Multi-connectivity in Android-based infrastructure-less networks

Bachelor Thesis

by

Taiyou Thomas Teramachi

born in

Milton Keynes, GB

submitted to

Technology of Social Networks Lab

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

Heinrich-Heine-Universität Düsseldorf

October 2016

Supervisor:

Raphael Bialon, M. Sc.

Abstract

Data transfer over radio networks by the Standard of 802.11 (WLAN) is getting more significant and therefore ways for optimizing this concept are to be found. In order to lower network traffic and simultaneously increase connectivity different options have to be investigated. One way of doing this is by establishing a multi-connection to multiple networks operating on different wavebands by using a relay device as a bridge. In this thesis this idea is illustrated and analyzed by using a mobile device running with Android as an interstation between a 2.4 GHz and a 5 GHz network. This device is going to capture signed data and for that it needs an additional external WLAN card and the necessary requirements have to be met in order to realize this setup. The sending and the receiving end were implemented using Java. Various aspects of this structure such as the ideal transmission rate, the travel time and the effect of different packet sizes have been analyzed. Additionally, a real life example in the form of an image transfer has been conducted to show that all kinds of data and not only sensory data can be relayed. The results proved the functional capability of this setup for relaying data, but room for improvement was limited caused by the used network card chip only supporting the IEEE 802.11 WLAN standards b and g. Overall, the setup showed to be scientifically valuable to further investigate.

Contents

List of Figures ix

List of Tables ix

1 Introduction 1

1.1 Motivation 1

1.2 Structure of Thesis 2

2 Related Work 5

2.1 Multi-Cell MIMO Cooperative Networks 5

2.2 Distributed MIMO Systems 6

2.3 Triple MIMO Communication 6

2.4 Device-to-Device Communication 6

2.5 Optimizing MIMO Relay Systems 7

2.6 Relay System on a Cargo Ship 7

2.7 Android External Wifi 7

2.8 Summary 8

3 Fundamentals 9

3.1 Terminology 9

3.1.1 Monitor Mode 9

3.1.2 PCAP 10

3.1.3 Magic Number 10

3.1.4 Interleaving 10

3.2 Standards of 802.11 WLAN 11

3.3 Internet Protocol Version 4 Header 11

3.4 User Datagram Protocol Header 12
3.4.1 UDP Checksum 12

4 Implementation **13**
4.1 General Setup 13
4.2 MyServer 14
4.2.1 Mechanics 14
4.2.2 Magic Number Header 16
4.3 Android External Wifi 16
4.3.1 IP Filter 17
4.3.2 Magic Number Processing 17
4.3.3 Relaying Process 18
4.4 MyReceiver 18
4.5 Timestamp Extension 19
4.5.1 MyServer 20
4.5.2 Android External Wifi 20
4.5.3 MyReceiver 20
4.6 Image Transfer Extension 21
4.6.1 MyServer 21
4.6.2 MyReceiverV2 22
4.7 Limitations and Known Problems 22
4.7.1 Limits 22
4.7.2 Channels 23
4.7.3 Screen Lock 23
4.8 Summary 23

5 Tests and Results **25**
5.1 General Test Requirements 25
5.2 Sources for Environmental Interference 26
5.2.1 Surrounding Traffic 26
5.2.2 Ideal Channel 28
5.3 Optimum Transmission Rate 30
5.4 Corrupt Packets 33
5.5 Varying Packet Sizes 33

5.6 Testing UDP as a Relay 35
5.7 Relay Time 35
5.8 Real Life Example 37
5.9 Summary 39

6 Summary **41**
6.1 Conclusion 41
6.2 Future Work 42
6.2.1 Relaying Optimization Functions 42
6.2.2 Transmission Rate Deviation 43
6.2.3 Workerpool 43
6.2.4 Capturing Behaviour 43
6.2.5 Packet Size Anomaly 44
6.2.6 Superior Hardware 44

Bibliography **45**

Abbreviations **49**

Glossary **51**

List of Figures

4.1 Planned network setup 14
4.2 The MyServer GUI 15
4.3 The Magic Number header 16
4.4 The MyReceiver GUI 19
4.5 Image assemble controls 22

5.1 Used network setup 26

List of Tables

5.1 Other network traffic around testing environment 27
5.2 Traffic on different channels at Position 1 29
5.3 Sent, captured and received frames 31
5.4 Actual transmission and capture rate 32
5.5 Percentage captured with different packet sizes 34
5.6 Capture rate and relay percentage with UDP 35
5.7 Relay Time 36
5.8 Cases in which all different ID's were received 38

Chapter 1

Introduction

1.1 Motivation

Nowadays almost every electronic device, sensor or service needs to transmit some kind of data. Through this development mobile data networks and Wireless Local Area Network (WLAN) by Institute of Electrical and Electronics Engineers (IEEE) standard 802.11 are getting more and more important for mobile communication, access to remote data and the internet. Due to this fact, the importance of availability and security as well as the underlying technology of the used network rises.
As the usage of networks increases, so does the strain on it, which makes it vulnerable to breakdowns by overloading or overlapping of radio frequencies resulting in data loss and countless re-transmissions.
For not all of these problems a solution exists, so that the influence by these factors on networks can only be reduced through high sturdiness or avoidance policies. In order to cater to rising demands and to contribute to the effectivity of these stability measures, one has to use special software or even hardware to control and monitor network traffic, or one has to even find alternative solutions.

In the past few years almost every average citizen in the developed world has acquired a smartphone and/or a tablet. Due to this spread of technology and increasing level of performance of these devices network utilization has massively increased leading to

network overload. Since possibilities are bound to physical laws network performance cannot be increased infinitely so different ways such as limiting transmitted data, better data compression or more effective routing and forwarding algorithms are to be found.

These developments come at significant cost, not only in research but also substituting hardware for newer one as new software and protocols are not always compatible with outdated hardware. Finding a way to effectively transmit data has become one of the main focuses. The limit has not been reached yet, but probably will in the near future.

This thesis investigates hot to extend networks by relaying data from one network into the next one by being multi-connected via an *Android* device. The Android device will serve as a bridge, monitoring and relaying data. The focus will be on receiving from a 2.4 Gigahertz (GHz) frequency network and relaying the marked data to a network on the 5.0 GHz waveband. Moreover, some aspects and characteristics of this capture and relay concept are going to be tested and analyzed in order to provide a statement on its capabilities. If this setup turns out to be efficient its main use will be with the *Opptain* application developed by the professorial chair. Opptain correlates mobile devices to an opportunistic network using WLAN. Participants connect ad-hoc to a larger network when they encounter each other. Through the occasional contact with the moving nodes the exchange of data over large distances without central infrastructure shall be enabled.

1.2 Structure of Thesis

In this thesis the Android External Wifi (AEW) application will be extended in order to relay data in *WLAN*. The data needs to be marked in so that it can be identified. Furthermore, the experiment needs to be carried out in such a way so that the framework can be extended in the future.

Chapter 2 provides an overview on kindred solutions found in the current academic literature. These articles mainly focus on the multiple in multiple out (MIMO) technology and subsidiary on relaying.

Chapter 3 introduces the basic terminology used and applied concepts. In addition, header layouts for different protocols which are relevant for subsequent chapters are explained.

Chapter 4 deals with the functionality and realization of different parts of the setup. This includes the developed *MyServer* and the *MyReceiver* program. Additionally it will be seen how the AEW framework was extended and why different implementations were chosen.

Afterwards Chapter 5 describes how the tests were performed and provides a short evaluation of results. The chapter starts off with some basic measurements for the environment, followed by some performance and functionality tests and closes with a real life example.

Chapter 6 summarizes the framework including a conclusion on operational capability derived from the tests. This is followed by an outlook on possible extensions of the framework and ideas for future tests.

All used abbreviations can be found in Abbreviations and all used technical terms in Glossary. All names and commands will be in *italic* on their first appearance.

Chapter 2

Related Work

In order to accomplish the goal of extending networks by having a multiconnection one needs hardware as well as reliable software. When looking at similar projects or solutions there are not many which meet the criteria as this is a new field of research. Therefore, this section focuses on several projects and on the AEW.

2.1 Multi-Cell MIMO Cooperative Networks

In [GHH+10], the theory and techniques behind multi-cell MIMO cooperation in wireless networks and challenges are discussed. Theoretical models are created and mathematically analyzed. Mostly related is [GHH+10, Chapter E] where a setup is dissected that has no direct link from the base to the mobile station. Therefore a full- and a half-duplex relay are considered and compared. The conclusion derived is that the major problem lies with information exchange such as user data and channel state.

2.2 Distributed MIMO Systems

In [BRMP12] and [BRM+13] examples of distributed MIMO systems and some aspects of them are investigated. This kind of system consists of some access points connected to a central server that acts as a large distributed multi-antenna access point. It is related in such a measure that synchronous communications take place and a multi-connection is established. The first paper makes an analysis of such systems followed by some experimental measurements with synchronization accuracy. The latter paper proposes a novel scheme that is accurate enough to allow a multiuser MIMO system. The performance is then analyzed theoretically and practically, finished off by presenting the used Media Access Control (MAC) protocol.

2.3 Triple MIMO Communication

In this paper [YYC13] a case is proposed where a communication between two MIMO users is relayed by a MIMO bridge using a two-way relay channels. A rather theoretical approach is taken where a system model is developed. Different factors and capabilities are assessed and the achievable volumes are analyzed. In contrast to this thesis, only a theoretical, but no practical approach of relaying data has been chosen. The results are promising for future research as the proposed system reaches a small difference between achievable and average sum-rates for each relay-antenna.

2.4 Device-to-Device Communication

[AWM13] defines and analyses the different types of communications which take place between mobile devices including contacting a base station or using other devices as a relay. An architecture is proposed in order to match the rising demands of mobile communication. Here factors such as peer discovery, resource allocation, power control, interference management and security are theoretically discussed and evaluated.

2.5 Optimizing MIMO Relay Systems

The [RM11] mentioned paper takes a theoretical approach to multiaccess communication by using multi-hop non-regenerative MIMO relays. This is done by establishing a theoretical system model and proposing two simplified optimization algorithms which reduce computational complexity and signal overhead display. These two cases are mathematically proven.

2.6 Relay System on a Cargo Ship

The paper [ML11] takes a look at a relay system transmitting from the lower to the upper part inside of a ship. Therefore the 2.4 GHz frequency with two links is used and a fitting relay system is proposed. The characteristics such as average received power, root mean square relay spread and ray decay factor are tested. These tests are done at a measurement site and afterwards analyzed mathematically. It is determined that the relay system is mainly limited by one link influenced by the metallic structure and guiding effect of the cargo hull. Besides this fact the data rate is still sufficient enough for voice or low-resolution video communication.

2.7 Android External Wifi

The AEW [Men16] is a framework build on the open source Application Programming Interface (API) of Android PCAP by Kismet Wireless. The idea behind this is trying to get the known Packet CAPture (PCAP) to work on an Android device without needing any specific rights. Especially the use of root rights is being avoided as this can cause irreparable damages to the operating system.
The application uses an external network card with the RTL8187 chip which is connected through an Universal Serial Bus (USB) On-The-Go (USB-OTG) cable to an Android device running on Android version 4 or higher. This chip is then put into monitor mode

and with the use of the Java Native Interface (JNI) *Linux* library *libpcap* used to capture passing by 2.4 GHz network traffic. Additionally the channel can be selected in order to specify the frequency to capture on. These captured frames are then saved to *.cap file format which later on can be read with applications such as Wireshark. This framework laid the groundwork for this thesis due to previously mentioned characteristics.

2.8 Summary

In this section some examples of papers which engage in different aspects of the MIMO concept were illustrated. This concept utilizes multi-connection as multiple frequencies can be accessed. In some cases this is even associated with data and communication relay, but rather operating on larger scale purposes. For this thesis the AEW served as the basis for further developments.

Chapter 3

Fundamentals

In this chapter terms and concepts used in this thesis are explained. Basics of monitor mode, Interleaving and the PCAP file format are summarized. Also crucial concepts such as the Magic Number or the used standards of the IEEE 802.11 WLAN are discussed briefly. Lastly the Internet Protocol (IP) and the User Datagram Protocol (UDP) protocol with the focus on their header structure are introduced.

3.1 Terminology

3.1.1 Monitor Mode

Monitor mode is one of six possible modes a WLAN card by the 802.11 standard can be in. In this mode all received frames are delivered to the operating system as mentioned in [mon16] and [tcp15]. In most cases this happens without any prior validation. This makes it possible to receive frames from neighbouring channels and others with flawed checksums. Also this mode does not use any filters for Service Set Identifier (SSID) or IP addresses. Therefore the whole frame including all the headers is passed on to the operating system. In other modes usually only the Ethernet frame is passed on.
In monitor mode the network card is only able to monitor passing-by traffic without being able to send anything anymore. This makes it possible to eavesdrop unnoticed on

networks and radio transmissions. Therefore data packets with an unknown decryption key can be received.

3.1.2 PCAP

Packet CAPture (PCAP) is an open source API used to monitor and capture network traffic as defined by [tcp15]. PCAP can produce network recordings which are saved to the format of *.cap or *.pcap files. These files can then be read, manipulated and analyzed with applications such as Wireshark or tcpdump. To monitor network traffic one needs a suitable network card which can be shifted to monitor mode. Then PCAP is switched between the card and the operating system and writes all the received frames into a logfile.

3.1.3 Magic Number

A Magic Number is usually defined as a sequence of special values put at the beginning of a file for identifying it. Special means that this constant value is hard-coded and special only to those parts which have to interpret it. Magic Numbers are usually chosen at random but in a way that they are easy to identify or in some cases sequences which do not occur very often in the used protocol. The general idea is mentioned in [Sim94, Chapter 6.4] and further details can be found in [mag15].

3.1.4 Interleaving

When speaking of Interleaving a concept of forward error correction is meant. It is commonly used in digital communication in order to correct occurring burst errors. The most efficient way of sending packets would be to send each ID multiple times before moving on to the next one. In case a burst error occurs which eliminates a whole packet ID the missing part cannot be restored. The basic idea of Interleaving is to send a variable

number of different packet ID's once before repeating this step multiple times consecutively and then moving on. Through this smaller burst errors can be corrected as briefly mentioned in [MMS+03, Chapter 2.5].

3.2 Standards of 802.11 WLAN

The IEEE 802.11 is a group of standards for radio networks on the basis of Ethernet. Originally it had the transmission rates 1 and 2 Mbitps and operated only on the 2.400 to 2.485 GHz frequencies. This standard was continuously extended, mainly to improve transmission rates, data security and the communication between different hardware devices as explained in [80216].
In the 802.11b standard once again the physical layer was enhanced adding 5.5 and 11 Mbitps rates with fallback rates of 1 and 2 Mbitps.
The 802.11a extension brought an improvement of the physical layer. It added the transmission rates 6, 9, 12, 18, 24, 36, 48 and 54 Mbitps. It also expanded the waveband by the 5 GHz frequencies. This standard is incompatible with 802.11b.
The standard 802.11g attempted to combine the best characteristics of the previous standards 802.11a and 802.11b. 802.11g access points are compatible with 802.11b wireless network adapters and vice versa.
The last mentionable standard is the 802.11n which was designed to improve the 802.11g. The idea was to increase the amount of bandwidth by utilizing multiple wireless signals and antennas, also know as the MIMO technology. This standard functions with both 802.11b and 802.11g standard.

3.3 Internet Protocol Version 4 Header

Important for this thesis is to understand how the IP header is structured. It is usually 20 byte long but with additional options it can grow up to 24 byte. Relevant are only the fields containing the source address, byte 12 to 15, and the destination address byte 16

to 19. This is defined in [Pos81, Chapter 3.1].

3.4 User Datagram Protocol Header

The UDP header is made up of four times four bytes and is defined in [Pos80]. The first field makes up the source port and the second one the receiver port. This is followed by four bytes representing the length including the UDP header. Finally the four byte Checksum used to verify the correctness of the received data is listed.

3.4.1 UDP Checksum

The 16-bit Checksum is calculated over the so-called pseudo header using information from the IP header, the UDP header and the data. If necessary it is filled up with eight bit of zeros at the end to make it a multiple of two octets.

Ipv4 Pseudo Header

The pseudo header is made up of the source and the destination address of the Internet Protocol Version 4 (IPv4) header. This is then followed by a byte of zeros and the protocol type, which in the case of UDP is 17 or in hexadecimal notation 0x11. This is followed by the UDP packet including the header with the Checksum set to zeroes.

UDP-Checksum Calculation

For the calculations the pseudo header is divided into multiple 16-bit words, which are added up using one's complement arithmetic. Each carry-out bit or better described as the 17th bit produced is then added up to the least significant bit. After the summing up has been carried out, the 16-bit word is complemented, resulting in the UDP Checksum value.

Chapter 4

Implementation

This chapter first provides a short explanation of the planned setup. Afterwards, the realization of this setup and the developed programs are explained.

4.1 General Setup

The general idea of this setup is to extend networks by being multiconnected through having two different 802.11 WLANs transmitting on different frequencies. In this case, a sender in a 2.4 GHz network sends signed data which is then identified and captured by a waypoint. This interstation is then connected to another network on the 5 GHz waveband and relaying the monitored data to a receiver.

Figure 4.1 shows the planned network setup.

In Figure 4.1 the designed structure can be seen. The sender application was named MyServer as the main function is to send out packets. The waypoint is going to be an adjusted version of the AEW as the necessary functions are not yet implemented. At last the receiver was named MyReceiver as its prime task is to receive data.

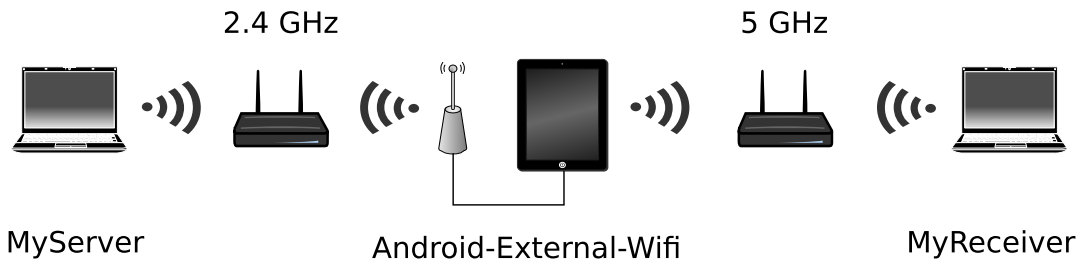

Figure 4.1: Planned network setup

4.2 MyServer

The MyServer program was created as a *JavaFX* application using UDP as a transmission protocol. The Graphical User Interface (GUI) is kept simple and only offers the basic functions needed. It allows to determine the packet size in bytes and to choose between sending measured in time or by the number of packets as an Integer. Afterwards one can add an IPv4 relay address or simply leave it empty. If left empty the packets will later be relayed by a Broadcast address per UDP. Otherwise Transmission Control Protocol (TCP) is used.
After filling in all the information one can hit the *Send Data* button which in the ideal case sends the packets. If anything was entered incorrectly the output label informs the user on the incorrect input.
The MyServer GUI can be seen in Figure 4.2.

4.2.1 Mechanics

The structure of the program is made up of three primary classes. First the usual *Main* which first checks all the network interfaces present and puts their Broadcast address into an *ArrayList*. Afterwards the GUI is setup and the *Controller* is added.

The Controller initializes the basic Strings and adds a *Listener* to the *ChoiceBox*. The

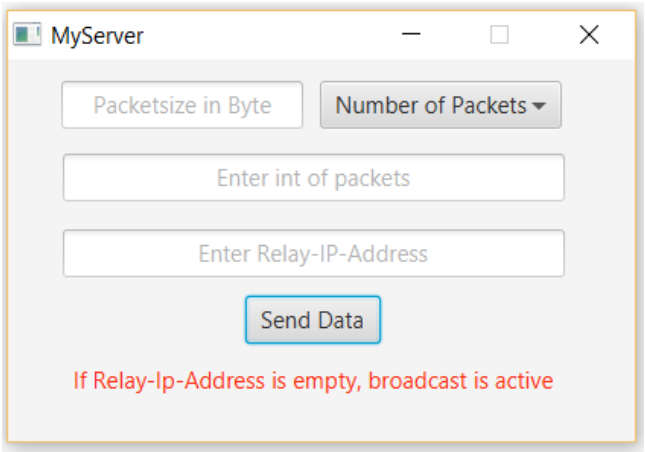

Figure 4.2: The MyServer GUI

method *handleSendData* is called when the Send Data button is clicked. It creates an object of *Packet*, opens a *DatagramSocket* and takes the input values entered in the GUI. If needed they will be parsed into the right data structure. Depending on the entered values it either goes into the if query which sends a fixed number of packets or for a fixed time. At this point it gets the ArrayList of the Main and sends it to each Broadcast address as pre-defined. Also the object Packet is transformed in to a *DatagramPacket*. If anything fails or cannot be carried out, the user is notified with the applicable message. One should mention that when sending by number of packets subsequent assignments can be given. Whereas with the time in seconds a *Timer* with the subclass *TimerTask* will be used to lock the GUI for the time sending so that no further tasks can be given.

The object Packet receives three attributes. First the relay destination IP address, second whether the prior address was valid and last the packet size. Depending on the forwarded information a byte array with varying size of the Magic Number header is created. In case the address was not valid an empty packet is created which is recognized as an incorrect IP address by the Controller. This is done in order to minimize the number of if queries.

4.2.2 Magic Number Header

The header is made up of four fields in total. The first field contains the Magic Number sequence which in this case is the value taken from [mag15]. The second field holds the control length. This is followed by the control bits. Up until now only two values have been assigned; the first one being only zeroes for a Broadcast relay and the second one, made up of zeroes and a one at the end, indicating that a relay address has been included. The last field is made up of the relay IP address and is therefore only optional. Figure 4.3 visualizes the components of the Magic Number header.

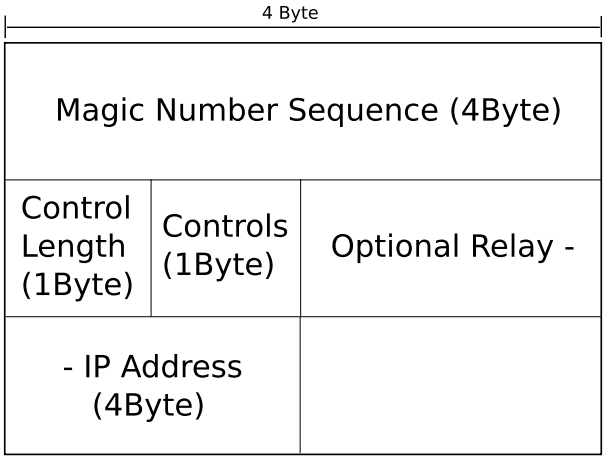

Figure 4.3: The Magic Number header

4.3 Android External Wifi

The program had to be extended in order to filter captured frames and to then relay them. The class *Rtl8187Card* has a byte stream in the *run* method. These packets are then delivered to the *handlePacket* method of the *PcapLogger* class. Here, major adjustments were made.

4.3.1 IP Filter

The first aim was to filter the incoming packets by source address. Initially the *MainActivity* got a new boolean value *boolFilter* which indicates whether the IP filtering is enabled or not. Also a new string named *filterAddress* was added, which holds the address filtered by. To select a filter the main menu of the AEW got a new button which is labelled with *Enter an IP-Filter-Address*. When the button is clicked a new pop-up window appears where an IPv4 address for filtering can be entered. This pop-up uses a new method which validates the address. If no IP is entered or an invalid IP then the filter is deactivated. In case a correct IPv4 address is submitted it is added into filterAddress. Then the text below the button label changes to "activated, filter: (entered address)" from "deactivated" and boolFilter is set to true.
This boolean value is checked by an if query in the handlePacket method in the PcapLogger class. Additionally it is checked if the packet is longer than 64 byte which otherwise could result in an error. Next the receiver IP is taken out of the packet and copied to a shorter byte array and afterwards parsed into a String. This string is then compared to the filterAddress of the MainActivity.

4.3.2 Magic Number Processing

After the address that is being filtered has been confirmed, it moves on to the next if query checking for the Magic Number sequence. If this also passes then the packet is registered as a processed packet as it has been validated. This is followed by all the Magic Number header values being transferred into variables. Also the UDP checksum is calculated over the IPv4 pseudo header. This is done by the method *checksumCalculator*. Now the calculated checksum and the one taken out of the packet are contrasted. Should the packet be flawless the Magic Number controls is analyzed. Otherwise it is counted as an corrupted packet in the *PacketHandler* and displayed in the GUI.
Depending on the value of the control field the receiver relay IP is taken out of the packet. At last the UDP length is examined and the packet data is copied into a new instance of *RelayPacket* with all the headers removed.
The RelayPacket contains only three variables; the first one being data. This is followed

by a boolean *tcpOrUdp*, which indicates if a relay address has been submitted or not. If so it is set true and an IP address can be found in the last String value.
After the RelayPacket has been created it is added to a *LinkedBlockingQueue* called *relayQueue* which takes in objects of the type RelayPacket. This queue was chosen as in its initialization it does not require a limit. This makes it more convenient to work with unknown amounts of packets as it can grow up to the maximum value of an Integer.

4.3.3 Relaying Process

The relay function has been realized by starting a new thread in the *startLogging* method of the PcapLogger. For this reason relaying is only active as long as the logging process is running. This thread is called *workerThread* and is an instance of *Worker*. It is called by in case the filtering has been enabled. This class holds a TCP client and an UDP server. It has one loop which goes on all the time assuring that when the logging is started and the filter is active, packets are being relayed. The second while loop checks whether the relayQueue is empty or not. If it is not empty a RelayPacket is extracted. Afterwards the boolean value tcpOrUdp of the packet is checked. If it is true the IP address is taken out and it is relayed by using TCP but beforehand the length of the data is sent. Here a TCP client was used instead of a server so that as soon as data is to be send a connection can be initiated.
In case the boolean value is set false the packet is relayed via UDP Broadcast. This address is taken out from the MainActivity in which a new ArrayList has been added for this purpose. The list is filled in the *onCreate* method by the *findBroadcast* method which is the same code as in the MyServer program.

4.4 MyReceiver

The MyReceiver was also created as a JavaFX application and it is made up of five different classes. The *Main* has only two major purposes. The first is setting up the GUI which is kept fairly simple. Through clicking the *Refresh* button all packets received until

now are displayed in the list and through selecting a packet one sees more information about it. So far, the only attribute implemented is whether it is relayed by TCP or UDP. Through clicking the *Reset* button the list is emptied. The list items are made up of the class *Packet* and only hold the basic values such as the number, the size and by which protocol, TCP or UDP, it has been relayed. The surface can be seen in Figure 4.4.

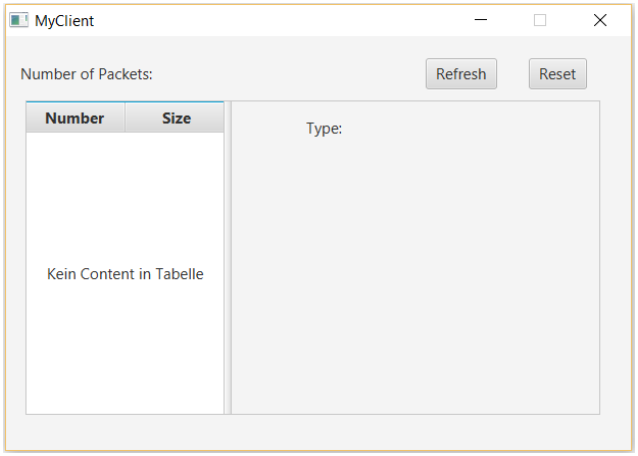

Figure 4.4: The MyReceiver GUI

The second purpose of the Main is to start two *Threads*. One Thread runs the UDP client and the other one the TCP server. The UDP client is included in the class *BroadcastReceiver* and is a simple client which accepts packets of any size. The TCP server is included in the class *Server*. It is a simple server construction which first receives an Integer, being the length of the following data, and then the actual data.

4.5 Timestamp Extension

In this section the first extension of the programs is outlined, where a simple timestamp is added to the data at each station it passes through. This was necessary so that the travel time of the data being transferred could be measured.

4.5.1 MyServer

The MyServer program was extended so that it can add a timestamp to the packet. This was realized by upgrading the Magic Number header. In this version the control field has two new option which are two and three. Two stands for using UDP and three for relaying with TCP, including a timestamp. This was important as depending on the protocol an address is included making the header larger. After the header the timestamp is included which is created by the method *System.currentTimeMillis()*. This function returns a long value and therefore eight bytes are needed. In order to always include the current time before sending the datagram creation had to be moved to the sending loop.
Lastly, a new checkbox was added to the GUI. If this box is clicked then packets are created with a timestamp.

4.5.2 Android External Wifi

The AEW was extended so that it can handle the two new control options of the Magic Number header. This was done by adjusting the relaying process and adding a new if-clause for each option. The packets are relayed as before but after the eight byte timestamp of the sending source an eight byte timestamp of the respective relaying device is added. This is again done with the command system.currentTimeMillis().

4.5.3 MyReceiver

The MyReceiver application was customized so that it can filter out both timestamps from a packet. These raw values are then subtracted from the returned time of system.currentTimeMillis() of the receiving device. These are the travel times of the packets in milliseconds. The first one being the time from the original sending source to the final receiving station and the second one the time from the bridging device to the end point. These two calculated values are then used to calculate the time needed from the initial source to the bridging device. These three values are then added on to a variable which was created in the *Main* class to hold the travel time of each path summed up.

The GUI was extended so that it can display the average travel time for all three cases. For this the variables of the Main class holding the sum of time travelled are then divided by the number of packets received.

4.6 Image Transfer Extension

In this section the second extension of the programs is explained. This was necessary so that a data transfer of an image could be realized in order to show that not only sensor data could be relayed. Further on, no adjustments had to be made to the AEW as the scenario was a simple relaying process for the bridging device.

4.6.1 MyServer

The MyServer application had to be adjusted so that it can relay images of the .jpg format. For this purpose the choicebox in the GUI received a new option. When this option is selected a new button appears with which a .jpg file can be selected. Further on a CheckBox has been added with which Interleaving can be activated. Finally a new field was added holding the value for sending repetitions, here called *repetition*. When the sending process is started the selected file is converted into a byte array. This array is then fitted to the entered packet size including the Magic Number header. In case Interleaving is deactivated each package is sent the entered number of repetition before moving on to the next ID. This is called the *casual concept*. In the other case each packet is sent once up to the maximum ID and this is then repeated afterwards as many times as entered in repetition minus one. As a relaying protocol only TCP can be used.
To additionally control the reassemble process eight control bytes have been defined. They are made up by four times four bytes consisting of *ID*, *maximum ID*, *file type* and *packet data length*. So far the type field is not used as only .jpg files are transferred. This can be seen in Figure 4.5.

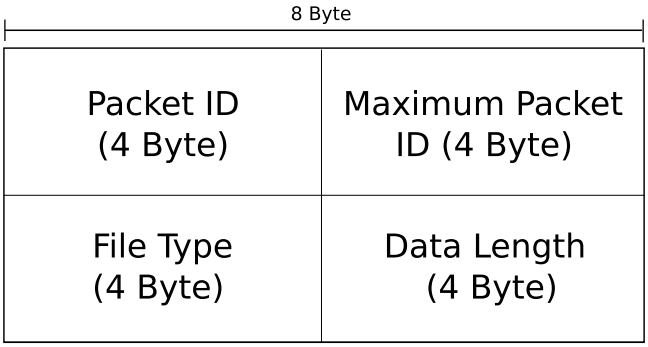

Figure 4.5: Image assemble controls

4.6.2 MyReceiverV2

The MyReceiverV2 is the known MyReceiver but all the GUI functions have been erased. There is only one function left which creates an image from the packets if all are received. This can be initiated and displayed through a click on the *Create .Jpg* button. A path has to be selected before the reconstruction process can begin. Through this action also the number of packet ID's received from the maximum number of ID's is displayed.

4.7 Limitations and Known Problems

In this section limitations caused by hardware, drivers and user space are outlined to get a better understanding of the restrictions with this setup. Additionally, known problems occurring with the developed setup and noticed during testing are addressed as they are outside the scope of this thesis and have not been fixed.

4.7.1 Limits

The biggest limitation in this setup is caused by the RTL8187 chip as it only supports the IEEE 802.11 standards a, b and g. By that it limits the top speed and cannot monitor the common IEEE 802.11 standard n.

Further on only non-encrypted traffic can be processed for obvious reasons.

4.7.2 Channels

During the extension of the framework and different tests while developing the MyServer program, even though channel four was locked on in the AEW application, the log printed out that the fixed channel was channel six. It cannot be said with certainty whether it was just printed out wrongly or if actually the wrong channel was locked on. This occurred a few times. there was no time to look into the problem as analyzing the whole application would have consumed too much time. Looking into this problem (analyzing major parts of the application) would have been beyond the scope of this thesis.

4.7.3 Screen Lock

Another problem occurring with the AEW application was that when the program was opened and the tablet locked the screen after the start of the logging function would result in no traffic being recorded. This could easily be worked around with a restart of the application or a re-connection of the external WLAN card.
Additionally, the orientation lock shall be activated while logging, as turning the screen while logging causes the application to crash.

4.8 Summary

This chapter showed how a version of network extension has been realized by being multi-connected. The packets made and signed by the MyServer program are send per UDP Broadcast. These packets are then captured by the AEW application and identified through the incorporated Magic Number sequence. All the additional unneeded overhead is cut off and the raw data is relayed by TCP or UDP depending on what the user of the MyServer program has chosen. Finally these packets are received by the MyReceiver

application.

Chapter 5

Tests and Results

This chapter firstly discusses the hardware used and the environment the tests are conducted in. This is continued by tests analyzing surrounding network traffic. After that the most functional transmission rates and packet sizes in this setup are discussed. Moreover, the transfer time will be investigated, followed by an example of real life application.

5.1 General Test Requirements

All tests have been conducted with the same hardware and software. Depending on the test setup a *Linksys WRT54GL*, supporting the 802.11 standards b and g, and a *Fritzbox 7490* router, supporting the ac and n standard but also being compatible with a, b and g, were used. The Linksys worked with the Firmware Version 4.30.14 and the Fritzbox with the Firmware Version 113.06.51. As measurement points the sending laptop, a *Acer Aspire V15 Nitro VN7-571g-55ZA* with Windows 10, supporting the 802.11 standards a, b, g and n, and the receiving laptop, a *Asus G750JX-T4070H* with Windows 8.1, supporting the 802.11 standards a, g, n and ac, were used. For the AEW application a *Nexus 9* with Android 5.1.1, supporting WLAN standards a, b, g, n and ac, and the *Alfa AWUS036H*, supporting the 802.11 b and g standard, were used.
The testing environment was an approximately 10 meter long room with no electronic devices inside besides the ones used for testing. The MyServer program, run on the

Acer and in the 2.4 GHz network, was used with Wireshark as a reference for outgoing packets. On the receiving side the Asus laptop with the MyReceiver program in the 5 GHz network were used. Again Wireshark was used as a reference for incoming traffic. The finalized network setup can be see in Figure 5.1.

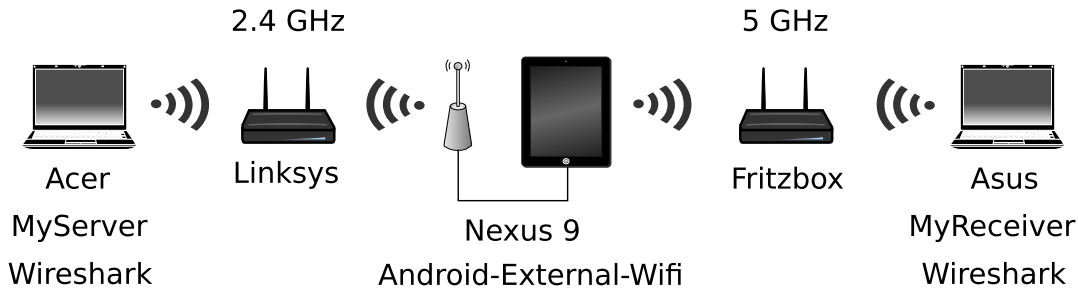

Figure 5.1: Used network setup

5.2 Sources for Environmental Interference

The tests in this section were conducted in order to identify possible interference sources and how high surrounding average traffic was on the 2.4 GHz waveband. This also served to identify the channel with the least communication for following tests to be conducted. Additionally this was done to identify surrounding Broadcast addresses which could have caused inaccurate results.

5.2.1 Surrounding Traffic

For the test on how high the surrounding traffic was, the Nexus 9 was used with the AEW application and channelhopping enabled. Three test rounds were conducted, each at a different spot on an axis approximately five meter apart. *Position 1* was at the beginning, *Position 2* in the middle and *Position 3* at the end of the axis. One round included five tests which each last five minutes long. At the beginning each position got three

consecutive rounds. Then each position received an additional two rounds. This is done in order to avoid possible variations in network traffic causing imprecise results.
After running the packet capture the PCAP files were analyzed in order to get different information about the surroundings. This included the number of neighbouring networks and their Broadcast addresses, which depending on the amount could have an influence on the results. Also it cannot be excluded that passing by devices might have sent Probe-Request-Frames, which would only have had a minor to no impact.

	Units	*Round 1*	*Round 2*	*Round 3*	*Round 4*	*Round 5*
Pos. 1	Quantity	7962	11312	7196	3912	2016
	Size in KB	722	1025	643	1430	440
Pos. 2	Quantity	4192	3063	2313	1701	1629
	Size in KB	520	724	415	420	400
Pos. 3	Quantity	1494	2253	1738	1571	1594
	Size in KB	304	376	322	382	374

Table 5.1: Other network traffic around testing environment

Table 5.1 shows the number of captured frames and their size. As one can easily see Position 1 has monitored the most. On closer examination of the PCAP files one easily notices, that the most traffic consists of a communication between a nearby WLAN router and a computer. This is made up of beacon, management and probe request frames. Also it can be seen that in the later rounds the traffic around Position 1 decreased, indicating that the previously mentioned communication had ended.
One can also see that there are some other routers in range, but not as close, since only very few of their frames were captured. The rest of the files were made up of different communications of devices connected to the surrounding networks.

Due to these results, in the following test scenarios the devices working with the 5 GHz frequency were set up closer to the position with the highest 2.4 GHz frequency traffic, Position 1. Further on in following tests the Broadcast address 192.168.1.7 was used as no surrounding network showed a similar broadcasting address.

5.2.2 Ideal Channel

In this scenario the spot, Position 1, with the highest traffic from the previous test was selected. At the hotspot the Nexus 9 with the AEW was used for five minutes in three rounds for each channel. The channels, ranging from one to eleven, were fixed. Due to the high time expenses the rounds were reduced from five to three.
The aim was to determine which channel had the lowest network traffic. As a reference, the built-in software of the Fritzbox 7490 router was used, displaying the channels with their traffic. In order to get precise information, the WLAN adapter of the used Acer laptop was turned off and connected via Local Area Network (LAN) cable to the router. This was done in a different turn as the Broadcast frames of the router had corrupted the AEW measurements. This was done in a ten minute test with a 30 second refreshing interval.

Table 5.2 shows the traffic on the different channels at Position 1. As one can easily point out the most traffic is on channel 6, followed by channel 4, 5, 7, 8 and 11. This indicates that the closest router is probably sending on channel 6 which also causes emission to the neighbouring channels. This was supported by the internal software of the Fritzbox 7490 router which in the whole 10 minute period displayed, that there was one router with a strong signal sending on channel 6 but also accessing channels 4, 5, 7 and 8, which explains the higher traffic on those frequencies. Additionally there seemed to be another router accessing channel 6 but with a very low signal.
Also on channel 11 was quite a lot of traffic. The Fritzbox and the PCAP files show that one further away router with a weak signal was sending on this channel. This was probably in order to avoid collision with the closer by router.

As a result further tests with the 2.4 GHz frequency were conducted on channel 2 as this channel showed the least traffic. Channel 10 was not selected, as this channel is between the channels 4 to 8 and 11. This would result in some emission of the neighbouring channels and therefore might have caused imprecise test outcomes.

	Units	*Round 1*	*Round 2*	*Round 3*
Ch. 1	Quantity	202	147	167
	Size in KB	53	33	40
Ch. 2	Quantity	27	14	22
	Size in KB	4	3	5
Ch. 3	Quantity	252	208	244
	Size in KB	51	44	48
Ch. 4	Quantity	4320	5494	4417
	Size in KB	100	1063	1000
Ch. 5	Quantity	733	937	1446
	Size in KB	199	226	241
Ch. 6	Quantity	9462	8344	7542
	Size in KB	2060	2004	1882
Ch. 7	Quantity	2123	2230	2175
	Size in KB	495	532	510
Ch. 8	Quantity	3404	2748	3369
	Size in KB	792	640	794
Ch. 9	Quantity	152	151	252
	Size in KB	34	34	45
Ch. 10	Quantity	51	52	40
	Size in KB	7	7	5
Ch. 11	Quantity	4102	3881	1266
	Size in KB	981	844	274

Table 5.2: Traffic on different channels at Position 1

5.3 Optimum Transmission Rate

In this scenario the ideal transmission rate for monitoring and relaying frames has been determined. On each of the twelve transmission rates the Acer sent packets of 1472 Byte for three times five minutes. After the transmission rate was adjusted, a 30 second pause was made. Again due to the high time expenses the number of rounds were reduced from five to three.
The packet size was aimed at 2244 Byte which with added header of 60 Byte would have equaled 2304 Byte, the Maximum Transmission Unit (MTU) of the 802.11 WLAN. This was done in order to maximize network traffic and to reach the full network utilization. As it turned out this was not possible because even though the Fragmentation Threshold was set to the maximum of 2346 it would have not been reached. This was due to the limit set by the Ethernet MTU limiting the packet size to 1500 and causing the Fragmentation Threshold not to be considered. The only possible way to reach the Fragmentation Threshold would have been to use a router which only supports WLAN. For this reason the packet size was set to the maximum of 1472 Byte inside the MyServer application. Any larger packet size would have caused fragmentation due to additional header.
The testbed was as follows: The sending source was placed around Position 3, the Nexus 9 was placed at Position 2. Between these two the 2.4 GHz router was placed. At Position 1 the receiving end was located, between there and Position 2 the 5 GHz router was placed. This setup was used in the following sections.
After five minutes had passed, the MyServer application stopped sending, but the AEW and the MyReceiver program kept running until the MyReceiver stopped receiving frames, because the tablet and possibly the 5 GHz network router had built up a buffer which had to be worked off.

It was expected that with the increasing transmission rate, more packets would be sent and also more frames would have been captured and relayed. It was highly possible that with the higher rates more collision would take place resulting in less frames being captured. Also the MyReceiver would receive as many frames captured by the AEW as TCP was being used for relaying. Additionally it was estimated that as the transmission rate would rise so would the number of corrupted data.

	Round 1		*Round 2*		*Round 3*	
Rate	*Sent*	*Captured*	*Sent*	*Captured*	*Sent*	*Captured*
1 Mbitps	131 241	16 754	128 888	16 947	132 007	16 904
2 Mbitps	170 134	31 720	191 838	29 672	193 165	29 806
5.5 Mbitps	324 808	62 505	327 013	61 632	330 807	62 413
6 Mbitps	336 190	64 978	332 339	65 271	339 934	62 735
9 Mbitps	383 333	84 070	391 973	78 341	390 874	80 369
11 Mbitps	433 010	90 607	434 564	90 509	435 936	90 726
12 Mbitps	463 688	81 216	460 996	82 645	451 776	83 723
18 Mbitps	525 506	114 998	530 983	112 313	481 276	127 430
24 Mbitps	589 567	107 607	602 770	96 575	606 143	94 713
36 Mbitps	651 745	37 513	652 277	43 217	655 589	35 446
48 Mbitps	680 209	308	676 643	319	681 625	586
54 Mbitps	701 243	111	691 206	46	698 319	31

Table 5.3: Sent, captured and received frames

Table 5.3 shows the number of frames sent by MyServer and captured with the AEW application. The frames received by MyReceiver were not included as they were the exact same values as the number of frames captured by the tablet. Also the differentiation between corrupt and intact packets was not displayed because in all rounds not a single corrupt packet was recorded. These figures were validated by analyzing the Wireshark files and the AEW PCAP files. Although the Acer files showed some minor variations with outgoing UDP packets, this should not have had a big impact on the results.
As the transmission rate increased so did the number of sent frames. Interesting is that simultaneously the number of captured frames increased up to the rate of 18 Mbitps. At this point the number of frames first started decreasing and then with 48 and 54 Mbitps dropped to a near null rate. It was also noticeable that at 18 and 24 Mbitps the fluctuation of the number of captured packets increased before dropping immensely.
This behavior might be explained by the increased traffic in the air, which caused more

overlapping and resulted in overstraining the capturing device. Additionally it is interesting that not a single corrupted frame was received in any turn. By taking a closer look at the captured files of the AEW with Wireshark and the filter for frames with bad Checksums, no frames appeared to be corrupted. This raises the question whether the frames were corrupted in a way that the receiver IP address and/or the Magic Number sequence were affected or if the AEW simply could not monitor corrupt frames as lower layer functions ignore tainted data.

With an increasing transmission rate, the time taken after the five minutes of sending for the Android tablet to work off the queue and relay packets also increased. This does not consider the irregularities caused by 18 Mbitps.

Rate	*Actual transmission rate*	*Percentage captured*
1 Mbitps	5.3 Mbitps	13 %
2 Mbitps	7.5 Mbitps	16 %
5.5 Mbitps	13.2 Mbitps	19 %
6 Mbitps	13.6 Mbitps	19 %
9 Mbitps	15.7 Mbitps	21 %
11 Mbitps	17.6 Mbitps	21 %
12 Mbitps	18.5 Mbitps	18 %
18 Mbitps	20.7 Mbitps	23 %
24 Mbitps	24.2 Mbitps	17 %
36 Mbitps	26.4 Mbitps	6 %
48 Mbitps	27.4 Mbitps	> 1 %
54 Mbitps	28.1 Mbitps	> 1 %

Table 5.4: Actual transmission and capture rate

Table 5.4 sums up the results from the actual transmission and capture rate. It shows the percentage captured by the AEW and the actual rate with which the Acer laptop sent signed frames. This is calculated from the number of frames sent, multiplied by the frame size and divided by the time of five minutes. As one can notice the actual transmission rate in most cases surpassed the appointed rate. The actual reason for this occurrence cannot be given but was probably caused by the used hardware and its driver.

Further investigation of this deviation was not possible due to closed source.
The results suggest that 18 Mbitps is the ideal rate, however, at this rate, the steadiness of the capturing capability was rather discontinuous. Therefore, instead, either a transmission rate of 9 or 11 would be ideal and should be selected.

5.4 Corrupt Packets

Initially it was planned to measure the corruption rate of the packets in an environment with low traffic relative to the distance with the ideal transmission rate. Looking at the PCAP files and the AEW output of the previous scenario it was realized that none of the signed UDP packets were monitored corrupted. It seems as if lower layer functions of the external IEEE 802.11 WLAN card initiated by hardware, firmware or driver filtered out corrupted frames. Investigating this would consume too much time and would be beyond the scope of this thesis.

5.5 Varying Packet Sizes

This test was done in order to see how different packet sizes influence the capturing rate. In the Section 5.3 we have seen that there is no distinct best transmission rate. So for this section the rates 9 and 18 Mbitps were selected as they showed the best capture rates. 11 Mbitps was not selected as it is close to 9 Mbitps which makes a better median value. Also 2 Mbitps was selected to have a lower bound rate for comparison. This test was done to investigate the behavior of the capture rate with different packet sizes. This will be done with five different sizes in five rounds of five minutes. In addition to the results of the maximum size of 1472 byte from Section 5.3, two additional runs were carried out to sum up to five. The other four sizes were made up of 75% 1104, 50% 736, 25% 378 and 5% 74 byte.
The same testbed as in Section 5.3 was used.
It was estimated that with shrinking packet size a lower capture rates would be achieved.

Rate	*Round*	*1472 byte*	*1104 byte*	*736 byte*	*378 byte*	*74 byte*
2 Mbitps	1	19 %	15 %	18 %	12 %	10 %
	2	15 %	15 %	15 %	14 %	11 %
	3	15 %	15 %	12 %	15 %	11 %
	4	17 %	15 %	14 %	17 %	10 %
	5	18 %	14 %	14 %	15 %	11 %
	ø	17 %	15 %	15 %	15 %	11 %
9 Mbitps	1	22 %	15 %	12 %	10 %	15 %
	2	20 %	15 %	12 %	10 %	16 %
	3	21 %	14 %	12 %	11 %	15 %
	4	17 %	15 %	12 %	12 %	17 %
	5	18 %	15 %	12 %	9 %	19 %
	ø	20 %	15 %	12 %	10 %	16 %
18 Mbitps	1	22 %	17 %	14 %	12 %	21 %
	2	21 %	17 %	14 %	12 %	21 %
	3	27 %	17 %	14 %	12 %	19 %
	4	20 %	17 %	14 %	12 %	18 %
	5	19 %	17 %	15 %	12 %	16 %
	ø	22 %	17 %	14 %	12 %	19 %

Table 5.5: Percentage captured with different packet sizes

As one can see in Table 5.5 the expectations were mostly correct. The capture rates went down as the packet size decreased, except for the minimum size of 74 byte with 9 and 18 Mbitps. It is interesting to see that these two cases both reached the seconds highest capture rates in their transmission rates even though for 2 Mbitps the rate stagnated further. There is no logical answer that can be given for this occurrence as smaller package sizes should lead to more packets being sent resulting in more overhead to be produced and therefore more if queries to be carried out.
Overall one can say that the maximum possible packet size and the highest transmission rate reached the best capture rate.

5.6 Testing UDP as a Relay

This test was designed to see how high the percentage was with which the bridging device would get the data through to the receiving end by using UDP as a relay protocol. This was done with the 18 Mbitps sending rate and a packet size of 1472 byte. The duration was five rounds each five minutes. The same testing setup as in the previous section had been used.
It was expected that most packets would arrive at the receiving end with possibly some minor losses. Further on the capture rate should have not been affected and as UDP is not connection oriented no queue should have been built up.

Round	*Capture rate*	*Relayed & received*
1	20 %	95 %
2	20 %	95 %
3	19 %	95 %
4	20 %	96 %
5	20 %	96 %
ø	20 %	96 %

Table 5.6: Capture rate and relay percentage with UDP

As expected most of the data could be relayed with only a small loss with a maximum of 5 %. This was to be expected as an unreliable data transfer was used. The only advantage by using UDP was that no queue was built up.

5.7 Relay Time

In this test, the time it takes for a packet to travel from the sending side over the bridging device to the receiving end was examined. Therefore the first extension described in Section 4.5 was used to document the time needed to travel.
Each device was synchronized with the same time server before testing with TCP and

then again before testing with UDP. The used server was *pool.ntp.org*. It is important to mention that unrooted Android devices cannot be synchronized manually. The only possibility to do so is to restart the device as after a restart the time is synchronized automatically. Each transmission protocol was used five rounds made up of five minutes. In each round 100 packets were sent every five seconds. This was done so that no queues can built up and one gets the pure transmission time. The same testing setups was used as before.
One can expect that the UDP transmission time would be lower than the TCP transmission time as it is not connection oriented.

Protocol	*Round*	*Total time*	*Acer - Android*	*Android - Asus*	*Capture rate*
TCP	1	1327 ms	902 ms	424 ms	91 %
	2	1265 ms	837 ms	427 ms	91 %
	3	1200 ms	790 ms	410 ms	90 %
	4	1124 ms	734 ms	390 ms	89 %
	5	1045 ms	682 ms	363 ms	91 %
	ø	1192 ms	809 ms	403 ms	90 %
UDP	1	823 ms	476 ms	346 ms	84 %
	2	798 ms	448 ms	349 ms	85 %
	3	757 ms	413 ms	343 ms	84 %
	4	717 ms	384 ms	332 ms	84 %
	5	669 ms	354 ms	315 ms	86 %
	ø	753 ms	415 ms	337 ms	85 %

Table 5.7: Relay Time

As one can see in Table 5.7 the expectations were met. The UDP travel time was almost half of the TCP time, which is to be expected with onesided communication. Additionally a clear trend in shrinking transmission time which is probably caused by the clocks diverging from the time server used to synchronize can be seen.
One can also notice that especially with UDP the time from the Acer to the Android device was on average about 100 ms larger than from the Android to the Asus device.

This is probably due to the capturing and relaying process with checking different parts of the protocols.
When speaking of capturing, one can see that the capture rate was included in the table as it was quite high. This points out that when sending small burst data the monitoring device seems to be more effective than with a constant data flood.

5.8 Real Life Example

This test case was performed in order to show the capabilities of extending networks in a realistic scenario by using the known setup and information. For this the software extension mentioned in Section 4.6 was used. As before the transmission rate of 18 Mbitps and the maximum packet size of 1472 byte were used.
Taking a theoretical approach, one knows that with a pre-defined capture rate on average 22 percent was achieved. As each packet has the same chance to be monitored and only two possible outcomes can be accomplished the *Binomial distribution* of probability theory can be applied. A general formula for the expected value of one is:

$$n = 1/p$$

where p is the capturing probability and n the number of repetitions to be done. Inserting our likelihood of 22 percent, being 0.22, one achieves an exact value of $4.\overline{54}$ repetitions to be done in order to gain an expected value of one capture done. As this value is almost the middle between four and five, sending four times can suffice but five times should be more efficient. This can be illustrated by rearranging the formula where e is the expected value:

$$p \times n = e.$$

With a value of four, e equals 0.88, whereas with five, e amounts to 1.10. It is important to mention that this is not taking outer interference sources such as other network traffic into account.

The image used had a size of 38308 byte which was split up in 27 packets, 26 being 1446 byte and one being 712 byte of data. Both forwarding concepts, casual and Interleaving

mentioned in Section 4.6, were tested. Additionally both repetition values of four and five were tested to verify that four repetitions would suffice but compared to five, would not be as successful. Each case got five rounds which consisted of ten transmissions and reassemblings executed. This was of course done with emptying the MyReceiverV2 array before sending again.
It was expected that four repetitions would reach an succession rate around 80 percent and five close to 100 percent due to the above mentioned expected value. Further on when using Interleaving the succession rate was expected to be slightly higher.

	Percentage fully received			
	4 repetitions		*5 repetitions*	
Round	*Interleaving off*	*Interleaving on*	*Interleaving off*	*Interleaving on*
1	70 %	90 %	70 %	100 %
2	60 %	100 %	80 %	100 %
3	80 %	60 %	80 %	90 %
4	90 %	90 %	90 %	100 %
5	80 %	90 %	100 %	100 %
ø	76 %	86 %	84 %	96 %

Table 5.8: Cases in which all different ID's were received

As one can see in Table 5.8 the expectations were rather accurate. With Interleaving enabled and five repetitions almost 100 percent and with four almost 88 percent were achieved. Further on with the casual forwarding concept a smaller succession rate was observed. This again points out that burst errors are likely to occur as with Interleaving the rate in both cases are at least ten percent higher.
It is important to mention that in all cases with five repetitions in which not all ID's were received a maximum of one was missing. When the packets were sent four times occasionally more than half of the ID's were missing.
Overall one can say that five repetitions proved to be more efficient and applying the bridging idea to a real life example is looking promising.

5.9 Summary

In this chapter, characteristics of the designed setup were displayed. First the surrounding area was analyzed to make more accurate statements about the results. Then the ideal transmission rate, being 18 Mbitps, was determined which was followed by the optimum packet size of 1472 byte. After that UDP as a relaying protocol was examined and proved to be more time efficient but considering a minor loss. In a last step, the transmission time was measured which was finished off by a successful real-life example of the transfer of an image.

Chapter 6

Summary

As described in Chapter 2 there are not many approaches which try to use a capturing system such as the AEW to connect two networks on different wavebands. So far this is a new strategy of using an Android machine as a bridging device.

In this thesis the AEW was used as a basis for extension. This framework was extended so that it can filter captured traffic by the IP address. So far only IPv4 addresses were tested, but this can easily be expanded and suffices as a proof of concept. Further on signed packets with a Magic Number sequence can be identified and depending on the chosen control bits be relayed as needed. Therefore a novel sending and a receiving application have been developed.

After the software had been prepared, different tests concerning functionality and capability were conducted. Also an everyday life example was performed to see how this concept can be applied.

6.1 Conclusion

As shown in the tests the extended version of the AEW is a good basis for expanding networks by being multi-connected. It was demonstrated that the medium transmission rates perform best as when the structure is under a high constant strain a lot of traffic

is lost by outcancelling itself or overextending the capturing device. The transmission rate of 18 Mbitps emerged as the ideal transmission rate, which was further proven in the following test. This test showed that the setup functions best with the largest packet size as less overhead was being generated. For unknown reasons nevertheless good results were achieved while using a packet size equivalent to the header size.
After that the travel time of the whole relay construction was tested and showed to be relatively low. Here as expected UDP performed better although considering a minor loss. This test revealed rather interesting results as capture rates between 84 and 91 percent were achieved. Following this an everyday life example was conducted with casual and forward error correction transmission. As anticipated, better results were achieved with Interleaving being activated. The succession rate was between 70 and 100 percent. This points out that the capturing property of the AEW functions better with occasional than with constant traffic as burst errors are likely to occur.
These results emphasize that the construction is a good starting position but especially the capturing behaviour needs in-depth analysis. The software and the Magic Number header are designed modularly and can easily be upgraded.
Overall, using the AEW as a multi-connected device for extending networks and therefore relaying data has been a success and ought to be investigated further as it offers a promising alternative for data transfer.

6.2 Future Work

In this section topics for additional research are suggested as they are out of scope of this thesis and therefore should be examined separately. The main focus is on improving the capturing capability of the AEW as this is the most crucial part of this setup.

6.2.1 Relaying Optimization Functions

A profitable extension to the AEW would be a function with which it could analyze surrounding network traffic. Based on the results the function would suggest the channel

with the least activity to be used in relaying processes. Additionally this feature could examine the Broadcast addresses and propose a suitable one.

6.2.2 Transmission Rate Deviation

While testing for the optimum transmission rate it was noticed that the actual output of the sending source exceeded the appointed transmission rate. Besides giving the blame to the hardware and its driver one should investigate this further by using different network cards and drivers to see if more precise rates can be achieved. This would be useful as more exact statements could be given to the corresponding transmission rate.

6.2.3 Workerpool

In order to improve the performance of the AEW one could replace the single relaying thread with a pool of working threads. This would probably keep the queue small and enhance the relaying process therefore improving performance.

6.2.4 Capturing Behaviour

Insightful would be an in-depth analysis of the capturing behaviour of the AEW. So far this was only analyzed as a secondary priority as it was noticed in the later testing phases. One should investigate how different time intervals between sending rhythms affect the monitoring capability.
Additionally so far the setup was only tested with one sending source. It would be interesting to study how the capturing device would behave with multiple senders.

6.2.5 Packet Size Anomaly

For unknown reasons packet sizes similar to the header size achieved a rather good capturing rate. It would be scientifically valuable to find out if the observations made were simply an anomaly. This could be done by using packet sizes smaller than the header, or even single bytes.

6.2.6 Superior Hardware

The external WLAN card Alfa AWUS036H is a rather common and basic device. Even if connected to a computer it only achieves average transmission rates. These rates suggest that either the sending and receiving performance of the card or the limitation through the chip might have been a source for only average transmission rates. One could compare different grades of cards and see if an improvement can be made.

Bibliography

[80216] *802.11 standards*. http://compnetworking.about.com/cs/wireless80211/a/aa80211standard.htm, 2016. Last checked online: 26.09.2016

[AWM13] ASADI, Arash; WANG, Qing; MANCUSO, Vincenzo: A Survey on Device-to-Device Communication in Cellular Networks. In: *CoRR* abs/1310.0720 (2013). http://arxiv.org/abs/1310.0720.

[BRM+13] BALAN, Horia V.; ROGALIN, Ryan; MICHALOLIAKOS, Antonios; PSOUNIS, Konstantinos; MEMBER, Senior: *AirSync: Enabling Distributed Multiuser MIMO with Full Spatial Multiplexing*. 2013.

[BRMP12] BALAN, Horia V.; ROGALIN, Ryan; MICHALOLIAKOS, Antonios; PSOUNIS, Konstantinos: *Achieving High Data Rates in a Distributed MIMO System*. 2012.

[GHH+10] GESBERT, David; HANLY, Stephen; HUANG, Howard; SHITZ, Shlomo S.; SIMEONE, Osvaldo; YU, Wei: Multi-Cell MIMO Cooperative Networks: A New Look at Interference. In: *J. Selec. Areas in Commun. (JSAC* (2010), S. 1380–1408.

[mag15] *UDP magic number*. https://tools.ietf.org/html/draft-herbert-udp-magic-numbers-00, 2015. Last checked online: 26.09.2016

[Men16] MENDE, Fabian: *Ein Android-Framework für Angriffsvektoren in mobilen Netzwerken*. 2016.

[ML11] MAO, Xiao H.; LEE, Yee H.: Performance of a Relay System for Two Extreme Ends of a Vessel at 2.4 GHz. In: *IEEE Antennas and Wireless Propagation Letters (Volume: 10)* (2011), S. 686–689.

[MMS+03] MEER, J. van d.; MACKIE, D.; SWAMINATHAN, V.; SINGER, D.; GENTRIC, P.: *RTP Payload Format for Transport of MPEG-4 Elementary Streams*. RFC 3640 (Proposed Standard). `http://www.ietf.org/rfc/rfc3640.txt`. Version: November 2003 (Request for Comments). Updated by RFC 5691

[mon16] *Monitor Mode defined by wireshark.wiki.org*. `https://wiki.wireshark.org/CaptureSetup/WLAN#A802.11_Filter_.28Modes.29`, 2016. Last checked online: 26.09.2016

[Pos80] POSTEL, J.: *User Datagram Protocol*. RFC 768 (Standard). `http://www.ietf.org/rfc/rfc768.txt`. Version: August 1980 (Request for Comments).

[Pos81] POSTEL, J.: *Internet Protocol*. RFC 791 (Standard). `http://www.ietf.org/rfc/rfc791.txt`. Version: September 1981 (Request for Comments). Updated by RFC 1349

[RM11] RONG, Yue; MEMBER, Senior: Simplified algorithms for optimizing multiuser multi-hop MIMO relay systems. In: *IEEE Trans. Commun* (2011), S. 2896–2904.

[Sim94] SIMPSON, W.: *The Point-to-Point Protocol (PPP)*. RFC 1661 (Standard). `http://www.ietf.org/rfc/rfc1661.txt`. Version: Juli 1994 (Request for Comments). Updated by RFC 2153

[tcp15] *tcpdump public repository*. `http://www.tcpdump.org/`, 2015. Last checked online: 26.09.2016

[YYC13] YUAN, Xiaojun; YANG, Tao; COLLINGS, Iain B.: Multiple-input multiple-output two-way relaying: A space-division approach. In: *IEEE TRANS. INF. THEORY* 59 (2013), Nr. 10, S. 6421–6440.

Abbreviations

***.apk** Android application package. *Glossary:* Android application package

AEW Android External Wifi. 2, 3, 5, 7, 8, 13, 17, 20, 21, 23, 25, 26, 28, 30–33, 41–43

API Application Programming Interface. 7, 10

CRC Cyclic Redundancy Check. *Glossary:* Cyclic Redundancy Check

GHz Gigahertz. 2, 7, 8, 11, 13, 26–28, 30

GUI Graphical User Interface. 14, 15, 17, 18, 20–22

IEEE Institute of Electrical and Electronics Engineers. 1, 9, 11, 22, 33

IP Internet Protocol. 9, 11, 12, 15–18, 32, 41, 51

IPv4 Internet Protocol Version 4. 12, 14, 17, 41, *Glossary:* Internet Protocol Version 4

JNI Java Native Interface. 8, 52

LAN Local Area Network. 28

MAC Media Access Control . 6, *Glossary:* Media Access Control

MIMO multiple in multiple out. 2, 5, 6, 8, 11

MTU Maximum Transmission Unit. 30, 51, *Glossary:* Maximum Transmission Unit

NDK Native Development Kit. *Glossary:* Native Development Kit

PCAP Packet CAPture. 7, 9, 10, 27, 28, 31

SSID Service Set Identifier. 9, *Glossary:* Service Set Identifier

TCP Transmission Control Protocol. 14, 18–21, 23, 30, 35, 36, *Glossary:* Transmission Control Protocol

UDP User Datagram Protocol. 9, 12, 14, 17–20, 23, 31, 35, 36, 39, 42, *Glossary:* User Datagram Protocol

USB Universal Serial Bus. 7, 50, 52

USB-OTG USB On-The-Go. 7, *Glossary:* USB On-The-Go

WEP Wired Equivalent Privacy . 52, *Glossary:* Wired Equivalent Privacy

WiFi Wireless Fidelity. 50, 52

WLAN Wireless Local Area Network. 1, 2, 9, 13, 23, 25, 27, 28, 30, 33, 44, 51, 52

WPA WiFi Protected Access. *Glossary:* Wireless Fidelity (WiFi) Protected Access

XML Extensible Markup Language. *Glossary:* Extensible Markup Language

Glossary

Android application package File format for installation of applications on Android. 49

Broadcast A method to transfer messages or packets to all devices on the network at the same time. 14–16, 18, 23, 26–28, 43

Checksum A small block of digital data to detect errors which may have been introduced during transmission or storage. 12, 32

Cyclic Redundancy Check An error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. 49

Fragmentation Threshold In WLAN this parameter is the same as MTU is for Ethernet. Fragments data packets if value is reached.. 30

Interleaving A form of forward error correction to avoid burst errors.. 9, 21, 37, 38, 42

Internet Protocol Version 4 The fourth version of the IPs using 32-bit (four-byte) addresses. 12, 49

Media Access Control The lower sublayer of the data link layer also know as layer 2 defined by the OSI model. 6, 49

Magic Number A constant numerical text or value used to identify a file format or protocol. 9, 10, 15–17, 20, 21, 23, 32, 41, 42

Maximum Transmission Unit Describes the maximum packet size of a protocol in the

network layer, which can be transmitted in frames without fragmentation through the data link layer of a network. 30, 49

Native Development Kit Allows part of the developed application to be implemented with JNI and native code. 50

RTL8187 Specific, commonly used by developers, WLAN chip from Realtek. 7, 22

Service Set Identifier Selectable name of a WLANs by which it is approachable. 9, 50

Transmission Control Protocol Connection oriented network protocol for reliable data transfer. 14, 50

User Datagram Protocol Connection-less network protocol for unreliable data transfer. 9, 50

USB On-The-Go A standard which enables a smart phone to act as a USB-Host. 7, 50

Wired Equivalent Privacy Encryption standard for WLAN networks, shouldn't be used anymore due to flaws. 50, 52

WiFi Protected Access Encryption standard for WLAN networks as a short-term solution between Wired Equivalent Privacy (WEP) and WPA2. 50

Extensible Markup Language Language for presentation of hierarchically structured data with good legibility for humans and computers. 50

Ehrenwörtliche Erklärung

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

Düsseldorf, 18.October 2016 Taiyou Thomas Teramachi

Diese DVD enthält:

- eine *pdf*-Version der vorliegenden Bachelorarbeit
- die LaTeX- und Grafik-Quelldateien der vorliegenden Bachelorarbeit samt aller verwendeten Skripte
- die Quelldateien der im Rahmen der Bachelorarbeit erstellten Software MyServer, MyReceiver, MyReceiverV2 and Android-External-Wifi
- den zur Auswertung verwendeten Datensatz an Aufzeichnungen
- die Websites der verwendeten Internetquellen

- ein Readme.txt mit Erläuterungen zu den Inhalten der DVD