines“. Dies bedeutet, dass mit Hilfe des *grep*-Befehls nach bestimmten Mustern in einer oder mehreren Dateien gesucht werden kann. Wird der Ausdruck in einer Datei gefunden, so leitet *grep* die gefundene Zeile, einen gefundenen Bereich, je nachdem was der Benutzer wünscht, an die Standardausgabe. Dabei wird die Eingabedatei nicht verändert. *Grep* muss mit folgender Syntax aufgerufen werden:

```
grep suchstring datei1 [datei2] ...
```

Auch besteht bei *grep* die Möglichkeit, dass mit Hilfe der Pipe (|) als Eingabe die Ausgabe eines anderen Befehls verwendet wird. Ein mögliches Beispiel wäre folgendes:

```
cat meinscript | grep Welt
```

Bei diesem Beispiel würde die Ausgabe „Hallo Welt!“ lauten, da mit cat die Datei ausgegeben wird, diese Ausgabe aber wieder der Eingabe des grep Befehls dient, und grep in dieser Eingabe nach dem String „Welt“ sucht und die gefundene Zeile ausgibt.

Neben dem normalen *grep* Befehl gibt es noch die Erweiterungen *egrep* (Extended Grep), *fgrep* (Fixed Grep), welches schneller als *grep* arbeitet, dafür aber auch weniger reguläre Ausdrücke versteht und *rgrep* (Rekursive Grep), welches bei rekursiven Suchen in Verzeichnissen mit Unterverzeichnissen eingesetzt wird.

In der folgenden Tabelle sind kurz die wichtigsten Optionen aufgelistet, die dem *grep* Kommando beim Starten übergeben werden können.

-A anzahl	Gibt die gefundene und alle nachfolgenden Zeilen aus
-B anzahl	Gibt die gefundene und alle vorherigen Zeilen aus
-b	Gibt die Position der gefundenen Stelle mit aus
-c	Gibt die Gesamtanzahl der gefundenen Stellen aus
-n	Gibt die Zeilennummer der gefundenen Stelle aus
-v	Gibt nur die Zeilen aus, in denen der Ausdruck nicht vorkommt
-E	Suche nach mehreren durch | getrennten Ausdrücken

Tabelle 1: Die wichtigsten grep-Kommandos

awk

Awk ist eigentlich nicht, wie oben beschrieben, ein einfaches Kommando, sondern genau genommen eine richtige Programmiersprache. Es ist der Programmiersprache C sehr ähnlich und wurde im Jahre 1977 von den drei Entwicklern Aho, Weinberg und Kerninghan entwickelt. Der Anfangsbuchstabe ihrer Namen ist auch zugleich für den Namen *awk* verantwortlich.

Awk bietet mächtige Hilfsmittel, um große Textdateien auszuwerten und zu bearbeiten. Allerdings eignet sich *awk* nicht zur Programmierung von größeren Programmen, da es relativ langsam ist. In dieser Arbeit wurden nur die grundlegenden Funktionen von *awk* benutzt, beispielsweise das Auftrennen von Strings in seine einzelnen Felder.

Die allgemeine Syntax lautet folgendermaßen:

```
awk [Muster] [Anweisung] [Datei]
```

Dabei ist der Parameter Muster ein regulärer Ausdruck und eine optionale Angabe. Der zweite Parameter ist, genau wie der Erste, eine optionale Angabe, allerdings muss mindestens einer der beiden Parameter vorhanden sein. Dieser zweite Parameter enthält den eigentlichen Programmteil und muss in geschweiften Klammern stehen. Der letzte

Parameter ist, wie die beiden ersten Parameter, eine optionale Angabe. Wird hier jedoch keine Datei angegeben, so ließt awk von der Standardeingabe oder aus der Pipe.

2.3 Zenity

Mit Zenity ist es möglich, aus Shell-Scripten GUI-Fenster zu erzeugen. Dank diesem Programm ist es möglich, bei einfachen GUI-Fenstern auf weit komplexere Programmiersprachen wie Pearl oder Phyton zu verzichten. Zenity ist in der Lage, je nachdem was für Parameter übergeben werden, verschiedenste Fenster anzuzeigen. So ist es beispielsweise möglich, einfache Informations-, Frage- oder Warndialoge darzustellen. Aber auch komplexere Dialoge wie Kalender, Listen, bis hin zu Dialogen, in denen Dateien ausgewählt werden können, sind möglich. Dabei ist der Grundaufbau bzw. die Syntax bei allen Dialogen nahezu identisch:

```
zenity --Art [--width] [--height] [--title] [--text]
```

Der Aufruf beginnt immer mit dem Schlüsselwort "zenity". Die wichtigsten Arten der Dialoge, mit den dafür zusätzlich wichtigsten Parametern, sind in der unten stehenden Tabelle 2 angegeben. Die zwei Parameter –width und –height geben die Größe des Fensters an und sind optional. Werden sie nicht gesetzt, wählt Zenity die für dieses Fenster beste Größe. Der vierte Parameter setzt den Titel des Fensters und ist ebenfalls optional. Wird er nicht angegeben, so setzt Zenity, je nachdem welcher Dialog gewählt wurde, den Titel selbstständig. Der letzte Parameter gibt den Text an, der in dem jeweiligen Fenster angezeigt werden soll. Wird er nicht gesetzt, so fügt Zenity einen Text ein.

Art	Beschreibung	Parameter	
info	Informations-Dialog	-	
entry	Eingabe-Dialog	-	
question	Frage-Dialog	-	
warning	Warn-Dialog	-	
error	Fehler-Dialog	-	
calendar	Kalender-Dialog	--day=INT --month=INT --year=INT	- Setzt den Tag - Setzt den Monat - Setzt das Jahr
file-selection	Datei-Auswahl	-- filename=FILENAME -- multiple	- Setzt den Dateinamen - Ermöglicht Mehrfachauswahl
progress	Progress-Status	-- percentage=INT -- auto-close	-Setzt Start Prozentsatz -Schließt bei 100% automatisch
list	Listen-Auswahl	-- column=STRING -- checklist, --radiolist	- Setzt den Spaltenkopf - Button in der ersten Spalte

Tabelle 2: Mögliche Zenity-Arten

3 Lokale Netzwerke

Ein lokales Netz ist ein Netzwerk, um zumeist private Rechner als auch Rechner in Firmen zu verbinden. Dabei gibt es mehrere Möglichkeiten dies zu realisieren. Zum einen gibt es die drahtgebundene, zum anderen die drahtlose Möglichkeit. Beides wird im folgenden Kapitel erörtert.

3.1 Drahtgebundene Netze

Die am häufigsten eingesetzte Technik ist heutzutage das Ethernet. Es wurde von der IEEE[12] mit der Norm 802.3 standardisiert. Weitere bekannte Techniken sind der Token Bus (802.4) und der Token Ring (802.5). Die beiden letzteren Techniken wurden weitestgehend vom Ethernet verdrängt und finden heutzutage kaum oder gar keine Verwendung mehr. Ethernet wird über Twisted-Pair- oder Glasfaserkabel übertragen. Die anfängliche Ethernet-Netzwerktechnologie 10Base-T wurde mit der Zeit von 100Base-T und mittlerweile auch von 1000Base-T verdrängt. Obwohl 100Base-T noch in den meisten Fällen eingesetzt wird, findet 1000Base-T heutzutage schon immer mehr an Beachtung. Mit 10GBase-T steht jedoch schon die nächste Technologie bereit. Neben den xBaseT-Techniken stehen noch eine weitere Anzahl an Techniken, wie zum Beispiel 10Base5, 10Base-SX über Glasfaser, 100BaseFX oder 1000Base-SX ebenfalls über Glasfaser, zur Verfügung.

Die allgemeine Reichweite, die mit Ethernet über normales Twisted-Pair Kabel erreicht werden kann, beträgt bis zu 100 m, abhängig von der verwendeten Technologie. Im Folgenden soll kurz auf das Ethernet-Paket eingegangen werden.

Der Ethernet-Frame hat eine Maximalgröße von 1518 Byte und muss mindestens 64 Byte groß sein. Wird dies nicht erreicht, so wird der Rahmen mit Padding-Bits aufgefüllt. Die ersten 8 Byte gehören nicht zum eigentlichen Ethernet-Rahmen, sondern sind nur für die Taktsynchronisation zuständig. Die ersten 12 Byte des Rahmens geben mit je 6 Byte die Ziel- und Quelladresse an. Weiterhin stehen 2 Byte für Typ/Länge zur Verfügung. Für die eigentlichen Daten bleiben von den 1518 Byte nur maximal 1500 Byte übrig. Die letzten 4 Byte des Rahmens werden mit FCS (Frame Check Sequence) bezeichnet und sind für die Fehlererkennung zuständig.

12 Institute of Electrical and Electronics Engineers

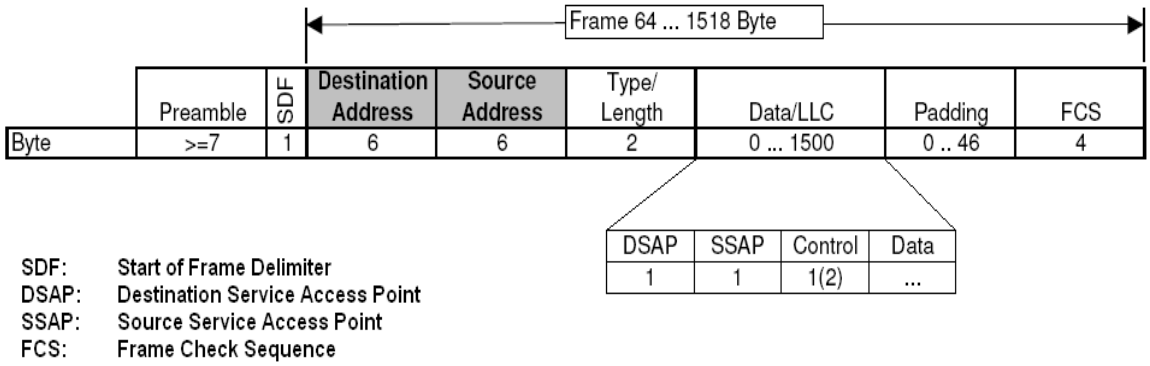

Abbildung 1: Ethernet-Frame frei nach [5]

3.2 Drahtlose Netze

Drahtlose Netzwerke werden mit WLAN bezeichnet und gehören der Gruppe der IEEE 802.11 Standards an. Drahtlose Netze sind, genau wie die drahtgebundenen Netze, für private Haushalte und für Firmen gedacht. Dank ihrer relativ hohen Reichweite werden sie gerne an Orten eingesetzt, wo sich drahtgebundene Netze nicht rentieren. Durch die relativ hohe Reichweite wurde jedoch ein neues Problem geschaffen, die Sicherheit. Für Angreifer ist es wesentlich leichter in Netzwerke einzudringen, die drahtlose Komponenten enthalten, als in rein drahtgebundene Netzwerke. Um diesem Problem entgegenzutreten wurden verschiedene Verschlüsselungsverfahren entwickelt, auf die in Kapitel 3.2.2 eingegangen wird. Zunächst einmal werden jedoch einige WLAN Standards vorgestellt.

3.2.1 WLAN-Standards

Die beiden ersten Standards 802.11a und 802.11b wurden im Jahre 1999 standardisiert. Der Netzwerkmodus 802.11a sendet im 5 GHz Bereich, d.h. bei Frequenzen von 5,725 GHz bis 5,850 GHz und schafft eine Übertragungsrate von 54 MBit/s bei einer Reichweite von ca. 15 m – 25 m. Da in Europa dieser Frequenzbereich noch zusätzlich vom Militär benutzt wird hat es zur Folge, dass dieser Standard nur in geschlossenen Gebäuden eingesetzt werden darf. Der zweite Netzwerkmodus von 1999 ist im 2,4 GHz-Bereich angesiedelt und schafft eine maximale Bruttodatenrate von ca. 11 MBit/s. Er wurde, genau wie der Netzwerkmodus 802.11a, in verschiedene Kanäle aufgeteilt, um das relativ große Frequenzband besser unterteilen zu können.

Heutzutage sind diese beiden Modi von dem zum 802.11b abwärtskompatiblen Netzwerkmodus 802.11g verdrängt worden. Der Vorteil dieses Standards liegt darin, dass

er, genau wie der Standard 802.11b, je nach Land, maximal 14 Kanäle benutzt (Europa 13, USA 11, Japan 14), dabei aber eine maximale Bruttodatenrate von 54 MBit/s (ca. 16 MBit/s Netto) erreicht. Er wurde im Jahre 2003 standardisiert, sendet ebenfalls im 2,4 GHz-Bereich und besitzt, genau wie der Modus 802.11b, in Gebäuden eine maximale Reichweite von 50 m und außerhalb von Gebäuden, mit speziellen Antennen, bis zu 300 m.

Der neuste Netzwerkmodus ist der Modus 802.11n. Er soll eine Bruttodatenrate von 540 MBit/s erreichen und zu den beiden Modi 802.11b und 802.11g abwärtskompatibel sein. Er ist bis zum heutigen Tag allerdings noch nicht standardisiert worden. Dieser neue Modus wird gebraucht, um die großen Datenmengen von beispielsweise Video- und Sprachübertragungen ohne Qualitätsverluste übertragen zu können. Aus diesem und anderen Gründen soll auch ein QoS[13]-Management integriert werden. Eine weitere Neuheit gegenüber den alten Netzwerkmodi ist, dass nicht nur eine Antenne, sondern mehrere Antennen zum Einsatz kommen sollen. Da es aber noch kaum Hardware für diesen Modus gibt, die obendrein zwischen verschiedenen Herstellern auch noch nicht 100%-ig kompatibel ist, was auf den noch fehlenden Standard zurück zu führen ist, wird es noch einige Zeit dauern, bis dieser Modus sich wirklich durchsetzen wird.

3.2.2 Verschlüsselungsverfahren

Ein wichtiges Thema im Bereich WLAN ist die Verschlüsselung. Es gibt die Verschlüsselungsmethoden WEP, WPA und WPA2, deren Funktionsweisen im Einzelnen vorgestellt werden.

WEP

Die erste Verschlüsselungsmethode, die im Zuge des WLAN-Standards IEEE 802.11 eingeführt wurde, war WEP. WEP arbeitet mit einem 64 Bit oder 128 Bit langen Schlüssel. Die ersten 24 Bit gehören zu dem so genannten Initialisierungsvektor (IV), die letzten 40 Bit, bzw. 104 Bit sind der eigentliche Schlüssel. Dabei ist der IV nicht konstant sondern zufällig und wird bei jeder zu übertragenden Nachricht neu erstellt und mit dem Schlüssel verknüpft. Der Schlüssel muss zum Entschlüsseln allen Teilnehmern bekannt sein. Das Ergebnis dieser Verknüpfung ist die Eingangsfolge des so genannten RC4-Algorithmus[14]. Mit Hilfe dieser Eingangsfolge erstellt der RC4-Algorithmus einen Keyframe. Des Weiteren

13 Quality of Service
14 Beim RC4-Algorithmus wird die Eingangsfolge Byte für Byte mit einer Zufallsfolge XOR verknüpft

wird an die zu übertragende Nachricht ein 32 Bit langer Prüfwert angehangen, welcher mittels zyklischer Redundanzprüfung (CRC[15]-32) erzeugt wurde. Das Ergebnis wird nun mit dem vom RC4-Algorithmus erzeugten Keyframe XOR verknüpft. Das Ergebnis dieser XOR-Verknüpfung und der IV, welcher vor die Verknüpfung geschrieben wird und zum späteren Entschlüsseln benötigt wird, bilden die fertige Nachricht.

Da der IV bei WEP nur 24 Bit lang ist, muss er sich in einem voll ausgelasteten Netz spätestens nach ca. 5 Stunden wiederholen. Werden nun zwei Pakete mit identischem IV miteinander XOR verknüpft, so lässt sich daraus der Schlüssel berechnen. WEP ist somit relativ unsicher und sollte nicht mehr eingesetzt werden.

WPA

Die Verschlüsselungsmethode, die den Zeitraum bis zum Erscheinen der Verschlüsselungsmethode WPA2 überbrücken sollte, heißt WPA und wurde im April 2003 eingeführt. WPA funktioniert im Wesentlichen wie WEP, hat aber noch einige zusätzliche Sicherheitsmechanismen an Bord und soll so die Schwächen von WEP beheben. WPA muss in zwei Arten unterteilt werden. Die erste Art ist WPA-PSK[16] mit TKIP[17]. Bei dieser Art werden die sich anmeldenden Nutzer mittels Pre-Shared-Key (PSK) authentifiziert, welcher sowohl bei jedem Nutzer als auch bei dem Access Point manuell eingetragen werden muss. Bei der zweiten Art, WPA-EAP[18] werden die Nutzer mit Hilfe eines (RADIUS-) Servers authentifiziert (IEEE 802.1x). Hier kommt kein spezieller Schlüssel zum Einsatz, sondern jeder Benutzer muss sich mit einem Benutzernamen und einem Passwort, welcher zuvor beim Server eingetragen werden muss, anmelden. Beide Arten benutzen das so genannte Temporal Key Integrity Protocol (TKIP). Dieses Protokoll benutzt, wie WEP, den RC4-Algorithmus zur Verschlüsselung, allerdings sind dabei die Schlüssel dynamisch und ändern sich, nachdem ein Datenpaket von 10 kB übertragen wurde. Um die Schwachstelle des Initialisierungsvektors (IV) in den Griff zu bekommen, wird bei WPA nicht wie bei WEP ein IV mit 24 Bit verwendet, sondern einer mit der Größe von 48 Bit, der in einen Lo-Teil (16 Bit) und einen Hi-Teil (32 Bit) aufgeteilt wird. Wird nun ein Datenpaket übertragen, so erhöht sich der Lo-Teil um ein Bit. Sollte am Empfänger ein schon vorhandener IV eintreffen, so wird das komplette Paket verworfen. Da TKIP noch zusätzlich die MAC-Adresse des Senders zum Erzeugen eines Schlüssels benutzt, ist sichergestellt, dass der

15 Cyclic Redundancy Check
16 Pre-Shared-Key
17 Temporal Key Integrity Protocol
18 Extensible Authentication Protocol

Schlüssel bei mehreren Sendern unterschiedlich ist.

Eine zusätzliche Möglichkeit von WPA ist die Verschlüsselung mit CCMP[19]/AES[20]. Die Verschlüsselungsmethode CCMP/AES funktioniert genau wie TKIP sowohl mit dem PSK als auch mit einem (RADIUS-) Server und ist dieselbe Methode, die auch bei WPA2 zur Verschlüsselung eingesetzt wird.

WPA2

WPA2 wurde im Juni 2004 unter dem IEEE-Standard 802.11i eingeführt. WPA2 benutzt, wie schon im vorherigen Abschnitt erwähnt, standardmäßig die neue Verschlüsselungsmethode CCMP/AES und soll somit die Sicherheit weiter erhöhen. WPA2 kann genauso wie WPA mit einem PSK als auch mit einem (RADIUS-) Server eingesetzt werden. Da CCMP/AES sehr rechenintensiv ist reicht es oftmals nicht aus, wie noch von WEP zu WPA, ein Softwareupdate bei den Access Points einzuspielen, sondern meist muss neue Hardware angeschafft werden.

CCMP basiert auf AES, welches der Nachfolger des im Jahre 1976 entwickelten DES[21] ist, und wurde im Oktober 2000 zum Standard. AES ist ein Blockchiffre und benutzt eine Blockgröße von 128 Bit und Schlüsselgrößen von 128, 192 oder 256 Bit.

3.2.3 Verschlüsselung unter Ubuntu

Unter Ubuntu findet das Programm wpa_supplicant beim Verbindungsaufbau zu Access Points Verwendung. Es besteht aus einer Konfigurationsdatei, die die Endung .conf trägt. Diese Datei enthält für verschiedene Verschlüsselungsarten unterschiedliche Parameter, ist im Allgemeinen jedoch gleich aufgebaut. Abbildung 2 zeigt den allgemeinen Aufbau dieser Datei.

Gestartet wird das Programm wpa_supplicant mit dem Befehl

```
wpa_supplicant -i Interface -D wext -
c/tmp/netroam/wpa_supplicant.conf
```

Dabei muss der Parameter Interface durch den Namen des jeweiligen WLAN-Interfaces ersetzt werden. Der zweite Parameter „-D wext“ gibt den zu verwendenden Treiber an. In der letzten Angabe steht die Konfigurationsdatei mit Pfadangabe.

19 Counter Mode with Cipher Block Chaining Message Authentication Code Protocol
20 Advanced Encryption Standard
21 Data Encryption Standard

```
network={
ssid="SSID"
scan_ssid=1
proto=WPA
key_mgmt=WPA-PSK
pairwise=TKIP
group=TKIP
psk="KEY"
}
```

Abbildung 2: Aufbau der wpa_supplicant Konfigurationsdatei

Die hier abgebildete Datei ist für den Aufbau einer Verbindung, bei der WPA eingesetzt wird, vorgesehen. Dabei kann der Key mit Hilfe des Programms wpa_passphrase erzeugt werden, welches so aufgerufen wird

wpa_passphrase *SSID KEY*

und als Ausgabe einen fertigen Schlüssel in hexadezimaler Form erzeugt, wobei zu beachten ist, dass der Parameter KEY eine Länge von 8 bis 63 Zeichen besitzen muss.
Eine weitere Möglichkeit sich mit einem Access Point zu verbinden, geschieht mit dem Kommando *iwconfig*. Dies funktioniert jedoch nur bei WEP-Verschlüsselung, da dort keine besonderen Einstellungen vorgenommen werden müssen. Um so eine Verbindung aufzubauen, muss der Befehl

iwconfig *Interface* **essid** *SSID* **key** *KEY*

in ein Konsolenfenster eingegeben werden. Dabei müssen die Parameter Interface, SSID und KEY durch die jeweiligen richtigen Werte ersetzt werden.

3.3 Adressierung

Die im Internet und in lokalen Netzen benutzte Adressierung ist die IP-Adressierung. Jedes System, welches über das Internet ansprechbar sein soll, ist über eine IP-Adresse erreichbar. Eine IP-Adresse ist immer eindeutig, d.h. zum selben Zeitpunkt besitzt nur ein Gerät und kein weiteres diese Adresse. Das momentan noch verwendete Protokoll ist das IPv4[22]-Protokoll welches im September 1981 eingeführt wurde und im RFC 791 definiert ist. IPv4 besitzt eine Länge von 32 Bit und wird der besseren Übersicht halber in vier 8 Bit große Blöcke unterteilt. Jeder dieser Blöcke kann die Werte zwischen 0 und 255 annehmen.

22 Internet Protocol Version 4

Eine IP-Adresse, beispielsweise 209.85.129.104, ist in zwei Teile unterteilt. Der erste Teil ist der so genannte Netzteil (oftmals 24 Bit) und der zweite Teil ist der Adressteil (8 Bit). Sollte der Netzteil der IP-Adresse bei zwei Systemen gleich sein, so befinden sich die Systeme im gleichen Netz. Da der Netzteil nicht immer 24 Bit lang sein muss, kommt in jedem Netzwerk noch eine so genannte Netzmaske zum Einsatz. Die Netzmaske hat dieselbe Länge wie die IP-Adresse und unterteilt diese explizit in Netz- und Adressteil. Dabei sind alle Bits der Netzmaske, die den Netzteil kennzeichnen 1 und alle Bits, die den Adressteil kennzeichnen 0. Ein Beispiel für eine Netzmaske kann 255.255.255.128 sein, was dieselbe Schreibweise für 11111111.11111111.11111111.10000000 ist. Es bleiben in diesem Fall somit noch 128 Adressen übrig, wobei die kleinst mögliche Adresse immer die Netzadresse und die größt mögliche Adresse die Braodcastadresse ist. Somit können beide nicht als normale IP-Adressen an Clients vergeben werden. Mit Hilfe einer Broadcastadresse (z.B. 209.85.129.255), lassen sich alle Clients in diesem Subnetz erreichen und man erhält beispielsweise bei einem Ping auf diese Adresse eine Antwort von allen aktiven Clients.

<table>
<tr><td>Version
(4 Bit)</td><td>IHL
(4 Bit)</td><td>Type of Service
(8 Bit)</td><td colspan="2">Length
(16 Bit)</td></tr>
<tr><td colspan="3">Identification
(16 Bit)</td><td>Flags
(3 Bit)</td><td>Fragment Offset
(13 Bit)</td></tr>
<tr><td colspan="2">Time-to-live
(4 Bit)</td><td>Protocol
(4 Bit)</td><td colspan="2">Header Checksum
(16 Bit)</td></tr>
<tr><td colspan="5">Source Address
(32 Bit)</td></tr>
<tr><td colspan="5">Destination Address
(32 Bit)</td></tr>
<tr><td colspan="4">Options
(24 Bit)</td><td>Padding
(8 Bit)</td></tr>
</table>

Abbildung 3: IPv4-Header

Wie in Abbildung 3 zu sehen ist, beträgt die Größe des IPv4-Headers 20 Byte. Damit beträgt die maximale Größe eines IP-Paketes 65535 Byte was 64 kB entspricht.

Da immer mehr Geräte heutzutage eine IP-Adresse benötigen und die IP-Adressen bei IPv4 immer knapper werden, ist ein neues IP-Protokoll (IPv6[23]) eingeführt worden. IPv6 wurde von der IETF[24] im RFC[25] 1883 spezifiziert. Eine IPv6-Adresse hat die Länge von 128 Bit und wird in 8 Blöcke mit je 16 Bit aufgeteilt. Mit dieser Vergrößerung von 32 Bit auf 128 Bit

23 Internet Protocol Version 6
24 Internet Engineering Task Force
25 Requests for Comments

lassen sich 3,4 × 10^{38} Adressen erzeugen. IPv6-Adressen werden nicht wie IPv4-Adressen in dezimaler Form und durch Punkte, sondern in hexadezimaler Form und durch Doppelpunkte getrennt geschrieben. 0000:0000:9256:0a12:20ff:fe12:0028 ist beispielsweise solch eine Adresse. Um solche Adressen besser handhaben zu können, können sowohl führende Nullen weggelassen als auch Blöcke mit lauter Nullen durch **::** abgekürzt werden. Dabei ist jedoch zu beachten, dass Blöcke mit Nullen nur einmal in einer Adresse durch zwei Doppelpunkte abgekürzt werden dürfen. Das oben gezeigte Beispiel könnte somit auch so geschrieben werden: ::9256:a12:20ff:fe12:28. Die neuen Adressen werden genau wie die IPv4-Adressen in Netzteil und Adressteil unterteilt, welche jeweils eine Größe von 64 Bit besitzen. Eine weitere Veränderung von IPv4 zu IPv6 ist der Header, welcher nun eine feste Größe von 40 Bytes hat.

<table>
<tr><td>Version
(4 Bit)</td><td>Traffic Class
(8 Bit)</td><td colspan="3">Flow Label
(20 Bit)</td></tr>
<tr><td colspan="3">Payload Length
(16 Bit)</td><td>Next Header
(8 Bit)</td><td>Hop Limit
(8 Bit)</td></tr>
<tr><td colspan="5">Source Address
(128 Bit)</td></tr>
<tr><td colspan="5">Destination Address
(128 Bit)</td></tr>
</table>

Abbildung 4: IPv6-Header

Wie in Abbildung 4 zu erkennen ist, wurde der IPv6-Header gegenüber dem IPv4-Header wesentlich vereinfacht. Aus diesem Grund wurden bei diesem Protokoll Erweiterungsheader eingeführt. Diese Header stehen zwischen dem eigentlichen IPv6-Header und den Daten, und können Informationen über Fragmentierung, Authentisierung Verschlüsselung, Ziel und Optionen für das Routing enthalten.

3.3.1 statische Adressvergabe

Unter der statischen Adressvergabe versteht man das manuelle Setzen der IP-Adresse. Neben der IP-Adresse muss zusätzlich das Gateway eingetragen werden, welches meistens die IP-Adresse des momentan verbundenen Routers ist. Des Weiteren ist noch die Netzmaske, der DNS[26]-Server und die Broadcast-Adresse einzutragen.

Wie die Daten unter Ubuntu eingetragen werden, wird in Kapitel 3.3.3 beschrieben.

26 Domain Name System

3.3.2 dynamische Adressvergabe

Bei der dynamischen Adressvergabe übernimmt das DHCP[27]-Protokoll, welches im RFC 2131 und früher in RFC 1531 und RFC 1541 spezifiziert ist, das Setzen der im vorherigen Abschnitt genannten Parameter. Dabei erhält das Gerät, bei dem DHCP aktiviert sein muss, die erforderlichen Daten von einem DHCP-Server, der gleichzeitig auch das Gateway darstellen kann. DHCP ist immer dann hilfreich, wenn man ein Gerät oft in verschiedenen Netzwerken betreibt oder wenn sich in einem vorhandenen Netzwerk die Geräte stetig ändern.

Die Kommunikation zwischen Client und DHCP-Server läuft immer gleich ab. Sobald ein Gerät (Client) in ein bestehendes Netzwerk eingebracht wird, sendet der Client eine DHCPDISCOVER-Nachricht mit seiner eigenen MAC-Adresse per Netzwerkbroadcast an die 255.255.255.255. Da der Client zu diesem Zeitpunkt noch keine IP-Adresse besitzt, ist die Sende-IP-Adresse die 0.0.0.0. Gibt es einen oder mehrere DHCP-Server in dem Netz, so senden diese ein DHCPOFFER mit einer freien IP-Adresse ebenfalls an die Adresse 255.255.255.255. Kommen mehrere dieser DHCPOFFER-Nachrichten beim Client an, so muss dieser sich für eine entscheiden und sendet nochmals per Netzwerkbroadcast, diesmal aber mit den ihm bekannten Informationen des ausgewählten DHCP-Servers, ein DHCPREQUEST. Dieser antwortet mit einem DHCPACK und verschickt gleichzeitig die noch benötigten Daten wie Gateway, DNS usw. an den Client.

Da IP-Adressen nicht unbegrenzt haltbar sind, und nach einiger Zeit verfallen (Leasetime), muss der Client in regelmäßigen Abständen (nach der Hälfte der Leasetime) DHCPREQUEST-Nachrichten an den Server schicken, um zu erreichen, dass er weiterhin eine IP-Adresse besitzt. Die Antwort des Servers ist dann zumeist ein DHCPACK mit denselben Daten wie zuvor. Bekommt der Client vom Server keine Antwort, so muss der Client zu einem späteren Zeitpunkt (nach 7/8 der Leasetime) eine weitere DHCPREQUEST-Nachricht ins Netz schicken, in der Hoffnung, von einem anderen Server eine IP-Adresse zu erhalten. In der nachfolgenden Tabelle sind alle möglichen Nachrichten aufgeführt, die zwischen Client und Server gesendet werden können.

27 Dynamic Host Configuration Protocol

Nachricht	Beschreibung
DHCPDISCOVER	Broadcast eines Clients der einen Server sucht.
DHCPOFFER	Antwort der Server mit einem Parameter-Vorschlag.
DHCPREQUEST	Broadcast des Clients an den bevorzugten Server. Implizite Ablehnung aller anderen Server.
DHCPACK	Der Server liefert eine IP-Adresse.
DHCPNACK	Der Server lehnt die Anfrage des Clients ab.
DHCPDECLINE	Der Client hat ein Problem mit der angebotenen IP-Adresse und lehnt ab.
DHCPRELEASE	Der Client gibt die IP-Adresse vor Ablauf der Lease zurück.
DHCPINFORM	Anfrage des Clients nach Daten ohne IP-Adresse.

Tabelle 3: Mögliche DHCP-Nachrichten[6]

Da in den DHCP-Nachrichten relativ viele Daten mitgesendet werden, soll nun kurz der Aufbau des DHCP-Pakets dargestellt werden.

OP (8 Bit)	Hardware Type (8 Bit)	Hardware Address Length (8 Bit)	Hops (8 Bit)
Transaction ID (32 Bit)			
Seconds (16 Bit)		Flags (16 Bit)	
Client IP-Address (32 Bit)			
Your IP-Address (32 Bit)			
Server IP-Address (32 Bit)			
Relay-Agent-IP-Address (32 Bit)			
Client Mac Address (32 Bit)			
Server Hostname (32 Bit)			
Options			

Abbildung 5: DHCP-Paket

3.3.3 Interfacekonfiguration

Unter Ubuntu ist es möglich das Interface bei statischer Adressvergabe auf zwei Arten zu konfigurieren. Die erste Möglichkeit ist nur temporär und die Einstellungen sind nach einem Neustart des Rechners nicht mehr vorhanden. Bei dieser Methode wird in der Konsole der Befehl

ifconfig *Interface IP-Adresse* **netmask** *Netzmaske* **broadcast** Broadcastadresse

eingetragen. Dabei sind die kursiv geschriebenen Parameter Interface, IP-Adresse, Netzmaske und Broadcastadresse durch die jeweiligen Daten zu ersetzen.

Die zweite Möglichkeit der Interfacekonfiguration besteht darin, die Datei */etc/network/interfaces* anzupassen. Ein Ausschnitt aus dieser Datei soll verdeutlichen wie die Daten eingetragen werden müssen.

```
// etc/network/interfaces

auto eth0
iface eth0 inet static
address 139.6.19.200
netmask 255.255.255.224
gateway 139.6.19.192
```

Die erste Zeile gibt an, dass das jeweilige Interface beim Starten des Rechners mit gestartet werden soll. Wird dies nicht gewünscht, so kann diese Zeile einfach mit einer Raute (#) auskommentiert werden. In der zweiten Zeile, *iface <Interface> inet static*, wird festgelegt, dass das Interface statisch, also manuell konfiguriert wird. Die letzten drei Zeilen geben der Reihe nach die IP-Adresse, die Netzmaske und das Gateway an.

Soll das Interface nicht manuell, sondern per DHCP, konfiguriert werden, so ist es auch möglich, dies in der Datei einzutragen. Dazu muss nur die zweite Zeile durch die Zeile *iface <Interface> inet dhcp* ersetzt werden. In diesem Fall brauchen keine weiteren Einträge für Adresse, Netzmaske und Gateway eingetragen werden.

```
// etc/network/interfaces

auto eth0
iface eth0 inet dhcp
```

Wie die erste Zeile (auto <Interface>) in den beiden Ausschnitten zeigt, ist diese Art der Interfacekonfiguration nicht nur temporär, sondern übersteht auch einen Neustart des Rechners. Änderungen an dieser Datei werden im Allgemeinen nur nach einem Neustart wirksam. Da dies aber bei vielen Änderungen recht aufwändig wäre, gibt es noch zwei andere Möglichkeiten die Änderungen wirksam zu machen. Zum einen können die kompletten Interfaces des Rechners mit dem Befehl */etc/init.d/networking restart* in einer Konsole neu gestartet werden, zum anderen kann mittels *ifdown <Interface>* gefolgt von *ifup <Interface>* nur ein einzelnes Interface neu gestartet werden und so die Änderungen auch nur für dieses eine Interface wirksam machen. Sollte ein Rechner mehrere Interfaces besitzen, die auch alle aktiv in einem Netzwerk sind, ist die zweite Möglichkeit der ersten Variante vorzuziehen.

4 Routing

Da in einem Datenpaket, welches über ein Netzwerk verschickt werden soll, immer nur die Anfangsadresse und die Zieladresse, nie aber der Weg zur Zieladresse bekannt ist, ist das Routing ein wichtiger Vorgang in der Netzwerktechnik. Beim Routing wird anhand von Tabellen Entscheidungen getroffen, über welchen Weg ein Datenpaket zu laufen hat, um auf kürzestem Wege beim Empfänger anzukommen.

4.1 Die Routingtabelle

Eine Tabelle, eine so genannte Routingtabelle, mittels derer Ubuntu entscheidet über welches Interface die Datenpakete den Rechner verlassen sollen, kann beispielsweise wie in Abbildung 6 aussehen. Um solch eine Routing-Tabelle angezeigt zu bekommen, muss der Befehl *route* in ein Konsolen-Fenster eingegeben werden. Die Routingtabelle stammt von einem PC, der mit Hilfe von zwei Routern, die gleichzeitig als Gateway fungieren, mit den Netzen 120.3.1.192 per LAN und 192.168.1.0 per WLAN verbunden ist.

Kernel IP Routentabelle

Ziel	Router	Genmask	Flags	Metric	Ref	Use	Iface
120.3.1.192	*	255.255.255.192	U	0	0	0	eth0
192.168.1.0	*	255.255.255.0	U	0	0	0	eth1
default	192.168.1.1	0.0.0.0	UG	0	0	0	eth1
default	120.3.1.193	0.0.0.0	UG	0	0	0	eth0

Abbildung 6: Routingtabelle

In der hier abgebildeten Routingtabelle sind Routen für die beiden Interfaces eth0 und eth1 festgelegt. Dabei ist die Tabelle in zwei Bereiche aufgeteilt. Der erste Bereich enthält die ersten beiden Zeilen der Einträge, der zweite Bereich die letzten beiden Zeilen. Der erste Bereich beschreibt spezielle Netze in die geroutet werden soll an Hand der Netzadresse, welche in der ersten Spalte stehen, wobei diese Einträge numerisch sortiert sind. In diesem Beispiel ist das Netz mit der Netzadresse 120.3.1.192 über das Interface eth0 und das Netz mit der Netzadresse 192.168.1.0 über das Interface eth1 erreichbar. Die zweite Spalte ist für den Router, bzw. für das Gateway vorgesehen, über welchen man das Netz erreicht (next hop). Der Stern (*) in diesem Beispiel beschreibt, dass kein Gateway benötigt wird um in dieses Netz zu gelangen, da man direkt mit dem Netz verbunden ist. Für den zweiten Bereich werden in der ersten Spalte keine speziellen Adressen verwendet. Der Eintrag *default* beschreibt hier, dass alle Pakete, zu dessen

Zieladresse kein anderer Eintrag aus der Routingtabelle passt, über das Gateway aus der zweiten Spalte geroutet werden. Gibt es mehrere default-Einträge, so wird immer der Eintrag genommen, der als oberstes dieser default-Einträge steht.

In der dritten Spalte steht die Netzmaske der Netze, in die geroutet werden soll. Bei default-Einträgen hat die Netzmaske den Wert 0.0.0.0, da keine spezielle Maske existiert. In der nächsten Spalte stehen die so genannten Flags. Die möglichen Werte und deren Bedeutung die in dieser Spalte stehen können, sind in nachfolgender Tabelle aufgeführt.

U	Route ist verfügbar
H	Ziel ist ein einzelner Rechner
G	Ein Gateway wird benutzt
R	Eine Route wird bei dynamischem Routing modifiziert
D	Dynamisch von einem Daemon oder weitergeleiteten Paket erzeugt worden
M	Von einem Daemon oder weitergeleiteten Paket geändert worden
A	Von addrconf erstellt worden
C	Eintrag aus dem Speicher
!	Ausgemusterte Route

Tabelle 4: Flags [7]

In der fünften Spalte steht die Metrik. Die Metrik ist ein Maß für die Qualität der Verbindung zwischen dem Sender bzw. dem Router und dem Ziel des Paketes. Mit ihrer Hilfe entscheidet der Router bzw. der Sender an welchen Next Hop im Netz er das Paket weitergibt. Routing-Metriken beziehen sich nicht ausschließlich auf die Länge des Weges zwischen Sender und Empfänger, sondern können auch andere Merkmale, wie beispielsweise die Qualität der Leitungen, die Bandbreite oder die Auslastung in die Entscheidung mit einbeziehen. [8]

Wie oben schon beschrieben, wird bei mehreren default-Einträgen immer der erste Eintrag verwendet. Dabei sind die default-Einträge nach der Größe der Metrik sortiert. Eine default-Route mit der Metrik 0 steht somit immer vor/über einer default-Route mit der Metrik 100.

Die sechste Spalte beschreibt die Anzahl der Referenzen auf die einzelne Route, allerdings wird dies von dem Linux-Kernel nicht unterstützt. Die vorletzte Spalte mit der Bezeichnung „Use" enthält statistische Werte zu den einzelnen Routen. Die letzte Spalte gibt, wie schon beschrieben, das Interface an, über das die Pakete weitergeleitet werden, wenn der zugehörige Routingeintrag benutzt wird.

4.2 Routing vs. Forwarding

In Routern muss zwischen Routingtabellen und Forwardingtabellen unterschieden werden. Beim Routing entstehen Routingtabellen durch die Kommunikation mittels Routing-Protokollen (siehe Kapitel 4.4), bei denen Informationen zwischen den einzelnen Routern ausgetauscht werden. Diese Informationen variieren je nach Protokoll und enthalten zum Beispiel Informationen über Nachbarrouter. Sobald mehrere Routing-Protokolle auf einem Router laufen und diese alle eine separate Routingtabelle anlegen, wird der Aufwand der betrieben werden muss, sobald ein Datenpaket eintrifft, immens groß. Theoretisch müsste jede Routingtabelle nach einer passenden Route durchsucht werden und mit den anderen gefundenen Routen verglichen werden. Um dieses Problem zu beheben, werden so genannte Forwardingtabellen vom Router selbst erstellt. Forwardingtabellen sind im Prinzip die Haupt-Routingtabellen in Routern. Ein Router entscheidet anhand der Zieladresse im Datenpaket und dieser Forwardingtabelle, über welchen Weg das eintreffende Datenpaket den Router wieder verlässt.

Auch manuelle Eingriffe in die Routingtabellen sind durch einen Administrator möglich und haben beim Erstellen der Forwardingtabelle immer einen höheren Stellenwert (administrativen Distanz) als die vom Router durch die Routingprotokolle selbst erstellten Einträge. Sollte es beispielsweise beim Erstellen der Forwardingtabelle zu einem Widerspruch zwischen einer von einem Administrator erstellten Route und einer Route kommen, die der Router selbst erstellt hat, so würde die vom Administrator erstellte Route bevorzugt.

4.3 Routenmanipulation unter Ubuntu

Die oben dargestellte Routingtabelle kann sowohl von Ubuntu selbst als auch vom Benutzer mit entsprechenden Rechten manipuliert werden. Um beispielsweise eine „default-Route" manuell hinzuzufügen, muss in der Konsole folgendes Kommando eingegeben werden:

```
route add default dev Interface  [metric Metrik] gw Gateway
```

Die jeweiligen kursiv geschriebenen Parameter sind durch die entsprechenden Werte zu ersetzen. Der Parameter Metrik ist optional und sollte er nicht gesetzt werden, so setzt Ubuntu die Metrik selbstständig. Das Löschen der „default-Route" verläuft analog, lediglich der Befehl *add* muss durch den Befehl *del* ersetzt werden. Ebenfalls ist es möglich, nur den

Befehl *route del default* zu verwenden. Wird dieser eingegeben, so löscht er ebenfalls den default-Eintrag, wobei bei mehreren default-Einträgen immer der erste Eintrag gelöscht wird.

Auch besteht die Möglichkeit, Routen zu einzelnen Netzen hinzuzufügen. Dabei muss die folgende Syntax beachtet werden:

route add -net *IP-Adresse* **netmask** *Netzmaske* **dev** *Interface*

Beim Parameter *IP-Adresse* ist darauf zu achten, das nicht eine bestimmte IP-Adresse eingetragen wird, sondern das Netz selbst. Soll also beispielsweise in das Netz in der sich die IP-Adresse 139.6.19.10 befindet geroutet werden, so muss für den Parameter *IP-Adresse* die Netzadresse eingegeben werden, welche beispielsweise die 139.6.19.0 sein könnte. Das Löschen dieser Route gestaltet sich ebenso einfach wie bei der „default-Route", es muss auch hier lediglich der Befehl *add* durch den Befehl *del* ersetzt werden.

4.4 Routing-Protokolle

Routing-Protokolle sind, wie schon in Kapitel 4.2 beschrieben, für den Austausch von Informationen zwischen den einzelnen Routern zuständig. Mit Hilfe dieser Protokolle lassen sich dynamische Routing-Tabellen aufbauen. Dank dieser Dynamik sind Routing-Tabellen sehr flexibel und gut gegen Routerausfälle geschützt, da innerhalb von Sekunden eine neue Routing-Tabelle aufgebaut werden kann. Routing-Protokolle basieren dabei auf verschiedenen Algorithmen und können in Interior Gateway Protokolle (IGP) und Exterior Gateway Protokolle (EGP) aufgeteilt werden. Zu den IGPs gehören unter anderem das IGRP/EIGRP[28], das OSPF[29], das IS-IS[30] sowie das RIP[31] Protokoll. Zu den EGPs gehören BGP[32] und das EGP selbst.

28 Interior Gateway Routing Protocol/ Enhanced IGRP
29 Open Shortest Path First
30 Intermediate System to Intermediate System
31 Routing Information Protocol
32 Border Gateway Protocol

5 Entwicklung

Dieses Kapitel beschäftigt sich mit der Entwicklung des Programms NetRoam. Dazu wird zuerst auf den Lösungsansatz eingegangen, bevor das Programm im Einzelnen beschrieben wird. Zum besseren Überblick sind in der Beschreibung des Codes Programmablaufpläne eingefügt. Dafür werden Kenntnisse in der Shell-Programmierung vorausgesetzt.

5.1 Lösungsansatz

Um das Ziel dieser Bachelorarbeit zu erreichen, wurde auf die Shell-Programmierung zurückgegriffen. Mit Hilfe der Shell-Programmierung ist es möglich, alle für diese Arbeit nötigen Systemeingriffe und Systemabfragen durchzuführen. Anfänglich war geplant, der Übersicht halber mehrere Scripte zu entwickeln. Die Kommunikation zwischen diesen Scripten müsste dabei über Dateien laufen, da verschiedene Scripte aus Sicherheitsgründen nicht direkt miteinander kommunizieren können. Da dabei aber eine große Anzahl an Dateien entstanden wäre, in denen oftmals nicht mehr als eine einzelne Zeile gestanden hätte, wurde dieses Konzept verworfen und die einzelnen Scripte wurden in ein großes Script zusammengefügt. Dabei wurden die Inhalte der einzelnen Scripte in Funktionen geschrieben, sodass die Struktur weiterhin gewahrt bleibt. Die Kommunikation zwischen den einzelnen Funktionen kann nun komfortabel mittels Variablen realisiert werden. Da die Shell-Programmierung keine mehrdimensionalen Arrays vorsieht, konnte an bestimmten Stellen jedoch trotzdem nicht auf das Abspeichern von Werten in Dateien verzichtet werden.

Aufgrund der großen Anzahl an Ein- und Ausgaben des entwickelten Programms, wurde ein zusätzliches Webfrontend erstellt. Die Schnittstellen bilden hierbei wieder Dateien. Mit dem Webfrontend ist es möglich, beispielsweise Access Points oder Router hinzuzufügen oder zu löschen. Aber auch Ausgaben des Programms können mit diesem Webfrontend komfortabel betrachtet werden. Auf den Aufbau und die Erklärung des Webfrontend wird in Kapitel 8.2 eingegangen.

Das Problem, das während der Entwicklung aufgetreten ist, war die Steuerung der Routingtabelle. Da Kubuntu die Steuerung bei Netzwerkwechsel selbstständig übernimmt und man bei IP-Zuweisungen per DHCP keine Eingriffsmöglichkeit in die Routingtabelle hat, musste das Script (dhclient-script) welches Kubuntu dabei benutzt, angepasst werden. Die

notwendigen Modifikationen können in Kapitel 5.3 nachgelesen werden. Nach dieser Anpassung hat nun NetRoam die volle Kontrolle über die Routingtabelle und kann diese, wenn nötig, manipulieren.

5.2 Implementierung NetRoam

Das Programm NetRoam besteht, wie im vorherigen Kapitel beschrieben, aus zahlreichen Funktionen. Das Hauptprogramm beschränkt sich dagegen nur auf einige Zeilen. Ein Programmablaufplan des Hauptprogramms ist in nachfolgender Abbildung gezeigt. Dabei ist dieser und die nachfolgenden Ablaufpläne relativ grob gehalten und dienen lediglich der Übersicht. Einen genaueren Einblick in das Programm bieten die Codeausschnitte mit den dazugehörigen Erklärungen.

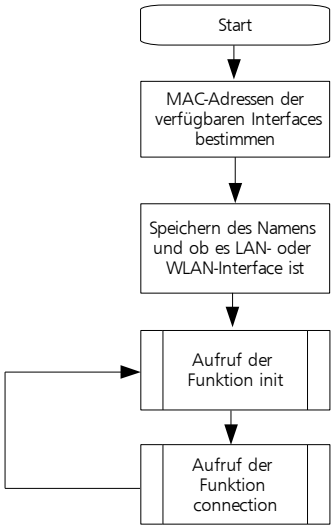

Abbildung 7: Ablaufplan des Hauptprogramms

Wie in Abbildung 7 zu sehen ist, besteht das Hauptprogramm aus drei Teilen. Im ersten Teil werden die Mac-Adressen der LAN- und WLAN-Interfaces ermittelt.

```
#Speichern der Mac Adressen in das Array mac
help=`ifconfig -a 2>/dev/null |grep Hardware| grep -v avah | awk '{print $5}'`
mac=(${help// /'" "'})
```

Ubuntu bringt in der Version 7.04 das Programm Avahi mit, welches zum einfachen Vernetzen von Geräten über das Netzwerk gedacht ist. Avahi taucht auch in den Interfaces unter *ifconfig* auf (beispielsweise mit eth0:avah). Daher musste sichergestellt werden, dass

diese Interfaces nicht berücksichtigt werden. Dies geschieht, wie im Programmausschnitt zu sehen ist, beim Aufrufen der Interfaces mit Hilfe des Befehls *ifconfig,* mit dem Befehl *grep -v avah.* Der zweite Teil des Hauptprogramms beschäftigt sich mit dem Speichern der Namen der Interfaces, zu denen im vorherigen Abschnitt die MAC-Adressen gespeichert wurden. Außerdem wird in diesem Teil gespeichert, ob es sich bei dem jeweiligen Interface um ein LAN- oder WLAN -Interface handelt. Dies ist wichtig, damit die Interfaces zwischen LAN- und WLAN-Interfaces unterschieden werden können.

```
#Speichern der Namen der Interfaces in das array interface
n=0
for i in ${mac[*]}
do
interface[n]=`ifconfig -a 2>/dev/null|grep $i|grep -v avah|awk'{print $1}'`

      #Speichern in das Array name, ob es sich um ein WLAN oder LAN Interface handelt
      var=`iwconfig ${interface[n]} 2>/dev/null | grep ESSID`
      if test "$var" == ""
      then
             name[$n]="lan"
      else
             name[$n]="wlan"
      fi
      n=`expr $n + 1`
 done
```

Da die beiden ersten Teile nur zu Beginn des Programms ausgeführt werden, ist zu beachten, das auch nur die beim Start vorhandenen Interfaces berücksichtigt werden. Wird im laufenden Betrieb ein weiteres Interface eingebaut, beispielsweise eine WLAN-Karte über den PCMCIA-Steckplatz, so wird diese erst nach einem Neustart des Programms erkannt.

Den dritten Teil des Hauptprogramms bilden die beiden letzten Blöcke aus dem Programmablaufplan. Diese symbolisieren den Aufruf der beiden Funktionen *init* und *connection*. Im Programm werden diese beiden Funktionen in einer Endlosschleife aufgerufen. Mit einem *sleep*-Befehl in der Endlosschleife soll verhindert werden, dass die Funktionen zu schnell hintereinander aufgerufen werden.

Die erste Funktion ist die Funktion *init*, deren Hauptaufgabe darin besteht zu prüfen, ob die gefundenen Interfaces aktiv sind. Sind diese aktiv, so wird, wie in Abbildung 8 zu sehen ist, für jede Art von Interface eine andere Funktion aufgerufen. Für ein aktives WLAN-Interface die Funktion *wlan* und für ein aktives LAN-Interface die Funktion *lan*. Sind die Interfaces wiederum nicht aktiv, bzw. ist kein LAN-Kabel eingesteckt, so wird dies in

der Datei *<Interface>.txt* mit dem Eintragen einer 0, bzw. bei LAN-Interfaces mit dem Leer lassen der Datei, vermerkt. Dabei wird der Parameter Interface im Programm durch den jeweiligen Namen des Interface ersetzt und dient hier lediglich der Übersicht. Da NetRoam im Laufe des Programms noch weitere Zahlen in die Datei *<Interface>.txt* schreibt, sind in den Tabellen 5 und 6 die möglichen Zahlen und ihre Bedeutung beschrieben.

"0"	WLAN-Karte nicht aktiv, oder kein bekannter Access Point gefunden
"1"	WLAN-Karte ist mit einem Access Point verbunden
"2"	WLAN-Karte hat Umgebung gescannt und bekannte Access Points gefunden

Tabelle 5: Mögliche Zahlenkürzel bei WLAN-Interfaces

" "	LAN-Kabel ist nicht eingesteckt
"0"	Kein bekannter Router wurde gefunden und DHCP-Anfrage ist erfolglos geblieben
"1"	Interface besitzt eine statische IP
"2"	IP wurde über DHCP zugewiesen

Tabelle 6: Mögliche Zahlenkürzel bei LAN-Interfaces

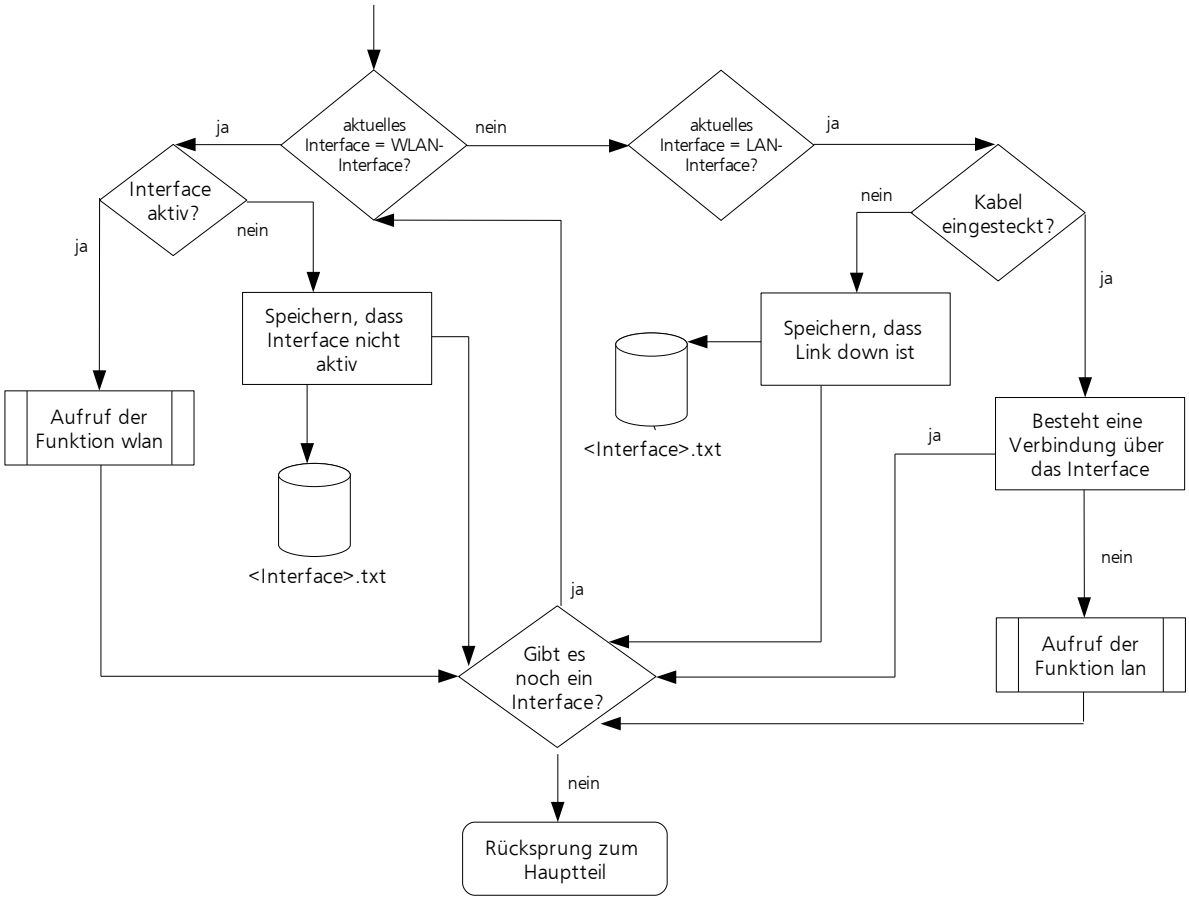

Abbildung 8: Ablaufplan der Funktion init

Die nachfolgenden Aufgaben der Funktion *init* sind nicht im Programmablaufplan in Abbildung 8 enthalten, da sonst die Übersicht über den eigentlich Ablauf verloren gegangen wäre.

Die erste zusätzliche Aufgabe der Funktion besteht darin zu prüfen, ob schon bekannte WLANs in der Datei *wlan.csv* hinterlegt wurden. (Auf die Schnittstellen für die Kommunikation zwischen dem Webfrontend, wo die Access Points und Router eingetragen werden können, und dem Programm, wird in Kapitel 5.4 eingegangen.) Ist dies nicht der Fall, so wird der Benutzer mittels eines Pop-Ups, sofern es auf aktiv steht, unter Verwendung von Zenity darauf hingewiesen. Dies ist wichtig, da nur dann ein bekanntes WLAN gefunden werden kann, wenn zuvor Access Points hinterlegt wurden.

```
leer=""
until(test "$leer" != "")
do
  #Prüfen ob Datei leer ist
  leer=`cat /etc/netroam/wlan.csv`
  if test "$leer" != ""
  then
```

```
        wlan #Aufruf der Funktion wlan
    else
        #Nur wenn in Datei popupconfig.txt Popup auf aktiv steht
        popup=`cat /etc/netroam/config.txt | grep "leer" | awk -F: '{print $2}'`
        if test "$popup" = "1"
        then
                zenity --question --title="NetRoam" --text="Bitte tragen Sie einen
                AccessPoint ein und bestätigen Sie anschließend mit OK, oder
                drücken Sie auf CANCEL um eine Lan-Verbindung aufzubauen (Bitte
                achten Sie darauf, dass ein Lan Kabel eingesteckt ist)"
                if [ $? != 0 ]
                then
                        leer="1"
                        echo "0" > /tmp/netroam/${interface[m]}.txt
                        echo -n "0" > /etc/netroam/prio.txt
                fi
                sleep 5 #Benutzer Zeit geben Daten einzutragen
        fi
    fi
done
```

Die nächste Aufgabe liegt darin zu bemerken, ob während einer aktiven Verbindung über das LAN-Interface das LAN-Kabel gezogen wird. Ist dies der Fall, so reagiert NetRoam ebenfalls und setzt die Priorität auf WLAN um, was zur Folge hat, dass versucht wird eine WLAN-Verbindung aufzubauen. Dafür wird einfach in die Datei *prio.txt* eine 1 eingetragen. Auch ist es die Aufgabe der Funktion zu bemerken, wenn ein LAN-Kabel wieder eingesteckt wurde. Wünscht der Benutzer nach dem Einstecken des Kabels die Priorität zu ändern und damit die Verbindung auf das LAN zu setzen, so wird eine 0 in die Datei *prio.txt* geschrieben.

Die Funktion *wlan*, die aufgerufen wird, wenn es sich bei dem aktuellen Interface um ein WLAN-Interface handelt, hat die Aufgabe zu prüfen, ob über das momentane Interface eine Verbindung zu einem Access Point besteht. Dazu wird geprüft, ob das Interface eine IP-Adresse und somit auch eine aktive Verbindung besitzt.

```
#Überprüfen ob eine Verbindung zu einem Access Point besteht
con_ap_ip=`ifconfig ${interface[m]} | grep "inet Adresse"`
```

Ist dies der Fall, so wird der alte Inhalt der Datei *<Interface>.txt* gelöscht und die Zahl 1, für Access Point verbunden (Tabelle 5), die MAC-Adresse und die momentane Signalstärke des verbundenen Access Points in die Datei geschrieben.

```
#Speichern der Mac-Adresse des AP und der Signalqualität
echo "1" > /tmp/netroam/${interface[m]}.txt
echo -n "$con_ap_mac" >> /tmp/netroam/${interface[m]}.txt
con_ap_signal=`iwconfig ${interface[m]}| grep "Signal level" | awk '{print $4}'| cut -c7-10`
echo ",$con_ap_signal" >> /tmp/netroam/${interface[m]}.txt
```

Die Datei *<Interface>.txt* könnte damit beispielsweise so aussehen:

Abbildung 9: Aufbau der Datei <Interface>.txt für ein WLAN-Interface bei bestehender Verbindung

Außerdem wird in der Datei *infowlan<Interface>.txt* die Signalstärke aktualisiert. Diese Datei dient der Kommunikation zwischen dem Programm NetRoam und dem Webfrontend. Die Datei *infowlan<Interface>.txt* wird in Kapitel 5.4 näher beschrieben.
Besitzt das Interface allerdings keine IP-Adresse, so wird die Umgebung nach verfügbaren Access Points gescannt.

```
ifconfig ${interface[m]} 0.0.0.0
ifconfig ${interface[m]} 1.1.1.1
help=`iwlist ${interface[m]} scanning`
```

Dabei ist es wichtig, dass der WLAN-Karte zuvor eine Pseudo-IP-Adresse zugewiesen wird, damit die Karte nicht „down" ist und mit dieser überhaupt gescannt werden kann. Zur Sicherheit wird der Karte zuvor noch die IP-Adresse 0.0.0.0 zugewiesen, damit sie alle vorherigen Einstellungen verliert. Des Weiteren werden die Daten wie SSID, Signalstärke und MAC-Adressen der gefundenen Access Points in verschiedenen Arrays gespeichert.
Damit die bekannten Access Points herausgefiltert werden können, müssen diese nach dem Scannen mit den hinterlegten Access Points verglichen werden. Dies geschieht mittels zweier for-Schleifen, in denen auch geprüft wird, ob ein gefundener Access Point auf der Blacklist steht. Ist dies der Fall, so wird er nicht gespeichert. Auf die Blacklist wird ebenfalls in Kapitel 5.4 ausführlicher eingegangen.
Da die gefundenen WLANs nicht nach Signalstärke sortiert, sondern beim Scannen immer unterschiedlich gefunden werden, ist es für die nachfolgenden Funktionen, die mit der Datei arbeiten in der die gefundenen Access Points gespeichert werden, wichtig, dass die Access Points der Signalstärke nach gespeichert werden.

```
#Sortieren der gefundenen AP nach SignalstÃ¤rke
if [ -e /tmp/netroam/h${interface[m]}.txt ]
then
    cat /tmp/netroam/h${interface[m]}.txt | sort -n -r  > /tmp/netroam/${interface[m]}.txt
    rm /tmp/netroam/h${interface[m]}.txt
fi
```

Dies geschieht mit dem Befehl sort, der als Eingabe die Datei mit den zu sortierenden Access Points erhält und die sortierte Liste in die Datei *<Interface>.txt* schreibt. Die fertige Datei, in der nun alle bekannten Access Points stehen, die zum momentanen Zeitpunkt verfügbar sind, könnte somit so aussehen:

```
2
-36,SSID,dhcp,00:45:C4:11:B1:A3,wpa,KEY
-45,SSID,static,00:C4:88:73:74:10,wpa2,KEY,192.168.0.100,255.255.255.0,192.168.1.255,192.168.0.1,192.168.0.1
...
```

Abbildung 10: Aufbau der Datei <Interface>.txt für ein WLAN-Interface bei keiner aktiven Verbindung

Wie in diesem Beispiel zu sehen ist, werden neben der Signalstärke und der MAC-Adresse noch weitere relevante Daten in dieser Datei gespeichert, welche aus der Datei *wlan.csv* stammen.

In Abbildung 11 ist noch einmal der Ablauf der Funktion in einem Programmablaufplan dargestellt.

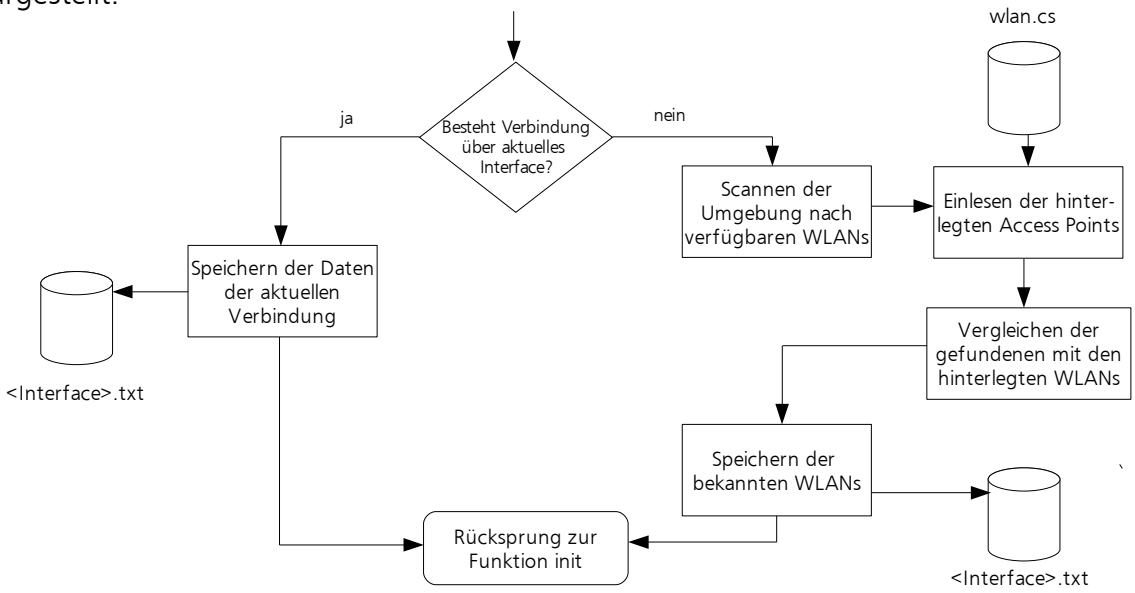

Abbildung 11: Ablaufplan der Funktion wlan

Bei der nächsten Funktion, der Funktion *lan*, die für die Ortsbestimmung zuständig ist, trat bei der Entwicklung ein unvorhergesehenes Problem auf. Da das Programm beim Einstecken eines LAN-Kabels nicht wissen kann, wo der Rechner sich befindet, das heißt somit auch nicht in welchem Netz er sich befindet, und welche Einstellungen gemacht werden müssen, musste ein Weg gefunden werden dies herauszufinden.
Dazu wurde zunächst versucht mittels eines Broadcast-Pings und der sich daraus aufbauenden ARP[33]-Liste, die MAC-Adresse des Routers bzw. des Gateways herauszufinden. Anschließend wird geprüft, ob für diesen Router Daten hinterlegt sind, um diese gegebenenfalls einzustellen. Da das Interface zu diesem Zeitpunkt allerdings noch keine IP-Adresse besitzt, und somit kein Ping abgesetzt werden kann, führte dieser Weg nicht zum Ziel. Auch das Zuweisen einer Pseudo-IP-Adresse brachte keinen Erfolg, womit dieser Lösungsansatz verworfen werden musste. Nach diesem Versuch war klar, dass jegliche Art von Ping, ohne zuvor korrekte eingestellte Daten am Interface, fehlschlägt. Es blieb nun nichts anderes mehr übrig, als der Reihe nach alle in der Datei *lan.csv* hinterlegten Daten auszuprobieren, um so das momentane Netz herauszufinden. Weil heutzutage allerdings in vielen Fällen das DHCP-Protokoll zur Adressierung eingesetzt wird, wird in der Funktion zuvor noch versucht via DHCP eine IP-Adresse zu erhalten. Erst wenn auch dieser Versuch fehlschlägt, wird auf die hinterlegten Daten zurückgegriffen. Die nachfolgende Abbildung zeigt den Ablauf der Funktion *lan*.

33 ARP (Address Resolution Protocol) ist für das Auflösen von MAC Adressen zuständig

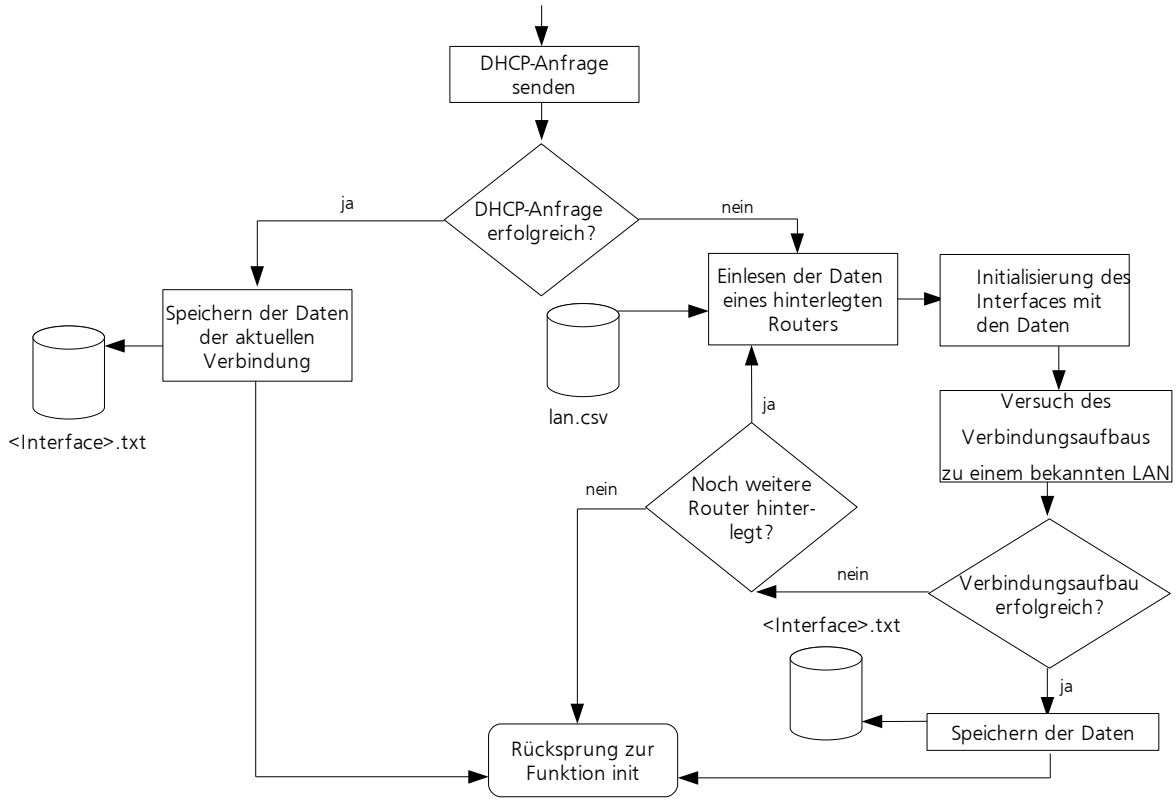

Abbildung 12: Ablaufplan Funktion lan

Ist die DHCP-Anfrage erfolgreich, werden die Daten wie Gateway, Netmask, IP-Adresse oder auch der DNS-Server abgespeichert. Ist dies nicht der Fall, so werden, wie oben schon beschrieben, die hinterlegten Router der Reihe nach durchprobiert. Dazu wird dem Interface zunächst die in der Datei *lan.csv* hinterlegte IP-Adresse, die Netzmaske und die Broadcastadresse zugewiesen, ehe eine Route für das Netzwerk gesetzt wird.

```
#Setzen der IP-Konfiguration
ifconfig ${interface[m]} $lip netmask $lnetmask broadcast $lbroadcast
#Setzen der Route
route add default dev ${interface[m]} metric 100 gw $lgateway 2>/dev/null
```

Es ist wichtig, dass hier die default-Route mit einer hohen Metrik versehen wird, da die neue Route sonst als wirkliche default-Route benutzt wird, weil sie als erstes in der Routing Tabelle stehen würde, und somit alle Daten über diese Route verschickt werden würden. Damit dies nicht der Fall ist, wird hier eine Metrik der Größe 100 gewählt. Anschließend wird versucht, das hinterlegte Gateway anzupingen. Funktioniert der Ping nicht bzw. gehen die Pakete verloren, so wird der nächste hinterlegte Router mit den Daten ausprobiert. Wird ein Router gefunden, so werden die Daten genau wie bei einer erfolgreichen DHCP-Anfrage in der Datei *<Interface>.txt* gespeichert und die default-Route

wieder gelöscht. Ist auch nach den letzten hinterlegten Daten das momentane Netz nicht identifiziert, so wird eine 0 in die Datei eingetragen. Die Datei *<Interface>.txt* kann bei Erfolg so aussehen:

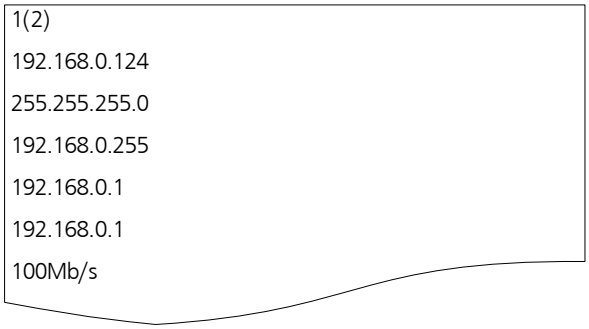

Abbildung 13: Aufbau der Datei <Interface>.txt für ein LAN-Interface bei aktiver Verbindung

Dabei bedeuten die einzelnen Zeilen der Reihe nach: 1 oder 2 steht für statisch oder DHCP, in der zweiten Zeile steht die IP-Adresse, in der dritten die Netzmaske, dann die Broadcastadresse, gefolgt vom Gateway, dem DNS-Server und der aktuellen Geschwindigkeit. Jedoch trat bei dem Gateway und dem DNS-Server ein weiteres Problem auf, weil das Gateway und der DNS-Server nicht so einfach ermittelt werden können. Da jedoch schon, wie im Lösungsansatz beschrieben, das Setzen der Route und auch des DNS-Servers durch Modifikationen im Quellcode der Datei *dhclient-script* verhindert und in eine Datei umgelenkt wird (mehr dazu in Kapitel 5.3), konnte bequem auf diese beiden Dateien (*route.txt* und *resolv<Interface>.txt*) zurückgegriffen und die Daten für Gateway und DNS-Server ausgelesen werden.

```
lgateway=`cat /tmp/netroam/route.txt | grep ${interface[m]} | tail -n 1 | awk '{print $7}'`
echo $lgateway >> /tmp/netroam/${interface[m]}.txt
ldns=`cat /tmp/netroam/resolv${interface[m]}.txt | head -n 1`
echo "$ldns" >> /tmp/netroam/${interface[m]}.txt
```

Die zweite Funktion, die in der Endlosschleife des Hauptprogramms aufgerufen wird, ist die Funktion *connection*. Sie ist in den Teil für LAN-Interfaces und in den Teil für WLAN-Interfaces unterteilt, wobei beide Teile nur dann durchlaufen werden, wenn auch die Priorität auf dem jeweiligen Teil, also LAN oder WLAN liegt. Der LAN-Teil ist wiederum in zwei Bereiche aufgeteilt. Im ersten Bereich, der bei statischer Adresszuweisung durchlaufen wird, wird zunächst erneut eine default-Route mit Hilfe der Daten aus der Datei *<Interface>.txt* gesetzt.

```
#Setzen der Route
route add default dev $t metric 100 gw $gateway 2>/dev/null
```

Das erneute Setzen der Route ist eine Vorbereitung für die Funktion *set_route*, die das Interface auf Internetkonnektivität prüft und die Routen setzt, und als nächstes aufgerufen wird. Auf den Aufbau dieser Funktion wird später noch eingegangen.
War der Verbindungsaufbau erfolgreich und ist über das Interface somit das Internet verfügbar, so werden in der Funktion *connection* die Daten der Verbindung für das Webfrontend in der Datei *infolan.txt* gespeichert. Dem Benutzer wird mitgeteilt, dass der

```
#Einfügen der aktuellen Verbindungsdaten für das Webfrontend
echo $status_lan > /tmp/netroam/infolan.txt
echo $ip >> /tmp/netroam/infolan.txt
echo $netmask >> /tmp/netroam/infolan.txt
echo $broadcast >> /tmp/netroam/infolan.txt
echo $gateway >> /tmp/netroam/infolan.txt
echo $dns >> /tmp/netroam/infolan.txt
echo -n $speed >> /tmp/netroam/infolan.txt
```

Verbindungsaufbau über das Interface erfolgreich war. Im Fehlerfall, also bei keiner Internetverfügbarkeit, wird dies dem Benutzer ebenfalls mitgeteilt, welcher dann die Möglichkeit hat die Priorität wieder auf WLAN zu setzen. In diesem Fall wird wieder eine 1 in die Datei *prio.txt* geschrieben und die Funktion *connection* verlassen. Sollte zuvor keine WLAN-Verbindung bestanden haben, wird nun versucht eine aufzubauen.
Der zweite Bereich des LAN-Teils, der aufgerufen wird wenn das Interface per DHCP eine Adresse bekommen hat, funktioniert bis auf eine Ausnahme analog zum ersten Bereich.
Der einzige Unterschied liegt in dem Setzen der Route. Hier wird auf die Datei *route.txt* zurückgegriffen in der die Route steht, die per DHCP übergeben wurde. Da diese jedoch noch ohne Metrik versehen ist, muss dies vor dem Setzen zunächst geändert werden.

```
#Auslesen der Route, die per dhcp Übergeben wurde
route=`cat /tmp/netroam/route.txt | grep "$t" | tail -n1`
#Setzen der neuen Route mit Metric 100
aroute=`echo "$route metric 100"`
exec $aroute &
```

Dafür wird der ursprünglichen Route aus der Datei *route.txt* die Metrik 100 angehangen, und anschließend mit dem Befehl *exec*[34] gesetzt.

34 Der Befehl exec ist normalerweise für das Ausführen von Programmen vorgesehen, kann jedoch auch für das Ausführen von Befehlen in Variablen benutzt werden.

Ebenfalls wird in dem LAN-Teil der Fall behandelt, wenn die Priorität auf LAN gesetzt wird, zuvor jedoch kein bekanntes Netzwerk gefunden wurde. In diesem Fall wird der Benutzer mittels Zenity ebenfalls informiert und je nachdem, ob der Benutzer es wünscht, die Priorität wieder auf WLAN geändert.

Bei dem zweiten Teil der Funktion *connection*, dem Teil für WLAN-Interfaces, ist zu beachten, dass dieser nur für ein WLAN-Interface aufgerufen wird, welches keine Verbindung besitzt und zuvor die Umgebung gescannt hat. Ist dies bei dem aktuellen Interface der Fall, muss zunächst überprüft werden, ob es neben diesem Interface noch ein weiteres gibt und welchen Status dieses besitzt.

```
q=0
for j in ${interface[*]}
do
	#wenn das Interface eine WLAN-Karte ist und es sich nicht um dieselbe Karte handelt
	if test "${name[q]}" = "wlan" && test "$j" != "$t"
	then
		ak=$j
		#auselesen des Status der zweiten wlan-Karte
		stat=`cat /tmp/netroam/$j.txt | head -n 1`
		if test "$stat" = "1"
		then
			#auslesen der Signalstärke mit dem momentan verbundenen Access Point
			signal_con=`cat /tmp/netroam/$j.txt | tail -n 1 | awk -F, '{print $2}'`
			mac_con=`cat /tmp/netroam/$j.txt | tail -n 1 | awk -F, '{print $1}'`
		fi

	fi
q=`expr $q + 1`
done
```

Besitzt es den Status 1, also verfügt es momentan über eine Verbindung, so wird die aktuelle Signalstärke und die MAC-Adresse des Access Point aus der Datei *<Interface>.txt* ausgelesen und zur späteren Verwendung in die Variable *signal_con* gespeichert.

Die weitere Vorgehensweise dieser Funktion ist nun auch wieder in zwei Bereiche unterteilt. Zum einen in den Bereich, wenn das zweite WLAN-Interface eine Verbindung besitzt, zum anderen wenn es keine Verbindung besitzt. Im ersten Teil wird geprüft, ob die aktuelle WLAN-Karte einen besseren Access Point gefunden hat. Dafür wird die aktuelle Signalstärke aus der Variablen *signal_con* mit der Signalstärke in der Datei *<Interface>.txt* verglichen. Wurde ein besserer Access Point gefunden, so wird versucht, sich mit Hilfe der Funktion *wlan2*, mit diesem zu verbinden. Besteht anschließend über diesen Access Point

eine Internetverbindung, welche wieder mittels der Funktion *set_route* getestet und in der Funktion *wlan2* aufgerufen wird, so erhält der Benutzer eine Bestätigung. Besteht wiederum keine Internetkonnektivität über diesen Access Point so wird noch mehrmals, je nachdem wie viele Versuche der Benutzer über das Webfrontend eingestellt hat, versucht, eine Internetverbindung über diesen Access Point zu erlangen.

```
#Auslesen der max. möglichen Versuche
anz_vers=`cat /etc/netroam/config.txt | grep "anz_vers" | awk -F: '{print $2}'`
for versuch in `seq 2 $anz_vers`
do
      wlan2 #Aufruf der Funktion wlan2
      if test "$ping2" = "da"
      then
           popup=`cat /etc/netroam/config.txt | grep "connectionok" | awk -F: '{print $2}'`
           if test "$popup" = "1"
           then
                zenity --info --title="NetRoam" --text="Verbindungsaufbau mit Access Point
                \"$ap_name2\" über Interface $t war erfolgreich"
           fi
           ...
           break
      fi
done
```

Sollte auch nach dem letzten Versuch keine Internetverbindung zustande kommen, so wird der aktuelle Router auf die Blacklist gesetzt und der nächste Access Point in der Liste ver-

```
if test "$ping2" != "" && test "$ping2" != "da"
then
      ...
      echo -n "$wmac,$ap_name2,$wenc" >>/etc/netroam/blacklist.txt
      blacklist=true
fi
```

sucht. Sollte der Verbindungsaufbau schließlich doch erfolgreich verlaufen sein und der Access Point auch über Internetkonnektivität verfügen, so werden die Daten der Verbindung in der Datei *infowlan<Interface>.txt* gespeichert. Tritt jedoch der Fall ein, dass auch nach dem letzten Access Point aus der Datei *<Interface>.txt* keine Internetverbindung zustande kam, so erhält der Benutzer eine Fehlermeldung und kann auf LAN wechseln.
Der zweite Bereich in dem WLAN-Teil, verläuft nahezu identisch zum ersten Teil. Da dieser Teil wie schon erwähnt jedoch nur durchlaufen wird, wenn das zweite WLAN-Interface keine Verbindung besitzt, oder es gar kein zweites Interface gibt, müssen hier keine Signalstärken verglichen werden, sondern es wird lediglich versucht über den ersten Access

Point in der Datei *<Interface>.txt* und damit auch den mit der besten Signalstärke, eine Internetverbindung aufzubauen. Auch hier wird der Access Point, wenn keine Internetverbindung zustande kommt, wenn gewünscht, mehrmals auf Internetkonnektivität überprüft. Gelingt dies nach dem letzten Versuch nicht, so wird dieser ebenfalls wieder auf die Blacklist gesetzt und der nächste Access Point aus der Datei versucht.

Auch behandelt der WLAN-Teil den Fall, wenn kein einziger bekannter Access Point gefunden wurde. In diesem Fall wird der Benutzer darauf hingewiesen und die Priorität, wenn gewünscht, wieder auf LAN gesetzt.

Der Programmablaufplan auf der nächsten Seite soll noch einmal den Vorgang und den Aufbau der Funktion *connection* verdeutlichen.

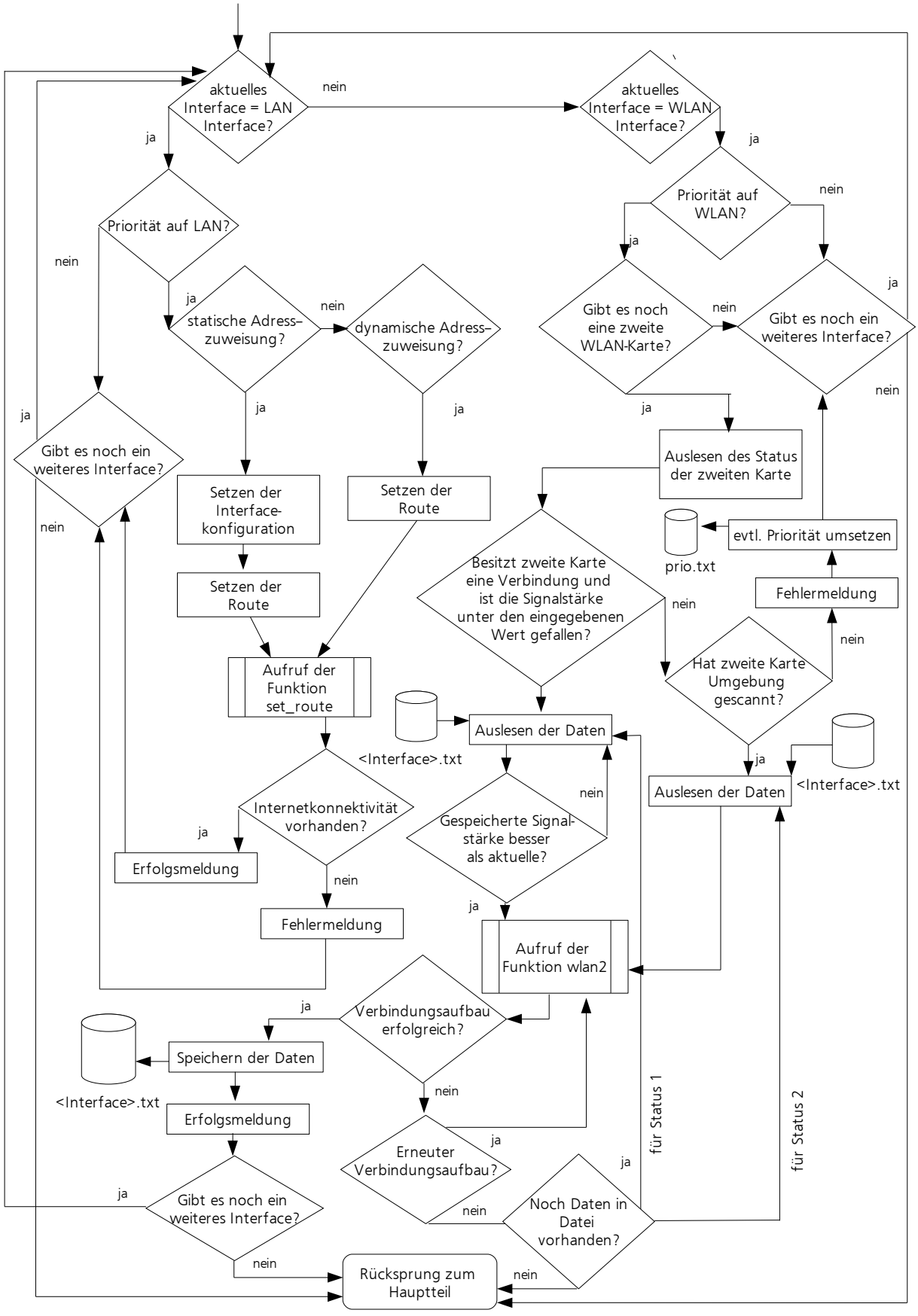

Abbildung 11: Ablaufplan Funktion connection

Bevor auf die Funktion *set_route* eingegangen werden kann, muss zunächst die Funktion *wlan2* erläutert werden, die für den Verbindungsaufbau zu einem Access Point zuständig ist. Dazu werden in der Funktion die Daten für die Verbindung ausgelesen und je nachdem, welche Verschlüsselung bei dem jeweiligen Router vorliegt, eine andere Funktion mittels einer Switch-Case-Anweisung aufgerufen.

```
#Auslesen der Verschlüsselung
wenc=`echo $best_ap | awk -F, '{print $5}'`
#ausführen der Funktion für die jeweilige Verschlüsselung
case $wenc in
wep)
	wep
;;
wpa)
	wpa
;;
wpa2)
	wpa2
;;
peap)
	peap
;;
esac
```

Dabei sind bis auf die WEP-Funktion alle drei Funktionen gleich aufgebaut. Sie erzeugen jeweils eine Konfigurationsdatei (siehe Kapitel 3.2.3), in der dann die spezifischen Daten eingefügt werden. Nur die Funktion WEP benutzt zum Verbindungsaufbau wie schon in Kapitel 3.2.3 erwähnt, den Befehl *iwconfig*. Wenn die Funktionen ohne Fehler durchlaufen wurden, wird in der Funktion *wlan2* die Konfiguration mit dem Programm wpa_supplicant ausgeführt. Dabei ergab sich jedoch bei der Entwicklung ein weiteres Problem. Da für jede neue WLAN-Verbindung ein neuer Prozess gestartet wird, also wenn das Programm wpa_supplicant ausgeführt wird, liefen anfangs mehrere wpa_supplicant-Prozesse im Hintergrund, obwohl sie gar nicht mehr benutzt wurden. Aus diesem Grund wurde versucht, beim Starten von wpa_supplicant, die PID des Prozesses zu speichern, damit der Prozess bei dem nächsten Aufbau einer WLAN-Verbindung beendet werden konnte. Da Ubuntu jedoch keinen Weg vorsieht, die PID beim Starten abzuspeichern, wurde mit Hilfe des Befehls ps, der alle Prozesse auflistet, und des Befehls grep die PID des wpa_supplicant Prozesses abgespeichert.

```
#Speichern der PID des wpa-Prozesses zum späteren Beenden
pid_wpa=`ps aux | grep "wpa_supplicant" | grep "$t" | awk '{print $2}'`
```

Die weitere Vorgehensweise der Funktion ähnelt nun der Vorgehensweise der Funktion *connection* im LAN-Teil. Auch hier werden zwei Teile verwendet, zum einen für statische,

zum anderen für dynamische Adresszuweisung. In beiden Teilen werden die Routen mit Metrik 100 gesetzt, wobei in dem Teil für dynamische Adressierung zuvor noch versucht wird von dem Access Point mit Hilfe des Befehls *dhclient* eine IP-Adresse zu bekommen. Erfolgt dies nicht, so wird der wpa_supplicant Prozess beendet.

```
#Wenn dhcp-Anfrage nicht erfolgreich, beenden des wpa_supplicant Prozesses
kill -9 $pid_wpa
```

Läuft jedoch die DHCP-Anfrage erfolgreich, so wird die Funktion *set_route* zum Testen des Access Points auf Internetkonnektivität und zum Setzen der Routen aufgerufen. Ebenfalls geschieht dies im Teil für statische Adressierung.

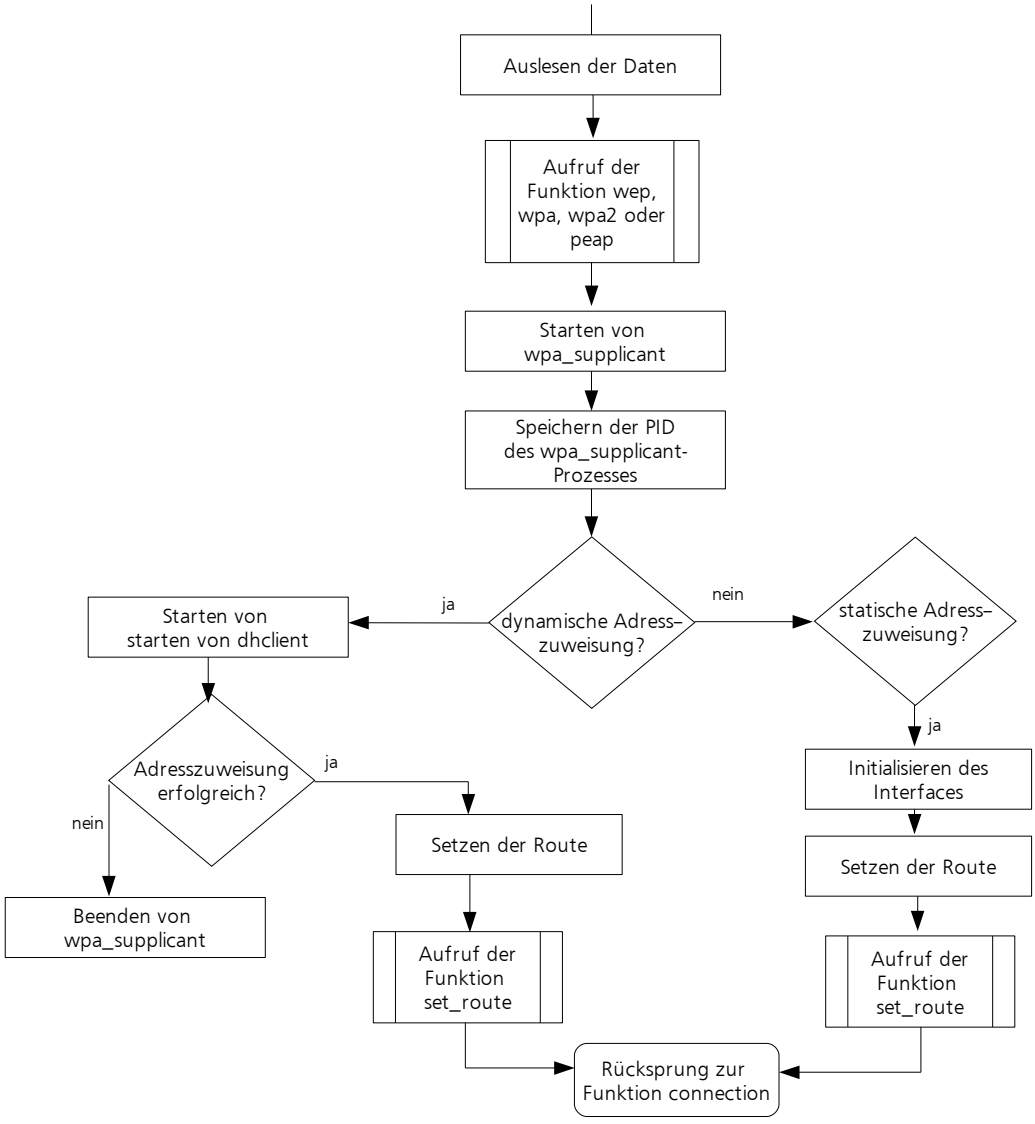

Abbildung 14: Ablaufplan der Funktion wlan2

Die Funktion *set_route* ist, wie schon mehrmals erwähnt, für das Testen der Internetverbindung und im Erfolgsfall für das Umsetzen der Routen verantwortlich. Aus diesem Grund wird zu Beginn der Funktion ein Nameserver im Internet angepingt, um zu erfahren, ob das jeweilige Interface über Internetkonnektivität verfügt.

```
#Testen der Konfiguration auf Internetverfügbarkeit
ping=`ping -I $2 217.237.150.141 -c1 2>/dev/null`
```

Verläuft dieser Versuch erfolgreich, werden alle default-Routen aus der Routingtabelle ausgelesen und gespeichert. Sowohl bei statischer als auch bei dynamischer Adressierung, werden nun die DNS-Server in die Datei *resolv.conf*[35] gesetzt. Anschließend wird eine neue Route mit Metrik 0 gesetzt und die Route mit Metrik 100, die zum Testen der Verbindung zuständig war, gelöscht.

```
#statischer Adressierung:
#setzen des DNS-Servers
echo "nameserver $dns" > /etc/resolv.conf
#Setzen einer neuen Route mit Metric 0
route add default dev $2 metric 0 gw $gateway 2>/dev/null
#Löschen der alten Route mit Metric 100
route del default dev $2 metric 100 gw $gateway 2>/dev/null
```

```
#dynamische Adressierung
#auslesen und setzen des DNS-Servers
dns=`cat /tmp/netroam/resolv$2.txt`
echo "nameserver $dns" > /etc/resolv.conf
#Setzen einer neuen Route mit Metric 0
exec $route & 2>/dev/null
route2=`echo $aroute | sed 's/add/del/'`
exec $route2 &  2>/dev/null #Löschen der alten Route mit Metric 100
```

Ab diesem Moment laufen alle Pakete über diese neue Route. Damit die Routingtabelle nicht unnötig vergrößert und unübersichtlich wird, werden die zuvor gefundenen default-Routen gelöscht. Dies geschieht im folgenden Codeausschnitt.

35 In der Datei resolv.conf steht unter Kubuntu der momentane Nameserver

```
h=0
for z in  ${inter_old[*]}
do
	#Wenn default-Routen mit Metric 0 gefunden wurden, löschen.
	if test "${metric_old[$h]}" = "0"
	then
		route del default dev $z metric 0 gw ${gw_old[$h]} 2>/dev/null
	fi
	h=`expr $h + 1`
done
```

In dieser Funktion müssen ebenfalls die wpa_supplicant-Prozesse behandelt werden. Da die Funktion sowohl von LAN- als auch WLAN-Interfaces aufgerufen werden kann, muss diese Unterscheidung auch in dieser Funktion auftreten. Im ersten Fall, wenn die Funktion *set_route* von einem LAN-Interface aufgerufen wird, können nach erfolgreicher Verbindung ohne Risiko alle wpa_supplicant-Prozesse beendet werden. In dem Fall, dass keine wpa_supplicant- Prozesse vorhanden sind, läuft das Beenden ins Leere.

```
#wenn Verbindung über LAN erfolgreich war: beenden aller wpa_supplicant Prozesse
killall wpa_supplicant 2>/dev/null
```

Wurde die Funktion *set_route* jedoch von einem WLAN-Interface aufgerufen, so muss zuerst geprüft werden, ob es einen alten wpa_supplicant-Prozess gibt.

```
#Auslesen ob es einen alten wpa_supplicant Prozess der zweiten WLAN Karte gibt
pidold=""
if [ -e /tmp/netroam/pid_wpa$ak.txt ]
then
	pidold=`cat /tmp/netroam/pid_wpa$ak.txt | head -n 1`
fi
```

Hierbei können nicht einfach alle wpa_supplicant-Prozesse beendet werden, da man damit auch die aktuelle Verbindung beenden würde. Wird ein alter Prozess gefunden, so wird nur dieser beendet.

```
if test "$pidold" != ""
then
	kill -9 $pidold
	echo "" >/tmp/netroam/pid_wpa$ak.txt
fi
```

Sollte der Fall eintreten, dass der Ping zum Nameserver nicht erfolgreich war, so wird dem aktuellen Interface die IP-Adresse 0.0.0.0 zugewiesen, sodass es alle Einstellungen verliert. Ebenfalls wird die schon gesetzte Route mit Metrik 100 entfernt. Handelt es sich zusätzlich

um ein WLAN-Interface, welches die Funktion *set-route* aufgerufen hat, so wird der dazugehörige wpa_supplicant-Prozess ebenfalls beendet.

```
#Löschen der Route mit Metrik 100 und zuweisen der IP 0.0.0.0
route del default dev $2 metric 100 gw $gateway
ifconfig $2 0.0.0.0
#Wenn Funktion set_route von einem wlan interface aufgerufen wurde
if test "$3" = "0"
then
      pidold=`cat /tmp/netroam/pid_wpa$2.txt | head -n 1`
      kill -9 $pidold
      echo "" >/tmp/netroam/pid_wpa$2.txt
fi
```

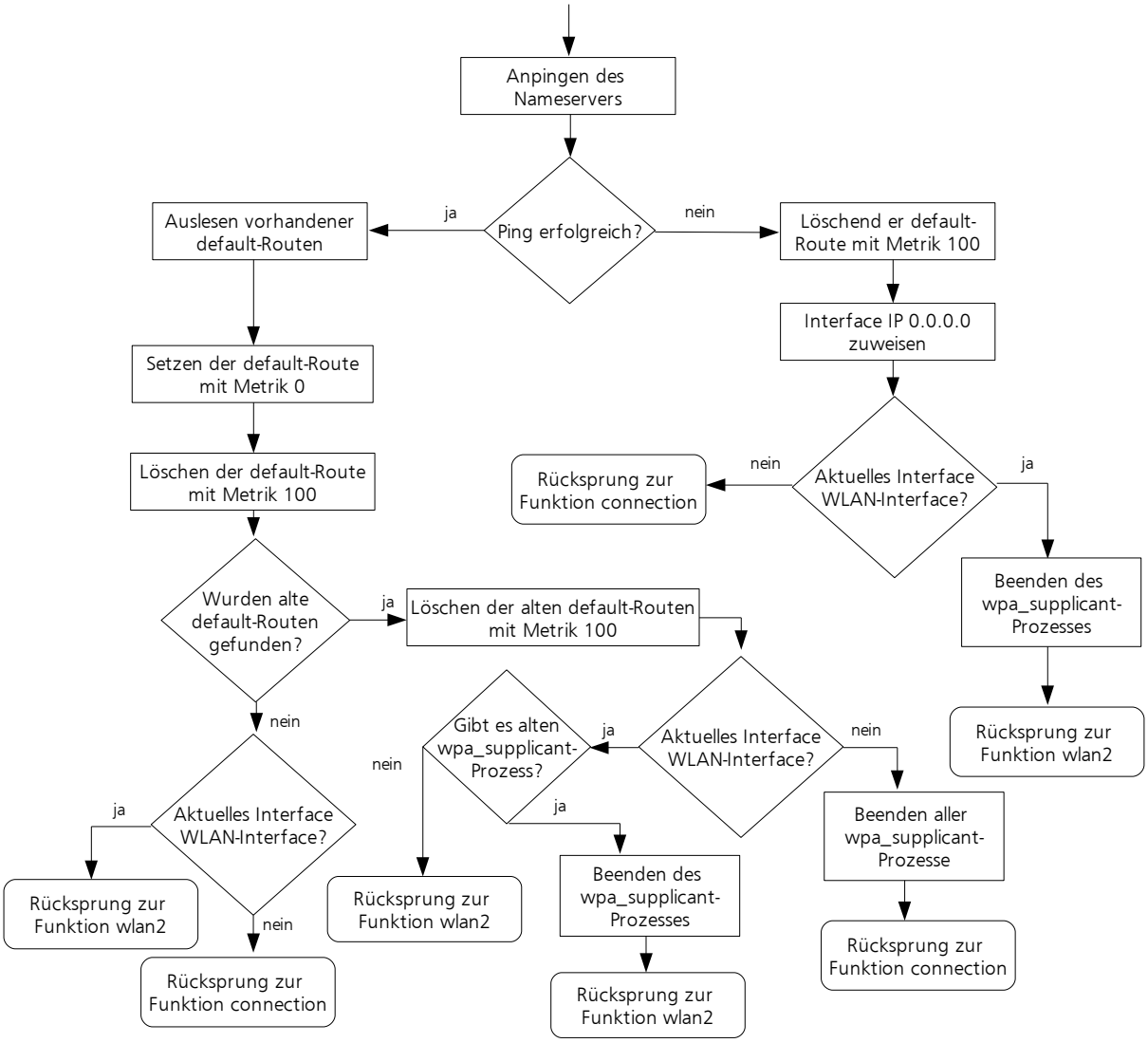

Abbildung 15: Ablaufplan Funktion set_route

Wurden schließlich alle Funktionen durchlaufen, beginnt der komplette Vorgang, beginnend mit der Funktion init, von Vorne.

5.3 Notwendige Modifikationen

Um das Problem, welches im Lösungsansatz beschrieben wurde, zu lösen, musste die Datei dhclient-script angepasst werden. Bei dieser Datei handelt es sich um ein Script, das immer dann ausgeführt wird, wenn dem System eine IP-Adresse per DHCP zugewiesen werden soll. Dies kann beim Starten des Rechners sein, oder nachdem im laufenden Betrieb der Befehl *dhclient* eingegeben wurde, welcher für eine erneute DHCP-Anfrage benutzt wird.

Bei der ersten Modifikation wurde ein Teil hinzugefügt, welcher wie folgt aussieht:

```
...
      # The action starts here

      # Invoke the local dhcp client enter hooks, if they exist.
      run_hook /etc/dhcp3/dhclient-enter-hooks
      run_hookdir /etc/dhcp3/dhclient-enter-hooks.d

      ######## ERWEITERUNG ########
      # Bachelor-Thesis #
      # Alexander Kniwel #
      ########################################
      #
      # Erstellen des Ordners /tmp/netroam/
      #
      ########################################

      if [ ! -d /tmp/netroam ]
      then
             mkdir /tmp/netroam
      fi

      ########## ENDE DER ERWEITERUNG ###########

      # Execute the operation
      case "$reason" in
             MEDIUM|ARPCHECK|ARPSEND)
             # Do nothing
             ;;
...
```

Hierbei wird geprüft, ob der Ordner */tmp/netroam/* schon existiert. Besteht dieser noch nicht, so wird er erstellt. Dies ist wichtig, da sonst die Dateien, die in diesem Ordner erzeugt werden sollen, nicht erzeugt werden und somit die abgefangenen Daten nicht gespeichert werden könnten.

Bei der nächsten Modifikation wird eine Änderung am Script vorgenommen. Diese Änderung sorgt dafür, dass die vom Script erzeugte default-Route nicht gesetzt, sondern in eine Datei umgelenkt wird. Die Änderung, die dafür nötig war, ist im nachfolgenden Codeausschnitt zu sehen.

```
...
      # point to point
             if [ "$new_subnet_mask" == "255.255.255.255" ]; then
                    for router in $new_routers; do
                           route add -host $router dev $interface
                    done
             fi
      ######## ÄNDERUNG ########
      # Bachelor-Thesis #
      # Alexander Kniwel #
      ##########################################
      #
      # Umlenken der Route in die Datei /tmp/netroam/route.txt
      #
      ##########################################

      for router in $new_routers; do
      route add default dev $interface gw $router $metric_arg
      echo "route add default dev $interface gw $router $metric_arg" >>
      /tmp/netroam/route.txt
      done

      ########## ENDE DER ÄNDERUNG ###########

      if [ "$new_ip_address" != "$alias_ip_address" -a -n "$alias_ip_address" ];
      then
...
```

Weil in dieser Datei auch mehrere Routen von verschiedenen Interfaces stehen können, muss sichergestellt sein, dass immer auf den letzten Routeneintrag des jeweiligen Interfaces zugegriffen werden kann. Dies kann durch den Befehl

```
#Auslesen der Route, die per DHCP übergeben wurde
route=`cat /tmp/netroam/route.txt | grep "$t" | tail -n1`
```

realisiert werden. Dabei kann sich zu Nutzen gemacht werden, dass in einer default-Route immer das dazugehörige Interface steht. Nach diesem Interface kann mit Hilfe des Befehls *grep* gesucht und mit dem Befehl *tail -n1* auf den letzten gefundenen Eintrag zugegriffen werden.

Die letzte Änderung, die in der Datei vorgenommen wurde, hat die Aufgabe, das Setzen des DNS-Servers in die Datei *resolv.conf* zu verhindern, da Ubuntu bei einer DHCP-Anfrage ebenfalls den Nameserver ändert. Dies darf aber, solange eine aktive Verbindung über ein anderes Interface besteht, nicht passieren, sondern wird vom Programm NetRoam zum richtigen Zeitpunkt ebenfalls übernommen. Die für diese Änderung nötige Modifikation ist in dem nachfolgenden Codeausschnitt dargestellt.

```
make_resolv_conf() {
if [ -n "$new_domain_name" -o -n "$new_domain_name_servers" ]; then
...
       ######## ÄNDERUNG ########
       # Bachelor-Thesis #
       # Alexander Kniwel #
       ##########################################
       #
       # Umlenken des DNS-Servers in die Datei /tmp/netroam/resolv$interface.txt
       #
       ##########################################
       mv -f $new_resolv_conf /etc/resolv.conf
       echo "$nameserver" > /tmp/netroam/resolv$interface.txt
       ########## ENDE DER ÄNDERUNG ###########
fi
}
```

Hierbei werden die DNS-Server für die jeweiligen Interfaces in verschiedenen Dateien gespeichert, da man sie sonst keinem Interface mehr zuweisen könnte.

Eine weitere Datei, die angepasst werden musste, ist die Datei *etc/network/interfaces*. Damit die Interfaces beim Starten des Rechners und beim späteren Zugreifen auf die Datei durch andere Programme nicht beachtet werden, wurden alle Einträge zu den jeweiligen Interfaces entfernt, sodass nur noch die Einstellungen für das Loopback-Interface vorhanden sind.

```
       auto lo
       iface lo inet loopback
       address 127.0.0.1
       netmask 255.0.0.0
```

Durch die Modifikation der beiden Dateien konnte das Problem der Routingtabelle und des DNS-Servers gelöst werden.

5.4 Schnittstellen zum Webfrontend

Wie schon des Öfteren erwähnt, wurden zahlreiche Schnittstellen für die Kommunikation zwischen dem Programm NetRoam und dem Webfrontend erzeugt. Diese Schnittstellen bestehen aus Dateien, in denen die jeweiligen Informationen des Programms oder des Webfrontend abgespeichert werden. Beide Teile können somit bequem auf die Dateien zugreifen und besitzen immer die aktuellen Daten des jeweils anderen Teils.

Die Schnittstellen müssen in zwei Arten aufgeteilt werden. Zum einen in temporäre Dateien, zum anderen in nicht temporäre Dateien. Die nicht temporären Dateien enthalten alle Daten, die auch nach einem Neustart des Rechners noch vorhanden sein sollen. Dies sind Informationen über die hinterlegten Router und Access Points in den Dateien *lan.csv* und *wlan.csv,* aber auch die Dateien *config.txt, blacklist.txt* und *prio.txt* gehören zu diesen nicht temporären Dateien.

In den temporären Dateien stehen die momentanen Verbindungsdaten der jeweiligen Interfaces. Für WLAN-Interfaces gibt es ebenfalls noch temporäre Dateien, in denen die gefundenen WLANs und die bekannten WLANs stehen. Im Folgenden soll auf die Dateien kurz eingegangen werden.

lan.csv

In der Datei *lan.csv* stehen die durch das Webfrontend eingetragenen Router. Ein Ausschnitt aus dieser Datei ist im Nachfolgenden gezeigt.

```
192.168.0.50,255.255.255.0,192.168.0.255,192.168.0.1,192.168.0.1
214.62.12..248,255.255.255.0,214.62.12.255,214.62.12.1,214.62.12.2
...
```

Abbildung 16: Aufbau der Datei lan.csv

Wie zu erkennen ist, stehen in dieser Datei nur die Router, bei denen keine Adressierung per DHCP stattfindet, sondern die auf statische Adressen setzen. Router, die eine Adressierung per DHCP zulassen werden in dieser Datei nicht gespeichert, da ja das Programm NetRoam diese Router selbständig erkennt.

wlan.csv

In die Datei *wlan.csv* werden alle bekannten Access Points geschrieben, mit denen sich das Programm NetRoam verbinden kann. In diese Datei werden nicht, wie bei der Datei *lan.csv* nur die Access Points eingetragen die über statische Adressierung verfügen, sondern auch die, die DHCP verwenden. Beide Dateien, *lan.csv* und *wlan.csv,* können, während das Programm NetRoam läuft, über das Webfrontend geändert werden.

```
Netroam,dhcp,00:18:F8:FA:12:34,wpa,key
WlanRoam,static,00:18:F8:FA:56:78,peap,Alex,Alex,192.168.0.10,255.255.255.0,192.168.0.255,192.168.0.1,192.168.0.1
Dnlab,static,00:18:F8:FA:FA:01,wep,key,192.168.1.43,255.255.255.0,192.168.1.255,192.168.1.1,192.168.1.1
Netroam2,dhcp,00:18:F3:57:3D:34,peap,Alex,Alex
...
```

Abbildung 17: Aufbau der Datei wlan.csv

config.txt

Die Datei *config.txt* wird mit Hilfe des Webfrontends unter dem Link "Settings" manipuliert. In dieser steht, neben der Signalstärke bei der ein WLAN gewechselt werden soll und der Anzahl der Versuche des Verbindungsaufbaus mit einem Access Point, auch, ob ein Zenity-Fenster erscheinen soll oder nicht.

```
signal_strength:-40
anz_vers:5
leer:0
lanup:1
landown:0
startconnection:1
newwlanfound:1
connectionok:1
connectionnotok:1
nointernet:1
norouter:0
noap:0
blacklist:1
nodhcp:0
```

Abbildung 18: Aufbau der Datei config.txt

Aus diesem Grund sind in der Datei die Fenster nacheinander aufgelistet und enthalten den Wert 0 für nicht erscheinen und den Wert 1 für erscheinen.

blacklist.txt

Die Datei *blacklist.txt* kann sowohl vom Programm NetRoam als auch vom Webfrontend manipuliert werden. In diese Datei schreibt NetRoam alle Access Points über die auch nach mehrmaligen Versuchen keine Internetverbindung zustande kam. Nur über das Webfrontend können diese Access Points wieder aus der Datei gelöscht werden. Der Aufbau dieser Datei sieht dabei so aus:

```
00:18:F8:FA:56:78, WlanRoam,peap
00:18:F8:FA:12:34,Netroam,wpa
...
```

Abbildung 19: Aufbau der Datei blacklist.txt

prio.txt

Die Datei *prio.txt* wird ebenfalls vom Programm NetRoam als auch vom Webfrontend manipuliert. In dieser steht immer welche Art von Interface priorisiert ist. Entweder liegt die Priorität auf LAN oder auf WLAN, welches in dieser Datei mit einer 0 bzw. mit einer 1 gekennzeichnet wird. Mit Hilfe des Webfrontends kann die Priorität umgesetzt werden, was ebenfalls das Umsetzen von 1 auf 0 (WLAN -> LAN) oder von 0 auf 1 (LAN -> WLAN) zur Folge hat.

infolan.txt

Die erste temporäre Datei ist die Datei *infolan.txt* und ist, wie der Name schon sagt, für das LAN-Interface zuständig. In dieser Datei werden immer die aktuellen Informationen über das LAN-Interface gespeichert, sodass zu jedem Zeitpunkt über das Webfrontend nachvollzogen werden kann, mit welchem Netz das Interface gerade verbunden ist. Die Datei ist gleich aufgebaut wie die Datei *<Interface>.txt* für LAN-Interfaces und enthält ebenfalls die aktuellen Verbindungsdaten (siehe Kapitel 5.2 Abbildung 13).

infowlan<Interface>.txt

Die Datei *infowlan<Interface>.txt* besitzt bei einer aktiven Verbindung dieselbe Aufgabe für WLAN-Interfaces, wie die Datei *infolan.txt* für LAN-Interfaces. In dieser Datei werden ebenfalls die aktuellen Verbindungsdaten für das jeweilige Interface gespeichert. Dabei kann es maximal zwei dieser Dateien geben, da für jedes WLAN-Interface eine eigene Datei erzeugt wird. Aus diesem Grund steht auch der Parameter <Interface> im Namen der Datei, welcher durch den jeweiligen Namen des Interfaces beim Erstellen ersetzt wird. Der Aufbau dieser Datei, bei einer aktiven Verbindung, ist in der nachfolgenden Abbildung dargestellt.

```
1,-50,Netroam,dhcp,00:18:F8:FA:12:34,wpa
```

Abbildung 20: Aufbau der Datei infowlan<Interface>.txt bei aktiver Verbindung

Besitzt ein Interface eine Verbindung, so kann das zweite Interface keine Verbindung mehr besitzen und muss die Umgebung nach bekannten Access Points gescannt haben. In diesem Fall sieht die Datei *infowlan<Interface>.txt* für das zweite Interface wie in Abbildung 21 dargestellt ist aus. Dabei ist zu beachten, dass hier nur die bekannten Access Points gespeichert werden.

```
2,-45,Netroam2,dhcp,00:18:F3:57:3D:34,peap
...
```

Abbildung 21: Aufbau der Datei infowlan<Interface>.txt bei keiner aktiven Verbindung

foundwlan<Interface>.txt

In der Datei *foundwlan<Interface>.txt* werden alle im Moment gefundenen Access Points gespeichert. Auch von dieser Datei gibt es, wie von der Datei *infowlan<Interface>.txt,* maximal zwei Stück. Besitzt ein Interface aktuell eine Verbindung, so steht in der für dieses Interface zugehörigen Datei lediglich das Wort connection, welches dem Webfrontend signalisieren soll, dass dieses Interface über eine Verbindung verfügt und nicht nach Access Points gescannt hat.

Für den Fall, dass ein Interface keine Verbindung besitzt und nach Access Points gescannt hat, sieht die Datei wie in Abbildung 22 aus.

```
Netroam,00:18:F8:FA:12:34,-34
Dnlab,00:18:F8:FA:FA:01,-65
...
```

Abbildung 22: Aufbau der Datei foundwlan<Interface>.txt bei keiner aktiven Verbindung

Besitzen beide WLAN-Interfaces keine aktive Verbindung, haben auch beide Interfaces nach verfügbaren Access Points gescannt. Dies ist wichtig, da beide Karten zumeist verschiedene Access Points finden.

6 Messung

In diesem Kapitel soll auf die Messungen, die nach der Fertigstellung des Programms NetRoam durchgeführt wurden, eingegangen werden. Zunächst werden die für die Messungen wichtigen Programme erläutert.

6.1 Software für die Messungen

6.1.1 Wireshark

Wireshark ist ein freier Netzwerk-Protokollanalysator und war früher unter dem Namen Ethereal bekannt. Mit Hilfe von Wireshark ist es möglich, den Datenverkehr zwischen dem eigenen System und einem fremden System aufzuzeichnen. Dazu hat man in Wireshark die Möglichkeit, den Datenverkehr über ein spezielles Interface mitzuschneiden, aber auch das gleichzeitige Mitschneiden von allen Interfaces ist möglich. Dabei ist es egal, ob es sich bei dem Interface um ein LAN- oder WLAN-Interface handelt. Wireshark liegt aktuell in der Version 0.99.6 vor, welche auch in dieser Arbeit verwendet wurde.

6.1.2 Ping

Das Kommandozeilen-Tool Ping wurde nicht nur, wie in der Implementierung (Kapitel 5.2) gezeigt, benutzt, um zu überprüfen, ob ein Interface über Internetkonnektivität verfügt, sondern spielte auch bei den Messungen eine wichtige Rolle.

Mit dem Tool Ping ist es möglich, Netzwerkpakete zu einem anderen Rechner zu schicken, der diese Pakete umgehend wieder an den Versender zurückschickt. Dazu muss der Befehl

ping *IP-Adresse*

in der Konsole eingegeben werden. Der Parameter IP-Adresse muss durch die anzupingende IP-Adresse ersetzt werden. Des Weiteren ist es möglich, mit dem Flag „-i" eine Zeit anzugeben, wie schnell neue Pakete versendet werden sollen. Standardmäßig ist dies auf 1 Sekunde gesetzt. In dieser Arbeit wurde zur exakteren Bestimmung eine Zeit von 1 ms gewählt.

6.1.3 Programm utime

Um die Zeit messen zu können, die das Programm NetRoam beim Durchlaufen eines Bereiches im Quellcode benötigt, wurde ein kleines C-Programm geschrieben, das in dem Programm mit dem Befehl *./utime* aufgerufen werden kann und einen aktuellen Zeitstempel ausgibt. Dieser Zeitstempel kann in eine Datei umgelenkt werden, sodass die verstrichene Zeit durch Subtraktion zweier Zeitstempel ermittelt werden kann. Der Quellcode des Programms ist im nachfolgenden Ausschnitt dargestellt.

```
#include <stdio.h>
#include <sys/time.h>

int main()
{
        struct timeval time;
        struct timezone tzp;
        gettimeofday(&time,&tzp);
        printf("%d.%.6d\n",time.tv_sec,time.tv_usec);

        return 0;
}
```

6.2 Aufbau der Messumgebung

Die für diese Bachelorarbeit aufgebaute Testumgebung bestand zum einen aus zwei Switchs, und zum anderen aus zwei Linksys WRT54G Access Points. Alle Geräte fungierten bei bestehender Verbindung als Gateway zum Netzwerk des Fachbereichs Datennetze.

Das Gerät, welches sich mit den bestehenden Netzen verbunden hat, war ein Notebook der Firma Toshiba. Die Interfaces dieses Notebooks bestanden aus dem Werksseitig eingebauten LAN-Interface (Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+) und dem WLAN-Interface (Intel Corporation PRO/Wireless 2200BG). Des Weiteren kam noch ein PCMCIA-WLAN-Interface (AtherosCommunications, Inc AR5212) zum Einsatz.

Damit ein Wechsel zwischen den beiden Access Points ohne Probleme stattfinden konnte, wurde die Signalstärke bei einem Access Point auf 10 mW herunter gesetzt. Trotz dieses relativ niedrigen Wertes, bestand in naher Umgebung zu dem Access Point die Möglichkeit, sich mit diesem zu verbinden. Wurde der Abstand jedoch größer, so konnte Dank der besseren Signalstärke des zweiten Access Points, zu diesem gewechselt werden.

Die Abbildung 23 zeigt die gesamte Testumgebung mit allen Access Points und Routern, jedoch ohne Notebook.

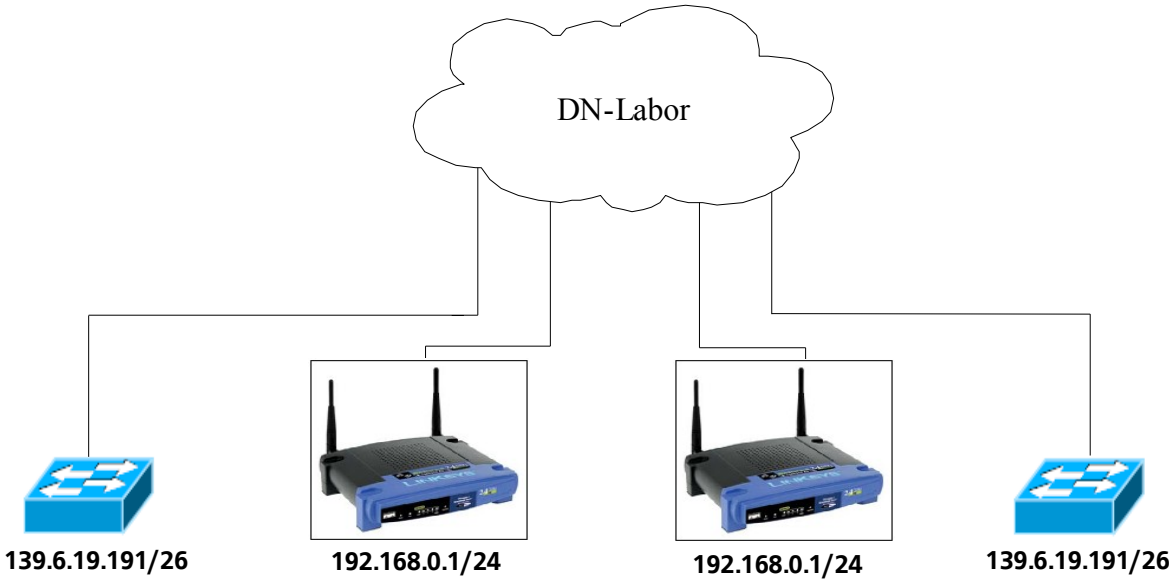

Abbildung 23: Aufbau der Messumgebung

6.3 Messergebnisse

In den nachfolgenden Unterkapiteln werden die Messergebnisse, die mit Hilfe von Wireshark auf dem Notebook aufgezeichnet wurden, gezeigt. Für jeden Netzwechsel ist ein eigenes Unterkapitel vorgesehen. Mit Hilfe dieser Messungen soll gezeigt werden, wie lange es dauert, bis die Verbindung von einem Interface auf ein anderes Interface umgesetzt wurde. Dazu wurde ein lokaler Rechner im DN-Netz angepingt und der Datenverkehr durch Wireshark mitgeschnitten. In Wireshark wurde nach den Paketen mit dem ICMP-Protokoll gefiltert, um nur die Ping-Pakete zu erhalten. Weiterhin wird auf die Messungen eingegangen, die mit Hilfe des Programms utime erstellt wurden.

Auf die Bewertung dieser Messungen wird in Kapitel 6.4 Bezug genommen.

6.3.1 Wechsel von LAN auf WLAN

Der erste Mitschnitt zeigt den Wechsel von einem LAN- zu einem WLAN-Interface.

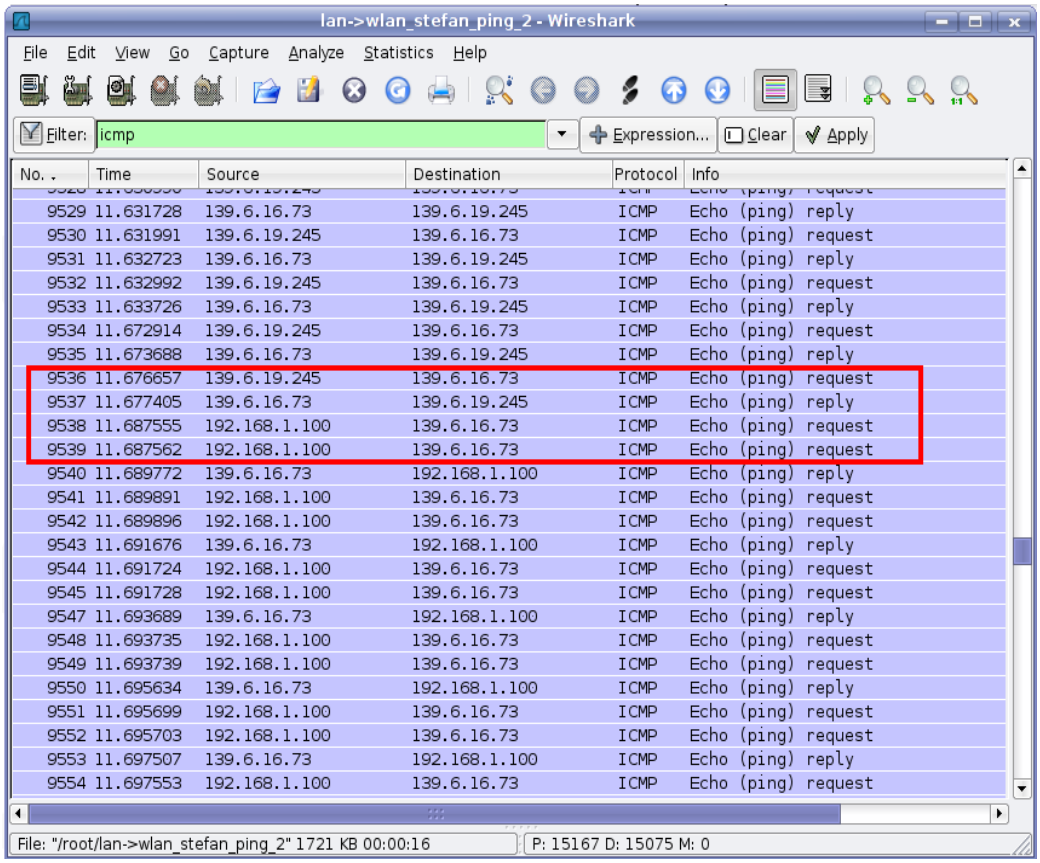

Abbildung 24: Wechsel von LAN auf WLAN

Wie in Abbildung 24 zu erkennen ist, hat der Wechsel von einem LAN-Interface zu einem WLAN-Interface reibungslos funktioniert. Das Paket mit der Nummer 9536 ist der letzte Ping, der das LAN-Interface mit der IP-Adresse 139.6.19.245 verlässt und das Paket mit der Nummer 9537 ist die Antwort dieses Pings. Schon das nächste Paket mit der Nummer 9538 läuft über das neue Interface mit der IP-Adresse 192.168.1.100. Die Zeit die zwischen Paket 9537 und Paket 9538 vergangen ist, gibt Aufschluss darüber, wie lange das Umsetzen der default-Routen und somit das Wechseln des Interfaces gedauert hat. In diesem Fall sind es 11.687555 s – 11.677405 s = 0,01015 s, was ca. 10 ms entspricht.

6.3.2 Wechsel von WLAN zu WLAN

Der nächste Mitschnitt zeigt den Wechsel von einem WLAN- zu einem zweiten WLAN-Interface.

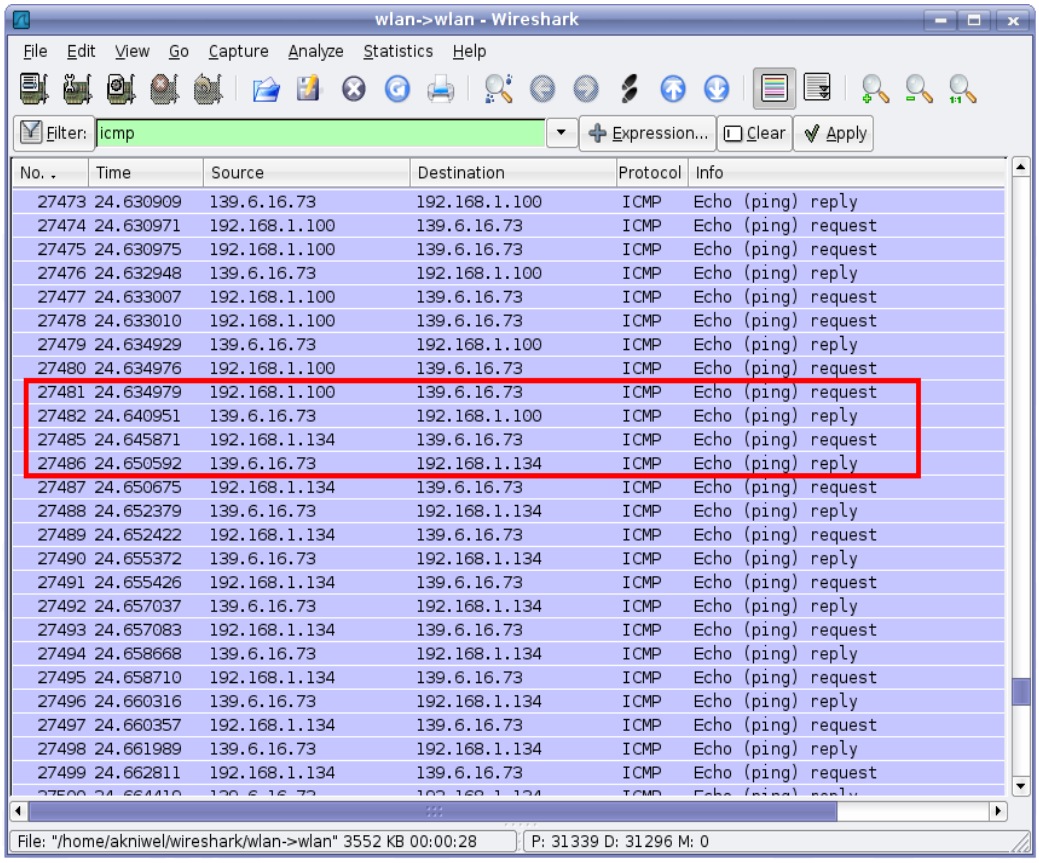

Abbildung 25: Wechsel von WLAN auf WLAN

In Abbildung 25 ist ebenfalls zu erkennen, dass der Wechsel ohne Probleme stattgefunden hat. Das Ping-Paket mit der Nummer 27481 und das Antwortpaket mit der Nummer 27482 sind die letzten beiden Ping-Pakete, die über das erste WLAN-Interface laufen. Über das zweite WLAN-Interface laufen die Ping-Pakete ab dem Paket mit der Nummer 27485. Die zwei fehlenden Pakete mit der Nummer 27483 und 27484 erscheinen durch den ICMP-Filter nicht in der Darstellung. Da dies aber auch nur zwei IGMP-Pakete sind, die für den Multicastverkehr zuständig sind, beeinflussen sie die Messung nicht. Die Zeit, die hier zum Wechsel benötigt wird, beträgt ca. 5 ms (24.645871 s – 24.640951 s = 0.00492 s).

6.3.3 Wechsel von WLAN auf LAN

Die letzte noch ausstehende Messung ist in Abbildung 26 dargestellt und behandelt den Wechsel von einem WLAN- zu einem LAN-Interface.

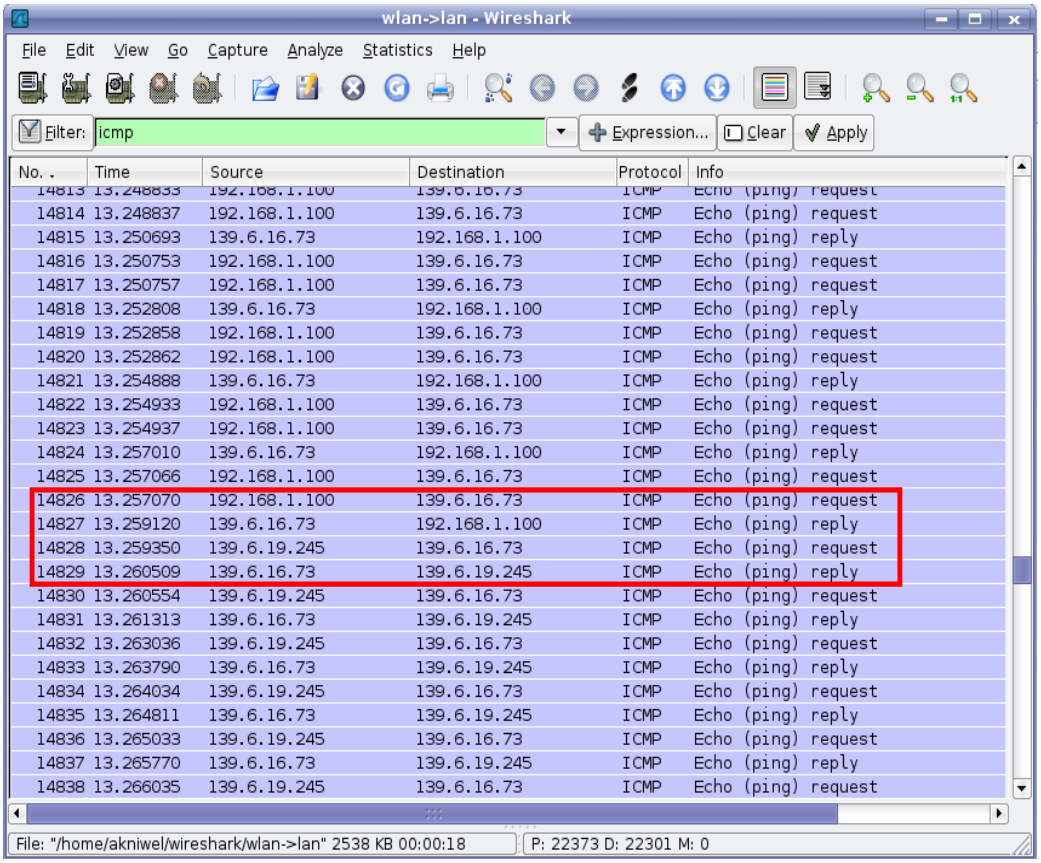

Abbildung 26: Wechsel von WLAN auf LAN

Wie bei den beiden vorherigen Messungen, zeigt sich auch hier, dass der Verbindungswechsel von einem WLAN-Interface zu einem LAN-Interface stattgefunden hat. Die Zeit, die dafür benötigt wurde, ist die kürzeste aller gemessenen Zeiten und beträgt lediglich 0,23 Millisekunden (13.259350 s – 13.259120 s = 0,00023 s).

6.3.4 Messungen des Programms utime

Die mit dem Programm *utime* erzeugten Messungen werden in diesem Kapitel dargestellt. Die erste Messung, die mit dem Programm durchgeführt wurde, gibt die Zeit an, bis NetRoam einen bekannten Access Point gefunden und sich mit diesem verbunden hat, nachdem, während einer aktiven LAN-Verbindung, das Netzwerkkabel gezogen wurde. Dabei wurde vor der Messung sichergestellt, dass der Access Point, der zum Einsatz kam, über Internetkonnektivität verfügt, damit die Messung nicht verzerrt wird. Auch wurde auf dem Access Point die Adresszuweisung via DHCP aktiviert. Des Weiteren wurden die Zenity-Fenster im Programm NetRoam ausgeschaltet, damit die Messungen durch den Benutzer nicht beeinflusst werden.

Um die Messung durchzuführen, wurde das Programm *utime* zum ersten Mal in der Funktion *init* zwischen dem Zenity-Fenster und dem Umsetzen der Priorität aufgerufen.

```
...
#Hinweis an den Benutzer, dass auf WLAN gewechselt wird, wenn einfach das Kabel gezogen wird
if test "$prio" = "0"
then
	#Nur wenn in Datei popupconfig.txt Popup auf aktiv steht
	popup=`cat /etc/netroam/config.txt | grep "landown" | awk -F: '{print $2}'`
	if test "$popup" = "1"
	then
		zenity --info --title="NetRoam" --text="Versuch des Verbindungsaufbaus über
		Wlan"
	fi
fi
./utime > /home/akniwel/messung/utime.txt
#Wenn LAN-Kabel abgezogen, dann setzen der Priorität auf WLAN
echo -n "1" > /etc/netroam/prio.txt
...
```

Der zweite Aufruf des Programms *utime* fand in der Funktion *set_route* zwischen dem Setzen der neuen default-Route und dem Löschen der default-Route mit Metrik 100 statt.

```
...
#Setzen einer neuen Route mit Metric 0
exec $route & 2>/dev/null
./utime >> /home/akniwel/messung/utime.txt
route2=`echo $aroute | sed 's/add/del/'`
#Löschen der alten Route mit Metric 100
exec $route2 &  2>/dev/null
...
```

Die Zeit, die beim Durchlaufen des Programms und der DHCP-Anfrage an den Router verstrichen ist, beträgt ca. 2 Sekunden (1187879803.195377 s – 1187879800.965882 s = 2,034118 s).
Bei der zweiten Messung wurde ebenfalls gemessen wie lange es dauert, bis man zu einem Access Point verbunden ist. Allerdings wurde in diesem Fall nicht die Adressierung über DHCP, sondern die statische Adressierung verwendet. Die Zeit, die sich dabei ergab, beträgt ca. 3 Sekunden (1187883484.883888 s – 1187883481.850592 s = 3,033296 s).
Auf den ersten Blick ist es verwunderlich, dass der Verbindungsaufbau mit einem Access Point der DHCP zur Adressierung verwendet, schneller erfolgt, als mit einem Access Point, der die statische Adressierung vorsieht. Allerdings gibt es mehrere Gründe, an denen der Zeitunterschied liegen kann. Zum einen wird in den Programm NetRoam ein DNS-Server im Internet angepingt, auf den man keinen Einfluss hat. Das heißt, ist das Netz oder der DNS-Server zum Zeitpunkt der Messungen stärker ausgelastet, so ist auch die Laufzeit zwischen den zwei gespeicherten Zeitstempeln größer. Eine weitere Möglichkeit liegt darin, dass der Router und der Computer, mit dem gemessen wurde, schneller eine Verbindung, bei der DHCP zum Einsatz kam, bearbeiten kann, als bei einer Verbindung mit statischer Adressierung. Auch ist es möglich, dass der Computer zum Zeitpunkt der Messungen mit DHCP-Adressierung stärker ausgelastet war als bei der Messung mit statischer Adressierung.
Die Dauer bis eine LAN-Verbindung mit einem Router, der DHCP unterstützt, aufgebaut ist, wurde mit einer weiteren Messung festgehalten. Sobald das Programm NetRoam in der Funktion *init* feststellt, dass die Priorität auf LAN liegt, beginnt die Messung mit dem Programm *utime*. Aus diesem Grund wurde das Programm *utime* in der Funktion *init*

```
...
if test "$lankabel" = "yes" && test "$lan" = "0" && test "$prio" = "0"
then
./utime >> /home/akniwel/messung/utime.txt
if [ -e /tmp/netroam/${interface[m]}.txt ]
then
...
```

und wiederum nach dem Setzen der default-Route aufgerufen. Die Zeit die zwischen diesen beiden Aufrufen verstrich, betrug ca. 2 Sekunden (1187884727.304179 s – 1187884725.346020 s = 1,958159 s). Dieselbe Messung bei statischer Adressierung wurde nicht durchgeführt, da die Zeit davon abhängig ist, wie viele Einträge in der Datei

lan.csv stehen und an wievielter Stelle der richtige Eintrag steht. Jedoch dauert der Verbindungsaufbau mindestens 30 Sekunden, da solange dem Script, welches für die DHCP-Anfrage zuständig ist, per default Zeit gegeben wird, um eine Adresse zu beziehen.
Auch wurde gemessen wie lange es dauert bis das Programm einen Access Point gefunden und eine Internetverbindung über diesen aufgebaut hat. Bei aktivierten DHCP auf dem Access Point dauerte der Verbindungsaufbau ca. 5,3 Sekunden (1187885285.192500 s – 1187885279.805645 s = 5,386855 s). Dabei ist zu beachten, das alleine die DHCP-Anfrage ca. 4 Sekunden gedauert hat (1187885909.053374 s – 1187885905.189284 s = 3,86409 s). Dieselbe Messung wurde bei statischer Adressierung durchgeführt. In diesem Fall betrug die Zeit lediglich ca. 1 Sekunde (1187885700.386172 s – 1187885699.169422 s = 1,21675 s).

6.4 Leistungsbewertung

Wie die Messungen in den Kapiteln 6.3.1 bis 6.3.3 zeigen, dauert der Wechsel von einem Interface zu einem anderen Interface in der Regel nie länger als 10 ms. Dies ist wichtig, da das Programm mit einem SIP-Client zusammenarbeiten können soll, über den es möglich ist, ein VoIP-Telefonat zu führen. Würde der Verbindungsaufbau wesentlich länger dauern, so könnte es passieren, dass bei einem Netzwechsel Datenpakete verloren gingen. Im schlimmsten Fall würde dies zu einem Verbindungsabbruch führen. Bei der Implementierung von NetRoam musste darauf geachtet werden, dass solange noch keine Verbindung über ein neues Interface besteht, die Daten weiterhin über das alte Interface geleitet werden.

7 Installation

7.1 Voraussetzungen für NetRoam

7.1.1 Hardware

Für NetRoam ist keine spezielle Hardware nötig. Lediglich ein LAN-Interface und ein WLAN-Interface werden gebraucht. Damit das Programm auch zwischen WLAN und WLAN, und nicht nur zwischen LAN und WLAN, wechseln kann, wird ein zweites WLAN-Interface benötigt. Dabei wird vorausgesetzt, dass alle Interfaces funktionstüchtig und alle zugehörigen Treiber installiert sind.

7.1.2 Software

Als Betriebssystem für NetRoam wird Kubuntu 7.04 „Feisty Fawn" vorausgesetzt. Dabei greift das Programm auf bereits standardmäßig installierte Programme, wie beispielsweise wpa_supplicant zurück. Sollte wpa_supplicant wider Erwartend nicht installiert sein, so kann dieses Programm mit dem Befehl

```
apt-get install wpasupplicant
```

nachinstalliert werden.

Des Weiteren muss das Programm Zenity für das Anzeigen der Pop-Up-Meldungen installiert werden. Dies geschieht mit dem Befehl

```
apt-get install zenity
```

Weitere Software die nachinstalliert werden müsste, ist für NetRoam nicht nötig. Um Konflikte zwischen NetRoam und dem standardmäßig für die Verwaltung der Verbindungen zuständigen Programm KNetworkManager, vorzubeugen, sollte dieser während des Gebrauchs von NetRoam deaktiviert werden.

7.2 Voraussetzungen für das Webfrontend

7.2.1 Software

Bei der Programmierung des Webfrontends wurde sowohl auf die Scriptsprache JavaScript als auch auf die Scriptsprache PHP zurückgegriffen. Aus diesem Grund wird ein Apache-Webserver auf dem Rechner benötigt, auf dem das Webfrontend laufen soll. Um diesen nutzen zu können, müssen die Pakete apache2, apache2.2-common und apache2-doc mit dem Befehl *apt-get install* in einem Konsolenfenster installiert werden. Nach der Installation dieser Pakete hat man einen lauffähigen Apache-Webserver, allerdings noch ohne PHP Unterstützung. Um PHP nutzen zu können, müssen zusätzlich die Pakete php5 und libapache2-mod-php5 ebenfalls mit dem Befehl *apt-get install* installiert werden.

Nach erfolgreicher Installation aller benötigten Pakete muss der Apache-Webserver mit dem Befehl

```
sudo /etc/init.d/apache2 start
```

gestartet werden. Ebenfalls kann der Server mit dem Befehl

```
sudo /etc/init.d/apache2 stop
```

gestoppt bzw. mit dem Befehl

```
sudo /etc/init.d/apache2 restart
```

neu gestartet werden.

Da auch, wie oben schon geschrieben, JavaScript bei der Implementierung zum Einsatz kam, muss ebenfalls sichergestellt sein, dass dies im jeweiligen Browser aktiviert ist.

7.3 Installation von NetRoam

Das Programm NetRoam und das Webfrontend befinden sich gepackt in dem Archiv *NetRoam.tar.gz*. Um dieses Archiv entpacken zu können, müssen in der Konsole die Befehle

```
gunzip NetRoam.tar.gz
```

und

```
tar -xvf NetRoam.tar
```

eingegeben werden. Dabei ist zu beachten, dass die Dateien in den Ordner kopiert werden, in dem man sich befindet. Nach erfolgreichem Entpacken, muss das Installationsscript *install* (siehe Kapitel 13.1) gestartet werden. Dieses sorgt dafür, dass die Dateien an die richtigen Stellen kopiert werden. Für das Deinstallieren von NetRoam gibt es ebenfalls ein Script, welches *deinstall* (siehe Kapitel 13.5) heißt.

Das Programm kann nach erfolgreicher Installation mit den Befehlen

```
sudo /etc/init.d/netroam <start|stop|restart>
```

kontrolliert werden. Damit NetRoam nach dem Starten des Systems ebenfalls startet, muss das Script *netroamstart* (siehe Kapitel 13.3) manuell in den Ordner */.kde/Autostart/* kopiert werden.

8 Bedienung

8.1 NetRoam

Wie schon in Kapitel 7.3 beschrieben kann das Programm NetRoam mit dem Befehl

```
sudo /etc/init.d/netroam <start|stop|restart>
```

kontrolliert werden. Die eigentliche Bedienung des Programms, wie beispielsweise das Hinzufügen und Löschen von Access Points bzw. Routern, findet über das Webfrontend statt, welches im nachfolgenden Kapitel erläutert wird. Neben diesem Webfrontend bietet das Programm allerdings noch die Möglichkeit, mit Hilfe von Zenity-Fenstern gesteuert zu werden. Diese Pop-Ups erscheinen immer bei verschiedenen Ereignissen und können in zwei Gruppen aufgeteilt werden. Die erste Gruppe sind die Informationsfenster, die den Benutzer nur über Ereignisse informieren sollen und wie in Abbildung 27 aussehen können.

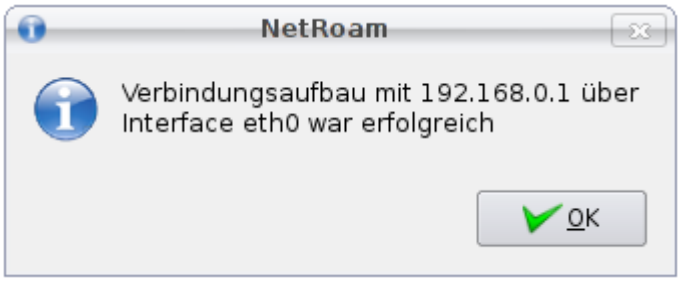

Abbildung 27: Zenity-Informationsfenster

Diese Informationsfenster erscheinen beispielsweise beim Versuch des Verbindungsaufbaus oder, wie hier zu sehen ist, wenn der Verbindungsaufbau erfolgreich verlaufen ist. Sie müssen nur mit einem Klick auf den OK-Button bestätigt werden.

Die zweite Gruppe beinhaltet die Fragefenster. Sie treten immer dann auf, wenn der Benutzer eine Entscheidung treffen muss. Dies kann zum Beispiel der Fall sein, wenn der Benutzer das LAN-Kabel eingesteckt hat und NetRoam mit einem Pop-Up nachfragt, ob die Verbindung auf das LAN-Kabel gewechselt werden soll. In diesem Fall gibt es die Möglichkeit mit einem Klick auf den OK-Button dies zu bestätigen oder aber es mit einem Klick auf den CANCEL-Button zu verhindern. Diese Art der Zenity-Fenster ist in Abbildung 28 dargestellt.

Abbildung 28: Zenity-Fragefenster

8.2 Webfrontend

Das Webfrontend wurde erstellt, damit die Möglichkeit gegeben ist, bequem Eingaben durchzuführen oder Ausgaben des Programms zu betrachten.

Gestartet wird das Webfrontend indem man den Ordner */var/www/Webfrontend/* öffnet und dort die Datei *Netroam.php* startet. Für den weiteren Gebrauch ist es ratsam, sich ein Lesezeichen dieser Seite zu erstellen.

Nach dem Aufrufen dieser Datei erscheint das Startfenster, welches in drei Bereiche unterteilt ist. Zum einen gibt es das Menü mit dem man die verschiedenen Seiten aufrufen kann. Zum anderen gibt es den Statusbereich. In diesem Bereich stehen Informationen über die einzelnen Interfaces zur Verfügung. Der dritte Bereich ist für das Umsetzen der Priorität zuständig und ist, wie das Menü, immer sichtbar, egal welche Unterseite aufgerufen wird. Abbildung 29 zeigt dieses Startfenster.

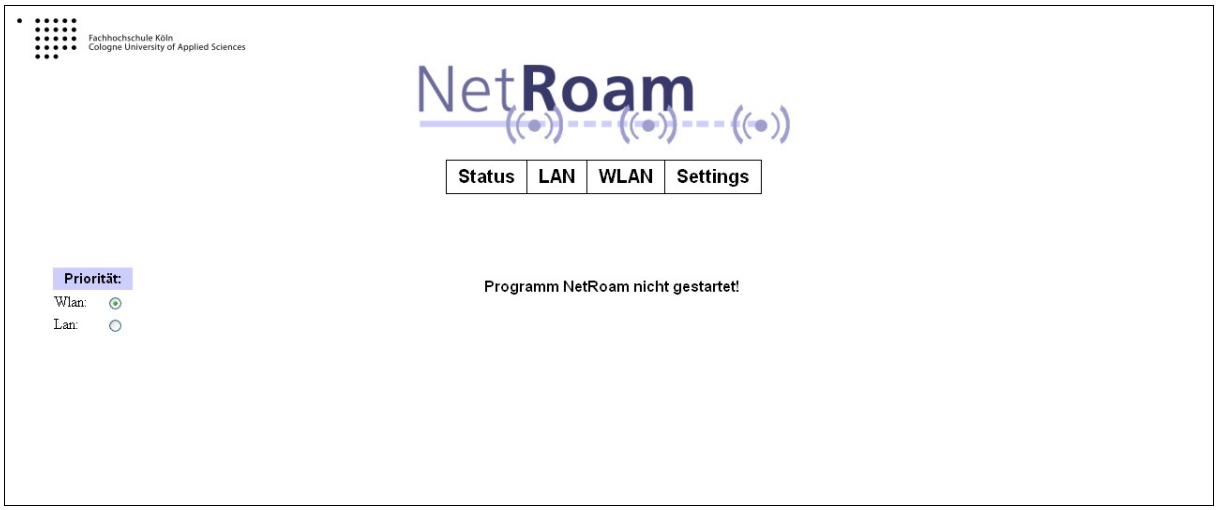

Abbildung 29: Webfrontend – Startfenster

In dieser Abbildung weist der Statusbereich darauf hin, dass das Programm NetRoam nicht

gestartet wurde. Wenn NetRoam jedoch läuft bietet dieser Bereich, wie schon erwähnt, einen Überblick über die verschiedenen Interfaces. Das Webfrontend führt über das LAN-Interface Informationen, wenn die Priorität auf WLAN liegt (siehe Abbildung 30) und wenn

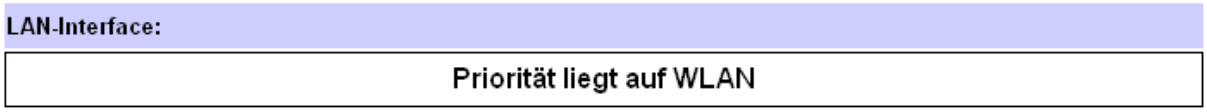

Abbildung 30: Webfrontend - LAN-Interface bei Priorität auf WLAN

über dieses Interface eine aktive Verbindung besteht (siehe Abbildung 31).

Abbildung 31: Webfrontend - LAN-Interface bei aktiver Verbindung

Für die WLAN-Interfaces werden ebenfalls Informationen dargestellt. Zum einen werden die gefundenen und auch bekannten Access Points aufgelistet, zum anderen werden alle

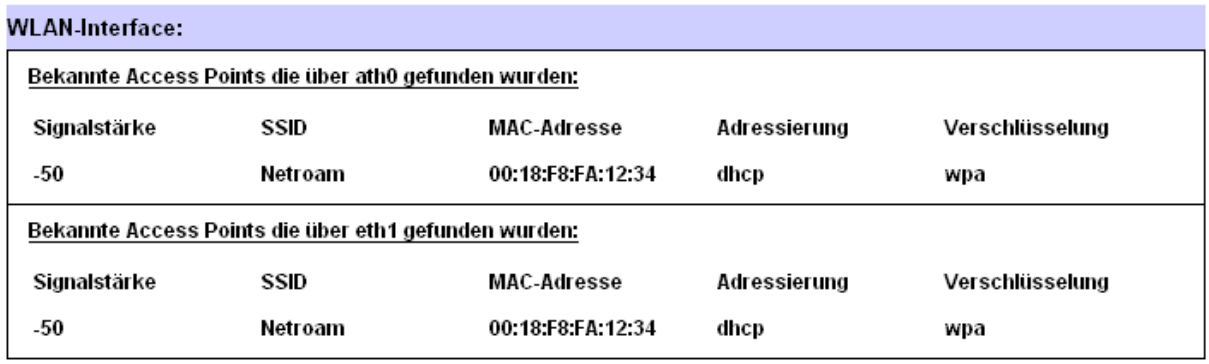

Abbildung 32: Webfrontend - bekannte und gefundene Access Points

gefundenen Access Points dargestellt (siehe Abbildung 33). Dabei kann es durchaus sein, dass zwei WLAN-Interfaces unterschiedliche Access Points finden. Bei den gefundenen Access Points besteht des Weiteren die Möglichkeit, diese mit einem Klick auf die Diskette zu speichern. Nach diesem Klick werden die verfügbaren Informationen über den Access Point in das Formular auf der Seite *Add Access Point* eingetragen.

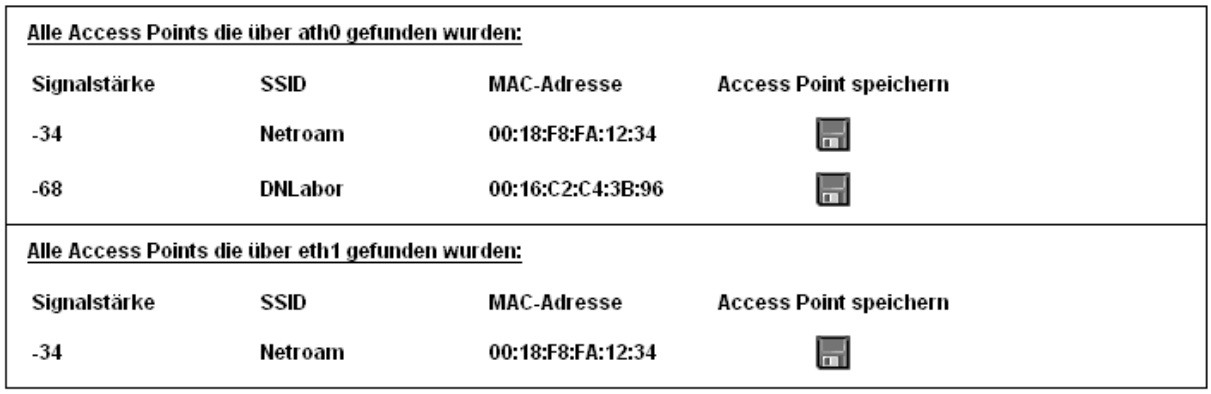

Abbildung 33: Webfrontend - alle gefundenen Access Points

Wenn über das LAN-Interface und über die WLAN-Interfaces kein bekanntes Netzwerk bzw. kein bekannter Access Point gefunden wurde, wird dies wie in Abbildung 34 dargestellt.

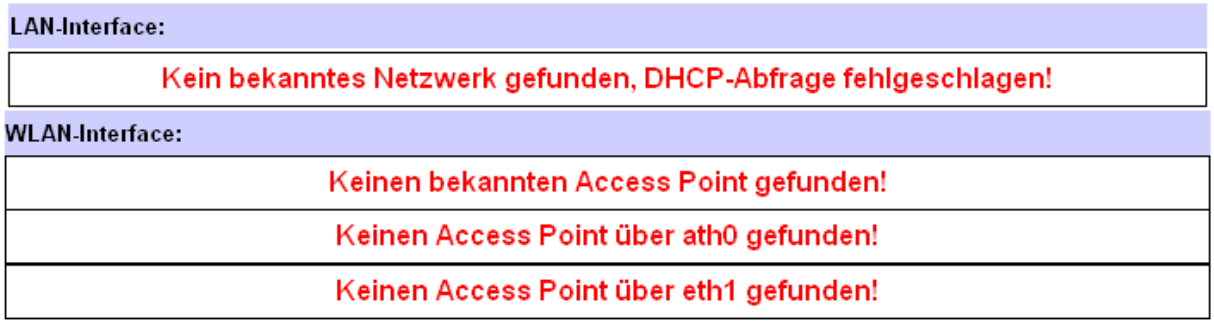

Abbildung 34: Webfrontend - Kein Netz bzw. Access Point wurde gefunden

Um neue Router in die Datei *wlan.csv* eintragen zu können, muss im Menü die Seite *LAN* ➜ *Add Router* aufgerufen werden.

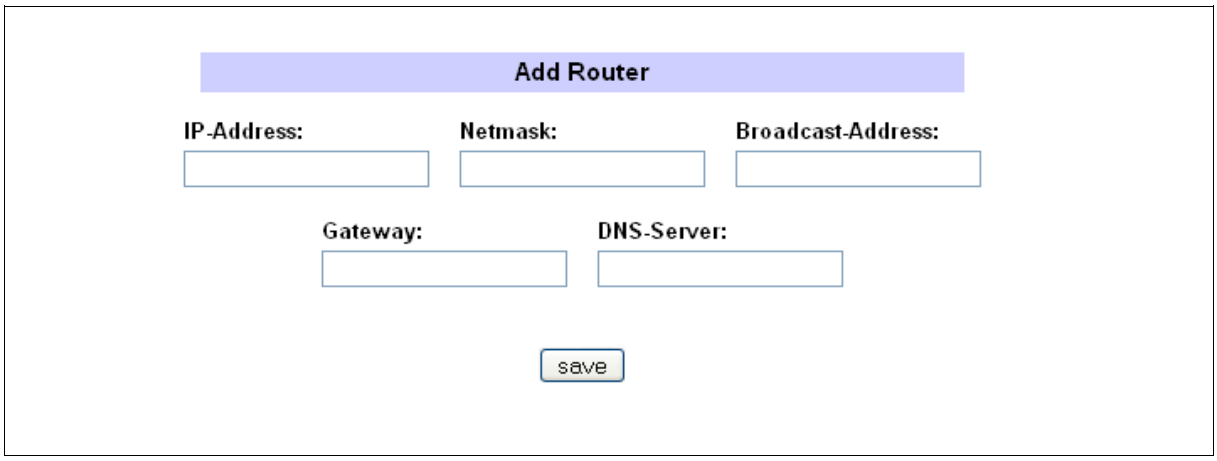

Abbildung 35: Webfrontend – Seite Add Router

Nach erfolgreichem Eintragen der Daten erscheint die Seite, in der alle eingetragenen Router stehen. Diese Seite kann man ebenfalls über das Menü erreichen (*LAN* ➜ *Router*) und sie ist in Abbildung 36 dargestellt.

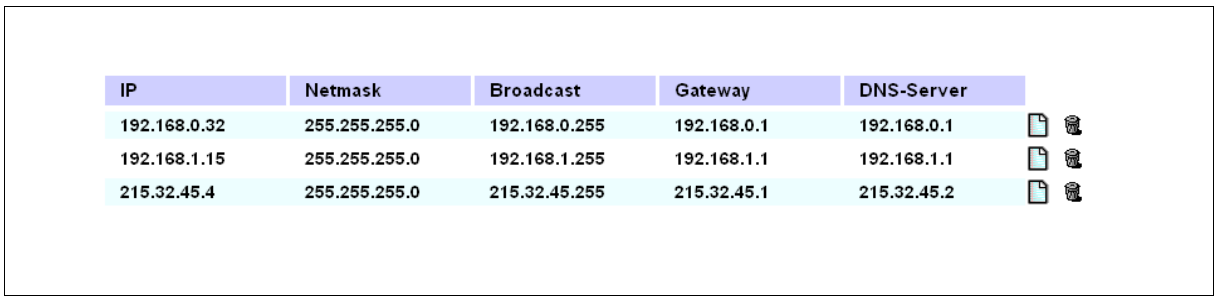

Abbildung 36: Webfrontend - Seite Router

Um einen Eintrag ändern zu können, kann auf das Dokument in der jeweiligen Zeile geklickt werden, was zur Wirkung hat, dass wieder das Formular auf der Seite *Add Route* aufgerufen wird und dort die Änderungen an dem Eintrag vorgenommen werden können. Ebenfalls ist es möglich den Eintrag aus der Datei *lan.csv* zu löschen. Dafür muss auf den Mülleimer in der jeweiligen Zeile geklickt werden.

Genau wie für die Datei *lan.csv,* in der die Router mit Hilfe der beiden Seiten eingetragen und betrachten werden können, gibt es für die Datei *wlan.csv* ebenfalls zwei Seiten. Die erste Seite ist für das Eintragen der Access Points vorgesehen und sieht wie in Abbildung 37 dargestellt ist, aus.

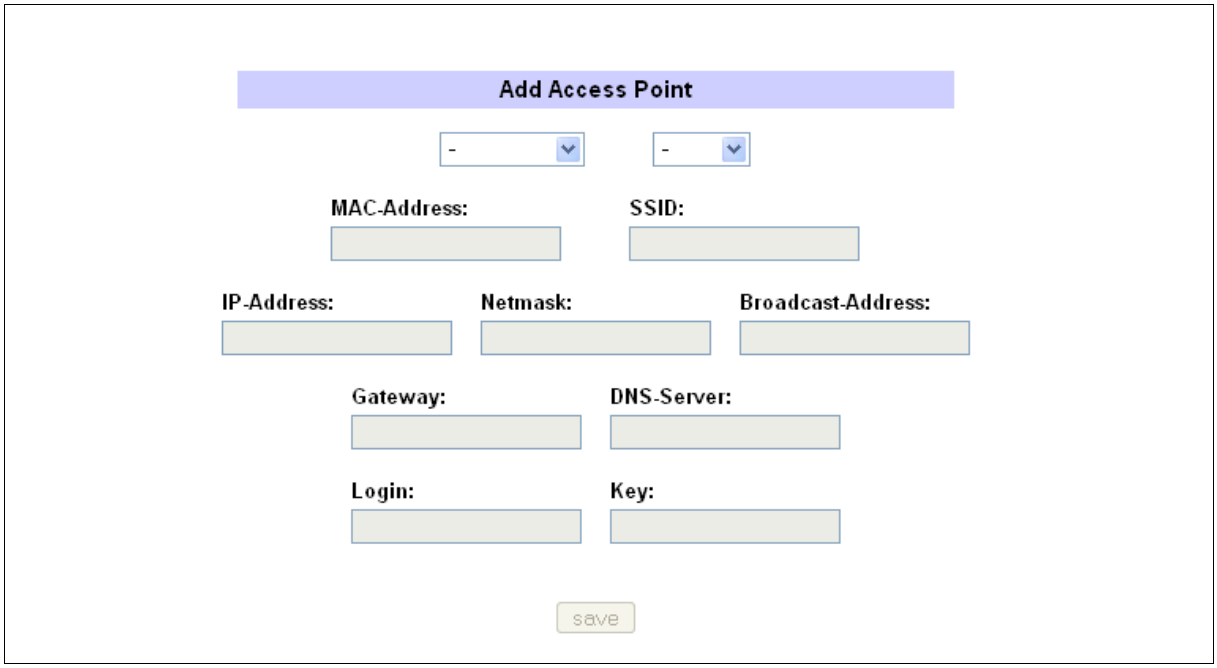

Abbildung 37: Webfrontend - Seite Add Access Point

Dabei ist zu beachten, dass erst Daten eingegeben werden könne, wenn in einem Pulldown-Menü ein Eintrag ausgewählt worden ist. Das erste Menü bestimmt die Verschlüsselung und gibt nach Auswahl die Felder *MAC-Adresse*, *SSID*, *Login* und *Key* frei. Der Key muss bei WPA und WPA2 im Klartext eingegeben werden, wohingegen der Key bei WEP in hexadezimaler Form eingetragen werden muss. Nach Auswahl der Adressierungsart, welche durch das zweite Pulldown-Menü eingestellt werden kann, werden bei statischer Adressierung die restlichen Eingabefelder freigegeben. Nur wenn Einstellungen in beiden Pulldown-Menüs durchgeführt wurden, kann der Access Point mit einem Klick auf den save-Button gespeichert werden. Nach erfolgreichem Eintragen des Access Points wird die Seite *Access Point* aufgerufen, die ebenfalls über das Menü (*WLAN* ➜ *Access Point*) aufgerufen werden kann. Auf dieser Seite sind alle in der Datei *wlan.csv* eingetragenen Access Points aufgeführt.

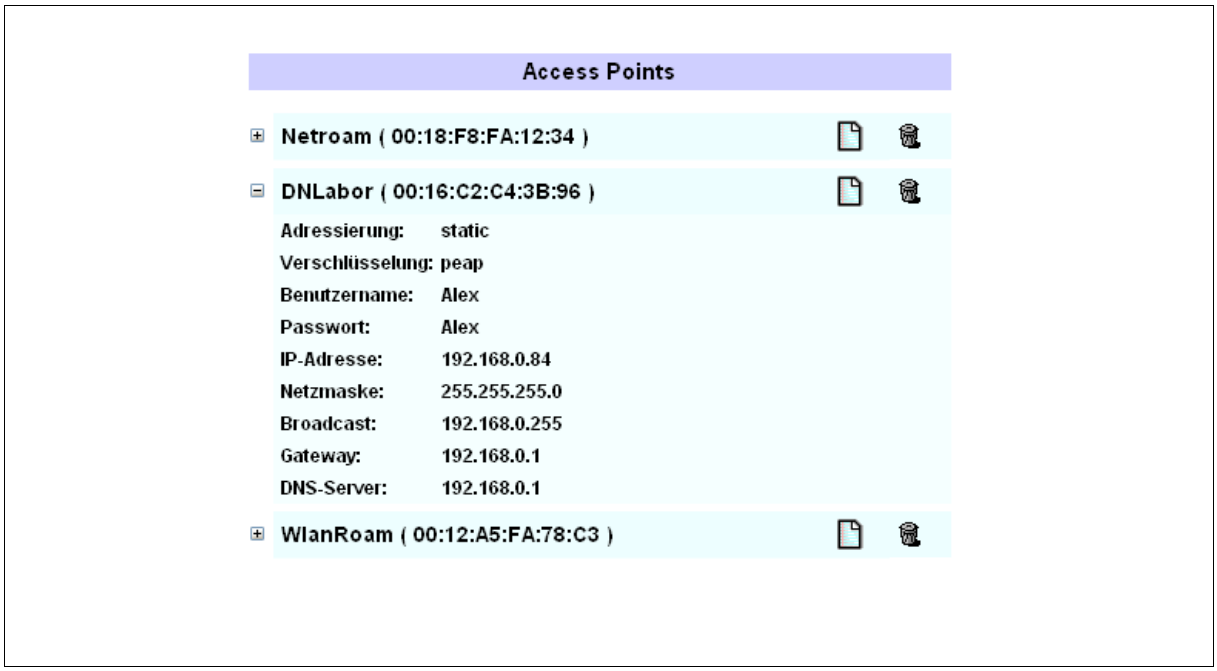

Abbildung 38: Webfrontend - Seite Access Points

Dabei ist zu beachten, dass am Anfang nur die SSID und die MAC-Adresse des jeweiligen Access Point sichtbar sind. Erst ein Klick auf den Plus-Button lässt weitere Daten erscheinen. Dies dient zum einen der Übersicht und zum anderen sind so nicht direkt die Passwörter sichtbar. Die Access Points können ebenfalls mit einem Klick auf das Dokument editiert und mit einem Klick auf den Mülleimer-Button aus der Datei *wlan.csv* gelöscht werden.

Eine weitere Seite die über den Menüpunkt WLAN aufgerufen werden kann, ist die Seite *Blacklist*. Auf dieser werden alle in der Datei *blacklist.txt* stehenden Access Points aufgelistet. Diese Datei aktualisiert sich im Gegensatz zur Statusseite, die sekündlich aktualisiert wird, alle 3 Sekunden. So ist ein durch das Programm NetRoam eingetragener Access Point direkt zu erkennen. In Abbildung 39 ist die Seite dargestellt.

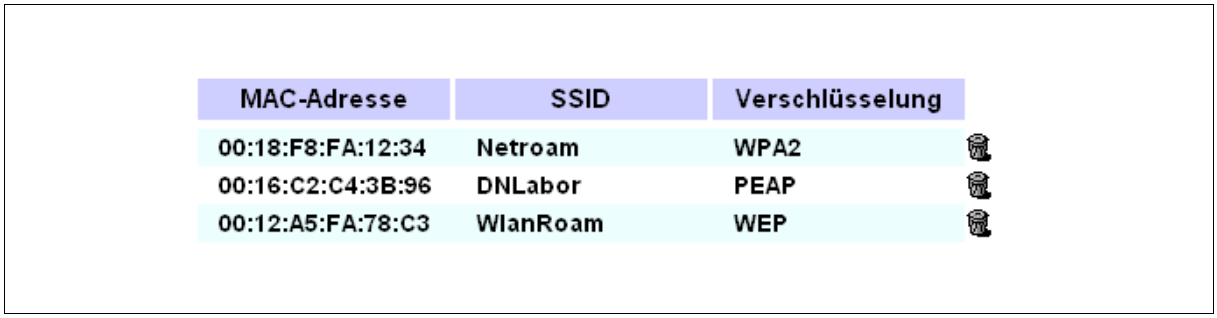

Abbildung 39: Webfrontend - Seite Blacklist

Auch hier kann ein Eintrag mit einem Klick auf den Mülleimer-Button gelöscht werden.

Die letzte Seite die über das Menü aufgerufen werden kann, ist die Seite Settings. Auf dieser Seite kann neben Einstellungen zu den Zenity-Fenstern auch eingestellt werden, ab welcher Signalstärke der Access Point gewechselt werden soll. Standardmäßig ist dies auf - 40 dB eingestellt. Ebenfalls ist es möglich einzustellen, wie viele Versuche durchgeführt werden sollen, bevor ein Access Point auf die Blacklist gesetzt wird. Standardmäßig sind hier 2 Versuche vorgesehen und die Zenity-Fenster sind standardmäßig alle auf aktiv gestellt. Der Aufbau der Seite Settings ist in Abbildung 40 dargestellt.

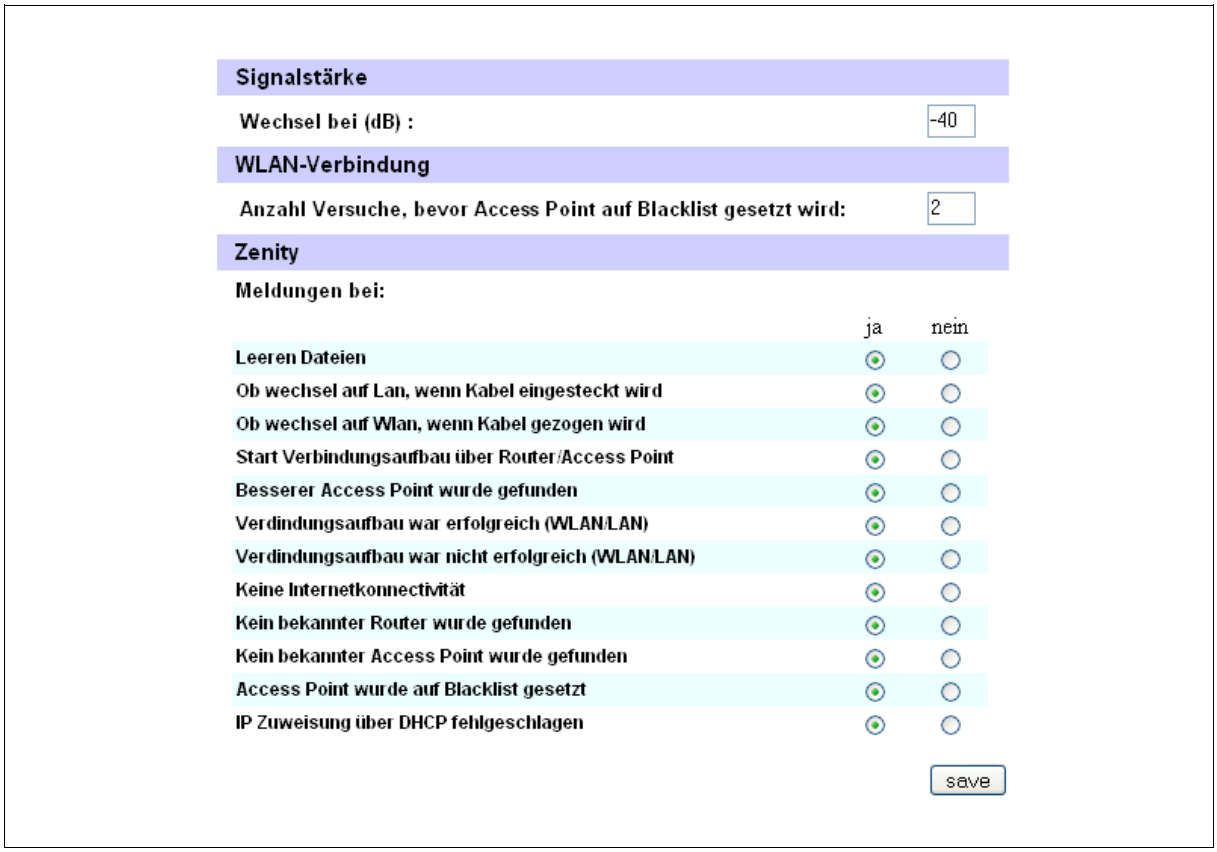

Abbildung 40: Webfrontend - Seite Settings

9 Schlussbetrachtung

Mit dem Programm NetRoam und dem Webfrontend sind die anfänglich definierten Ziele erreicht worden. Es ist mit Hilfe des Programms möglich, zwischen verschiedenen Netzwerken zu wechseln, ohne dass die Verbindung abbricht. Dabei ist wie gewollt unerheblich, ob es sich bei dem Netzwerk um ein drahtgebundenes oder drahtloses Netzwerk handelt. Auch eine Prioritätssteuerung wurde eingeführt, die entweder das LAN-Interface oder das WLAN-Interface bevorzugt.

Des Weiteren wurde an verschiedenen Stellen im Quellcode von NetRoam eine Basis für ein späteres Verwenden des Programms mit einem SIP-Client geschaffen. Dafür müssen nur die dafür nötigen Bereiche im Quellcode auskommentiert werden.

10 Abbildungsverzeichnis

Abbildung 1: Ethernet-Frame frei nach [5]....14
Abbildung 2: Aufbau der wpa_supplicant Konfigurationsdatei....18
Abbildung 3: IPv4-Header....19
Abbildung 4: IPv6-Header....20
Abbildung 5: DHCP-Paket....22
Abbildung 6: Routingtabelle....25
Abbildung 7: Ablaufplan des Hauptprogramms....30
Abbildung 8: Ablaufplan der Funktion init....33
Abbildung 9: Aufbau der Datei <Interface>.txt für ein WLAN-Interface bei bestehender Verbindung....35
Abbildung 10: Aufbau der Datei <Interface>.txt für ein WLAN-Interface bei keiner aktiven Verbindung....36
Abbildung 11: Ablaufplan der Funktion wlan....36
Abbildung 12: Ablaufplan Funktion lan....38
Abbildung 13: Aufbau der Datei <Interface>.txt für ein LAN-Interface bei aktiver Verbindung....39
Abbildung 14: Ablaufplan der Funktion wlan2....46
Abbildung 15: Ablaufplan Funktion set_route....49
Abbildung 16: Aufbau der Datei lan.csv....53
Abbildung 17: Aufbau der Datei wlan.csv....54
Abbildung 18: Aufbau der Datei config.txt....54
Abbildung 19: Aufbau der Datei blacklist.txt....55
Abbildung 20: Aufbau der Datei infowlan<Interface>.txt bei aktiver Verbindung....56
Abbildung 21: Aufbau der Datei infowlan<Interface>.txt bei keiner aktiven Verbindung...56
Abbildung 22: Aufbau der Datei foundwlan<Interface>.txt bei keiner aktiven Verbindung57
Abbildung 23: Aufbau der Messumgebung....60
Abbildung 24: Wechsel von LAN auf WLAN....61
Abbildung 25: Wechsel von WLAN auf WLAN....62
Abbildung 26: Wechsel von WLAN auf LAN....63
Abbildung 27: Zenity-Informationsfenster....70
Abbildung 28: Zenity-Fragefenster....71
Abbildung 29: Webfrontend – Startfenster....71
Abbildung 30: Webfrontend - LAN-Interface bei Priorität auf WLAN....72
Abbildung 31: Webfrontend - LAN-Interface bei aktiver Verbindung....72
Abbildung 32: Webfrontend - bekannte und gefundene Access Points....72
Abbildung 33: Webfrontend - alle gefundenen Access Points....73
Abbildung 34: Webfrontend - Kein Netz bzw. Access Point wurde gefunden....73
Abbildung 35: Webfrontend – Seite Add Router....73
Abbildung 36: Webfrontend - Seite Router....74
Abbildung 37: Webfrontend - Seite Add Access Point....74
Abbildung 38: Webfrontend - Seite Access Points....75
Abbildung 39: Webfrontend - Seite Blacklist....76
Abbildung 40: Webfrontend - Seite Settings....77

11 Tabellenverzeichnis

Tabelle 1: Die wichtigsten grep-Kommandos....10
Tabelle 2: Mögliche Zenity-Arten....12
Tabelle 3: Mögliche DHCP-Nachrichten[6]....22
Tabelle 4: Flags [7]....26
Tabelle 5: Mögliche Zahlenkürzel bei WLAN-Interfaces....32
Tabelle 6: Mögliche Zahlenkürzel bei LAN-Interfaces....32

12 Literaturverzeichnis

12.1 Printmedien

[1] Wolf, Jürgen: „Shell-Programmierung“, Galileo Press, 2005, ISBN 3-89842-683-1

[2] Fischer, Marcus: „Ubuntu GNU/Linux“, Galileo Press, 2007, ISBN 978-3-89842-848-4

[3] Wenz, Christian/Hauser, Tobias: „HTML“, Markt+Technik, 2003, ISBN 3-8272-68880-X

[4] Wenz, Christian: „JavaScript und AJAX“, Galileo Press, 2007, ISBN 978-3-89842-859-0

12.2 Internetquellen

[5] http://dnserver.nt.fh-koeln.de/grebe/DN/dn1-02-datalink-lan-a.pdf

[6] http://www.netzmafia.de/skripten/netze/netz9.html

[7] https://suxer.de/docs/linux/manpages/route.html

[8] http://www.bsi.bund.de/gshb/deutsch/m/m02276.htm

[9] http://www.linux.de

[10] http://www.wikipedia.org

[11] http://fedorawiki.de/index.php/Zenity

[12] http://www.informationsarchiv.net/statisch/wlan

[13] http://www.tomsnetworking.de/content/reports/j2003a/report.wlan.sicherheit

[14] http://www.netplanet.org

[15] http://www.tlab.ch/praktika/serieIII/c16/theorie.html

[16] http://www.foo.fh-furtwangen.de/~link/studium

13 Anhang

13.1 Script install

```
#!/bin/bash

echo "Willkommen bei NetRoam!"

if [ ! -d /etc/netroam/ ]
      then
      #Verzeichnis für die Konfigurtionsdateien (wlan.csv, lan.csv,...)
      echo "Erstelle Verzeichnis /etc/netroam/"
      mkdir /etc/netroam/
fi

if [ ! -d /usr/bin/netroam/ ]
then
      #Verzeichnis für die Skriptdateien
      echo "Erstelle Verzeichnis /usr/bin/netroam/"
      mkdir /usr/bin/netroam/
fi

echo "Setze Zugriffsrechte fuer /etc/netroam/"
chmod 755 /etc/netroam/

echo "Setze Zugriffsrechte fuer /usr/bin/netroam/"
chmod 755 /usr/bin/netroam/

echo "Kopiere Dateien nach /usr/bin/netroam/"
cp start /usr/bin/netroam/

if [ ! -d /usr/bin/netroam/save/ ]
      then
      mkdir /usr/bin/netroam/save/
fi

echo "Sichern der Datei /etc/network/interfaces nach /usr/bin/netroam/save/"
cp /etc/network/interfaces /usr/bin/netroam/save/

echo "Ersetzen der Datei /etc/network/interfaces"
cp interfaces /etc/network/interfaces

echo "Sichern der Datei /sbin/dhclient-script nach /usr/bin/netroam/save/"
cp /sbin/dhclient-script /usr/bin/netroam/save/

echo "Ersetzen der Datei /sbin/dhclient-script"
cp dhclient-script  /sbin/dhclient-script

if [ ! -e /etc/netroam/wlan.csv ]
then
      echo "Erstelle Datei wlan.csv"
      touch /etc/netroam/wlan.csv
fi

if [ ! -e /etc/netroam/lan.csv ]
then
      echo "Erstelle Datei lan.csv"
      touch /etc/netroam/lan.csv
fi

if [ ! -e /etc/netroam/blacklist.txt ]
then
```

```
      echo "Erstelle Datei blacklist.txt"
      touch /etc/netroam/blacklist.txt
fi

echo "Kopiere Dateien nach /etc/netroam/"
cp ./prio.txt /etc/netroam/
cp ./config.txt /etc/netroam/

chmod 666 /etc/netroam/wlan.csv
chmod 666 /etc/netroam/lan.csv
chmod 666 /etc/netroam/blacklist.txt
chmod 666 /etc/netroam/prio.txt
chmod 666 /etc/netroam/config.txt

echo "Kopiere Datei netroam nach /etc/init.d/"
cp netroam /etc/init.d/netroam

#Ordner für Webfrontend erstellen, falls noch nicht vorhanden
if [ ! -d /var/www ]
then
      mkdir /var/www/
fi

echo "Kopiere Dateien für Webfrontend nach /var/www/"
cp -r Webfrontend /var/www/

echo "Bitte tragen Sie die bekannten AccessPoints und die bekannten Router mit
Hilfe des Webfrontends ein, dieses finen Sie in dem Ordner /var/www/Webfrontend
unter der Datei Netroam.php"

echo "Sie können das Programm unter /etc/init.d/netroam mit dem Befehl 'start'
starten, ab dem nächsten Neustart, sollte das Skript automatischstarten,
kopieren Sie dafuer die Datei netroamstart in den Ordner /.kde/Autostart in ihrem
Home Verzeichnis"
```

13.2 Script netroam

```
#!/bin/bash

function start(){
      /usr/bin/netroam/start
}

function stop(){
      killall start 2>/dev/null
      killall lan 2>/dev/null
      killall wlan 2>/dev/null
      killall connection 2>/dev/null
      killall wpa_supplicant 2>/dev/null
}

if test "$1" = ""
then
      start
else
      case $1 in
      start)
            start
      ;;
      stop)
            stop
      ;;
```

```
      restart)
            stop
            start
      ;;
            esac
      fi
```

13.3 Script netroamstart

```
#!/bin/bash

xhost +
kdesu /etc/init.d/netroam start
```

13.4 Script start

Das Script *start* kann in dem Ordner *./Programm_Netroam/* auf der beiliegenden CD eingesehen werden.

13.5 Script deinstall

```
#!/bin/bash

echo "Netroam wird beendet"
#Beenden aller laufenden Prozesse
killall start 2>/dev/null
killall lan 2>/dev/null
killall wlan 2>/dev/null
killall connection 2>/dev/null
killall wpa_supplicant 2>/dev/null

echo "Stelle die alte Datei /etc/network/interfaces wieder her"
cp /usr/bin/netroam/save/interfaces /etc/network/interfaces

echo "Stelle die alte Datei /sbin/dhclient-script wieder her"
cp /usr/bin/netroam/save/dhclient-script /sbin/dhclient-script

echo "Löschen der Datei /etc/init.d/netroam"
rm /etc/init.d/netroam 2>/dev/null

echo "Löschen der Dateien in /usr/bin/netroam/"
rm -r /usr/bin/netroam/ 2>/dev/null
i=""
until( test "$i" = "j" || test "$i" = "n" )
do
      echo "Sollen auch die Konfigurationsdateien gelöscht werden?[j/n]"
      read i
      if test "$i" = "j"
      then
            echo "Löschen der Dateien in /etc/netroam/"
            rm -r /etc/netroam
      fi
done

j=""
until( test "$j" = "j" || test "$j" = "n" )
```

```
do
      echo "Sollen auch das Webfrontend gelöscht werden?[j/n]"
      read j
      if test "$j" = "j"
      then
            echo "Löschen der Dateien in /etc/netroam/"
            rm -r /var/www/Webfrontend
      fi
done

echo "Bitte löschen Sie noch die Datei netroamstart in ihrem Home-Order falls
Sie sie dorthin kopiert haben sollten"
```

13.6 Programm utime

```
#include <stdio.h>
#include <sys/time.h>

int main()
{
      struct timeval timechen;
      struct timezone tzp;
      gettimeofday(&timechen,&tzp);
      printf("%d.%.6d\n",timechen.tv_sec,timechen.tv_usec);
      return 0;
}
```

13.7 Beispielkonfiguration der wpa_supplicant.conf für WPA

```
network={
      ssid="NetRoam"
      scan_ssid=1
      proto=WPA
      key_mgmt=WPA-PSK
      pairwise=TKIP
      group=TKIP
      #psk="passwort"
      psk=a99f007c97c613b3c78d14eef50c475453ff21cc5167b0f52db605670a360850
}
```

13.8 Beispielkonfiguration der wpa_supplicant.conf für WPA2

```
network={
      ssid="Netroam"
      scan_ssid=1
      proto=RSN WPA
      key_mgmt=WPA-PSK
      pairwise=CCMP TKIP
      group=CCMP TKIP
      #psk="passwort"
      psk=9fa283f6bd139e4a3e7eefd22b6d9049caefcc2cb1cca458e6d33d64b2523f5a
}
```

13.9 Beispielkonfiguration der wpa_supplicant.conf für WPA2-RADIUS

```
network={
      ssid="NetRoam"
      proto=WPA2
      scan_ssid=1
      key_mgmt=WPA-EAP
      eap=PEAP
      identity="alex" #Benutzername
      password="alex" #Passwort
      phase2="auth=MSCHAPV2"
}
```