Network Light Control Protocol

Design and study of a novel real-time network protocol

Networks and Distributed System
Kai Benjamin Heinz - 10th semester

Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7
DK-9220 Aalborg

Department of Electronic Systems
Fredrik Bajers Vej 7
DK-9220 Aalborg Ø
http://es.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Network Light Control Protocol
- *Design and study of a novel real-time network protocol*

Theme:
Master Thesis

Project Period:
10^{th} semester
Networks and Distributed System
February – June, 2014

Project Group:
14gr1023

Participants:
Kai Benjamin Heinz

Supervisors:
Jimmy Jessen Nielsen
Lars Mikkelsen

Copies: 4

Page Numbers: 149

Date of Completion: June 3, 2014

Abstract:

This project designed a new network protocol to control stage units like lamps, moving heads and fog machines. Related protocols had been discovered to get an idea of the current market situation. Based on this knowledge a problem had been formulated, that states the limited address space and cable connection as the most challenging topics. The cable connection had than been replaced by a wireless 802.11g connection. The two most interesting measurements for this wireless network are range and number of lights. To find these limits a simulation had been programmed in OMNeT++ and a field test had been performed. A controller had been developed in Java and an adapter had been constructed to transform from this new to the DMX protocol. This way current stage lamps that do not understand this new protocol could be driven. The result of this project is a new stage light control protocol designed for wireless communication.

Preface

This is a report of a student project, on the 4^{th} semester of Networks and Distributed Systems, at Aalborg University. This project serves as a Master Thesis for this programme. The topic of this report is to develop a novel network protocol to control stage units like lamps, moving heads, fog machines, and so on.

The work is based on current standards, like DMX, and compared to it. Besides design, a simulation in OMNeT++ had been performed to find the maximum number of units in the network. A field test had been done to find the maximum range. A sample controller and receiver had been built.

The work had been supervised by a professor and PhD-student from Aalborg University, who did a great job and helped a lot when work got difficult and leaded me straight to the finish line.

Aalborg University, June 3, 2014

Kai Benjamin Heinz
<kheinz12@student.aau.dk>

Contents

1 Introduction and Motivation

This chapter shall provide an initial overview of stage lights and gives background information for the reader in order to classify this work. The introduction will focus on a very general description of stages and how the are lighted. This will be done by presenting a little bit of history, which will lead from ancient, over past to modern times. During this review the used technologies for stage lights will be presented and how they had been evolved. A special focus will be set upon current technologies and the area of currently used communication protocols, their drawbacks and why further development is needed. As said, this is only a very brief overview to give an understanding of the working area. The motivation section shows the personal, technological and the scientific interest in the development presented in this work. Lastly the chapter will be concluded and main points will be recapitulated. All the information will be used in order to present one main-question, which will be used as a guideline throughout this project and will finally be answered in this report.

1.1 Introduction

Since centuries mankind used stages to present and perform acts in front of an audience. The roots are going back to the ancient Greek culture [6] and further. From these first days, stages had been built in- and outdoors. Both had their advantages and disadvantages but at some point they had a common problem: Actors need to be presented in proper light. While outdoor stages had no problem solving this issue during day, it became a problem in the dark hours. For indoor stages solutions had to be found even earlier, for example while planning the building, which needed a lot of knowledge about architecture and natural sciences. The earliest solution people found is also probably the most dangerous one: open fire. Candles and gas lamps are the mostly known representatives of this group. The danger of this technology could be seen in the "Ringtheaterbrand" (Ringtheater Fire) where on December 8, 1881 the Ringtheater in Vienna burnt down and 384 people perished [12]. During the same time a new technology for lighting came up by the inventor Thomas Alva Edison. He was not the first one to show that electricity could be used to made materials to emit light, but he improved the technology a lot what lead to a patent in 1880[10]. The rise of electricity and electric lights was a large step towards a safer, more user friendly stage lighting technology.

Today's Stage lights are highly specialized devices improved for robustness, weight and easy handling. Stage lights may consist of a single light source like a high power light-bulb

or hundreds or thousands of small Light Emitting Diodes (LED). Beside the 'old' task of providing plain light to the stage, modern devices allow a stage director to choose from a variety of colours, brightnesses, angles and forms. However, the available functions may differ heavily dependant on the actual specialization of the device. While ancient lamps just needed to be enlightened by a stage worker (dependant on the lamp by fire or by switching them on), modern lamps need beside a ground power supply, a communication interface to send and receive control signals from a remote controller. This controller can be a special light control board or a PC/Laptop, which is running a specialized application. The connection to the lights is usually realised by a DMX-bus system. While controllers are already specialized to send out DMX signals, a PC often needs an adapter, which can be as small as a USB-Memory stick. The commonly used DMX offers advantages and disadvantages, which will be discussed in the coming section.

1.2 Motivation

DMX is the de-facto standard in inter stage light communication. Its abbreviation stands for **D**igital **M**ultiple**x** and had been standardized in 1986 [7] for the first time with the goal to provide a robust, yet easy to handle network protocol to control stage lights. At this time the focus of the developers were to control dimmers, which are simple, single coloured lamps. Today, however, the market provides lamps that can do much more than shine or not shine. It is possible that a light changes its colour, brightness, focus or even the angle in up to 3-axes. To control all these features a single lamp uses more than one byte of data, which is the maximum amount of data provided for a single channel in DMX.

A channel can be seen as the address of a lamp or one of it features. DMX had been defined to support 512 channels, this is why it is officially called DMX-512. To give an example of how channels are used in a stage lamp, a simple RGB-lamp will be taken as an example. This device has obviously one red, green and blue channel. So at least three channels are already used. On top a master and strobe channel are assumed to be available. The master is used to set a maximum value for all channels, so that the amount of available red, green or blue values will increase from 2^8 to 2^{16} values. This allows for a finer control of brightness by the cost of one channel. The Strobe channel is used to provide flashing lights. The value received on this channel by the lamp will indicate the frequency the lamp has to flash at. This is convenient, since the lamp can independently set (and adjust if new values are received) the frequency of the strobe, which allows the controller to use its computational power for other tasks. Together this simple lamp uses five channels. Scanners[†], moving heads[‡], light panels and other devices designed for use in light shows are much more complex, so that one of these devices can have easily more

[†]A scanner has a single light source, which beams onto a mirror, that is attached at an angle to a motor. By rotating the mirror the light beam moves around the room. Usually different light shapes, so called gobos are provided as well as different colours.

[‡]'Moving heads' have a special mounting of the light source, allowing to move the light beam to nearly every position, covering full 360 ° in 3-axes. Like scanners different shapes and colours are provided.

than 50 channels[†]. Figure 1.1 shows a modern concert where moving heads are used. The pure amount of devices shows that a limit of 512 channels is not longer acceptable.

Figure 1.1: Lightshow during the concert "The Voice of Germany" January 7 - 2014, Schleyerhalle Stuttgart, © mpc/Michael Brabender

Beside the limitation of channels, DMX is specified with a special cable on layer one. Connection of lamps is done via the daisy chain principle, meaning that every device has two connectors. One for input data, the other one for output data. The output port of a lamp will be connected to the input port of the next lamp and so on. In this chain the controller is meant to be the first device, so it does not need any input connector. Hence, it is expected to forward every piece of data that arrives at an input port, the last lamp in a chain has to terminate the control line with an end-resistor. The advantage of this bus system is that every device receives all the data and can filter it through its own rules (e. g. the pre-programmed channel number). This way all the lamps are synchronized and always have the latest values available. The down side of this approach is a practical one. Cables are heavy and for a big concert a huge amount of cables are needed. Beside the cost of transportation, the length of this cable-connection is limited, meaning that it can become very difficult to distributes lamps in a large area. Hence transport and installation has to be done regularly, it would be much more efficient to replace the cable with a radio connection. Beside lowering the costs of transportation and installation (basically no time needed for connecting lamps), the health of the stage worker might be protected by lifting less weights.

One example of an event where long distance cabling was needed will be shown in figure 1.2. It is a small stage on a Christmas market in a town in central Germany[‡]. Compared

[†]ELAR QUAD PANEL™from Elation Professional uses between 4 and 64 channels, dependant on the operating mode.

[‡]City of Kreuztal in the state North Rhine-Westphalia. GPS: 50.961728 N, 7.988142 E

to the scenario from figure 1.1, it can be seen that much less lights are used, so addressing becomes a secondary role in this case. However, the stage itself is only half of the light concept, where it had been planned to support the stage with lights distributed off-stage to create a nice, Christmas-like feeling. Figure 1.3 displays an example of the distribution of lights. It might be pointed out that the stage cannot even be seen from this point, indicating how far away they actually are. The problem addressed here is the one of large distances. For long distances, meaning heavily distributed lights also long cables are needed, which needs to be transported and installed. Again a wireless solution would be more efficient and easier to deploy.

These two basic difficulties are actual problems in real light applications. Technicians developed workarounds in the last years to handle it somehow. However, the problems are not gone and will probably increase over the next years. This calls for development of new network protocols especially for the lighting industry. Replacing the physical wire by a wireless radio communication yields for advantages by lowering costs of transportation, installation and maintaining the light set-up. The health of stage workers might be protected and less resources are needed. This master thesis will therefore discover existing protocols and propose a solution for these practical problems.

Figure 1.2: Stage on a christmas market, December - 2012, Dreslers Park, Kreuztal, Germany, © kreuztalkultur/Alexander Kiß

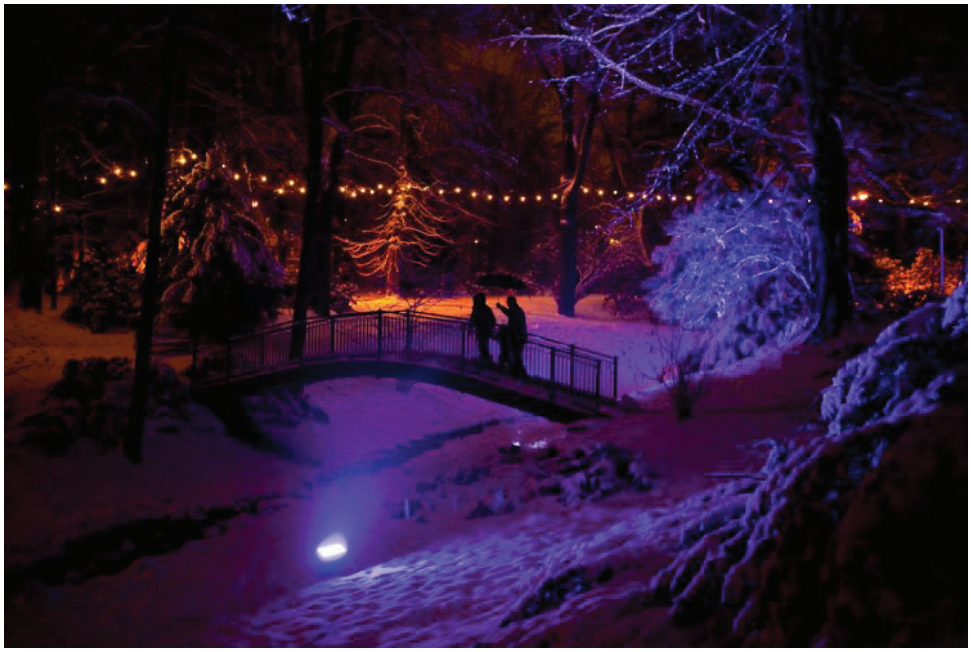

Figure 1.3: Surroundigs of a christmas market, December - 2012, Dreslers Park, Kreuztal, Germany, © kreuztalkultur/Alexander Kiß

1.3 Conclusion

The introduction gave a brief history of stages and how important stage lights are. Lamps were originally used for the purpose of just producing plain light. In the last century this role changed from a simple light source to a high-tech device that is able to change colours, brightness, angles, and so on. It had been shown that modern lamps are connected to each other in order to receive orders from a central controller. DMX is the de-facto standard in this working area; standardized first in 1989 [7] it is not longer the optimal protocol for stage lighting. The small address space of 512 available addresses is one problem of DMX. The fact that physical wires are needed seems to be not optimal, especially in large light applications as shown in figures 1.1, 1.2 and 1.3. A wireless solution might be more flexible and cheaper to transport. The given example of the Christmas-market will therefore be used throughout this report as a sample set-up to evaluate protocols. In the end of this thesis a solution will be provided to solve these common problems (see chapter 3) that occur with current protocols.

2 Related Protocols

In this project research is made in the field of stage light protocols. The first step in this process is to look around and find related protocols already available. This section covers the three most important network protocols for stage light applications and show their advantages and disadvantages. This will lead to a good understanding of what is currently done and where are the main points which can/should be improved in a newly proposed protocol. The three protocols that are examined are DMX, W-DMX and ArtNet. The only standardized protocol is DMX, however, the others are also used and built a de-fact standard in the lighting industry. The conclusion will compare the results of the independent chapters and will form the basis of the further work.

2.1 DMX-512

DMX-512, often just called DMX, is a standard in the stage lighting industry. It is defined as a network protocol containing a physical and application layer. DMX can nowadays be found in many lights and a lot of DIY-projects have been developed around it because of its simplicity.

DMX had been initially developed for controlling dimmers, which are drivers for uni-coloured stage lights. As technology developed, a single control signal could not deliver enough information to control all the functionalities of these modern lamps. This problem had only been partially solved and is still today one of the major challenges in stage lighting.

DMX had been developed in the USA and was standardized by the 'United States Institute for Theatre Technology' (USITT) in 1989 [7]. The name USITT:DMX512 had been chosen, which refers to the technology (Digital Multiplex) and the address space of 512. Since than a number of other authorities standardized this technology as well. In 2000 The German Institute for Standardization (German: Deutsches Institut für Normung e. V., short: DIN) set up the standard DIN 56930 (Bühnenlichtsysteme, English: stage light systems), which will be used throughout this report. This seems legal since all devices labelled regarding to [4, sec. 5] are interchangeable, and therefore this standard is as good as any other.

2.1.1 DMX Devices

DMX Controller

A DMX Controller forms the heart of a light fixture. Its main task is to convert a user input into DMX format and send it to the lights. The connection is done by a special cable

connection described in [4, sec. 4.2]. Being more specific a controller is a device a light technician or another trained person works on in order to control the appearance of the light fixture. This can be done by pressing buttons and dragging faders on a physical light board or by setting up a light show in a special software on a computer/laptop. Throughout the report it is not important what actual controller is used, as long as it generates an output signal that complies to DMX-512. A DMX Controller has one physical connector to the DMX network called DMX out.

DMX Devices
Most other devices in a DMX network are lamps or lamp alike devices able to display light in any form. However, also other devices, like fog machines might belong to this group. A DMX device has 2 physical connectors (input and output) which connects the device to the DMX network. This is done by the daisy chain principle, so that the output port of one device is connected to the input port of the next one, and so on. The end of a DMX control line must be terminated. Every DMX Device can be addressed by a number between 1 and 512, what is usually called a channel. If a device uses more than one channel, only the first address will be set. The other channels are continued from this point on.

Example: A common RGB-lamp has 5 channels, one for each colour plus master channel and flash light. The lamp is programmed to channel 8, which leads to the following adddresses: Red: 8, Green: 9, Blue: 10, Master: 11, Flash light: 12.

It is often sufficient that one property belongs to one channel, like in the example above. However, it might happen that some properties need a finer data granularity. In this case it is possible to use more than one channel for this property. This is the case for moving heads where for a full 360 ° 256 values result in a minimum resolution of $\frac{360\,^\circ}{256\text{ steps}} = \frac{1.41\,^\circ}{\text{step}}$. This can be unsatisfying and can only be solved by using a second channel ($2^{16} = 65536$), giving a much higher resolution of $\frac{360\,^\circ}{65536\text{ steps}} = \frac{0.0055\,^\circ}{\text{step}}$.

Other DMX Devices
Terminator: Every DMX control line must be terminated with a 120 Ω/0.25 W resistor. However, many modern lamps can recognize if they are forming the end of a line and if so set up a terminating resistor be itself. No need for an external terminator is given then. The details are described in [4, sec. 4.2.2.4].

Splitter: A Splitter is used to build an intersection in a DMX line. This is sometimes useful for literally split the signal into two dependant lines. It must be mentioned here, that the incoming DMX signal is not changed in any regard. It is just copied and send to two or more outputs. A splitter is not explained in detail in [4].

Repeater: A DMX line is restricted to a certain length, which cannot be exceeded. The control line also limited to a number of attached lamps, which is much less then 512. The restriction is due to a voltage drop on every lamp. To overcome this problem a repeater can be used to amplify the control signal. Like a splitter, the output is an exact copy of the input, meaning that the DMX signal is not modified in any regard. A repeater is described in [4, sec. 4.2.2.3, image 4].

2.1.2 Network Architecture

DMX operates on a field bus. Every device (controller, lamp, terminator) is connected to this bus. The most basic network structure can be seen as a line, starting at a controller going through all the lamps and ending at a terminator (see figure 2.1). This kind of connection is called daisy-chain, since the control signal is looped through every single device until the end. If this line exceeds a certain length a repeater must be included. This length is 1,200 m [4, sec. 4.2.2.5] or 32 lamps [4, sec. 4.2.2.6].

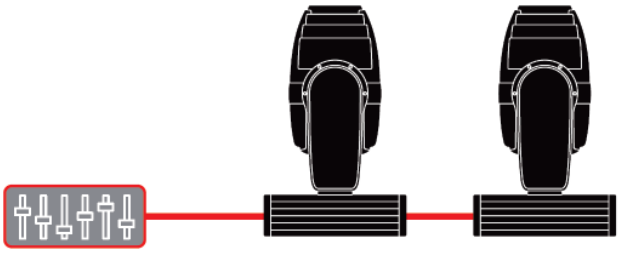

Figure 2.1: Sample DMX network, basic line architecture. The terminator is not shown and should be attached at the last light.

If a splitter is used, the line architecture shown in figure 2.1 changes to a real star network (see figure 2.2). As described above, the splitter just copies the signal and makes it possible to increase the number of control lines. This might be very handy in some applications where otherwise much more cable must had been installed. However, the splitter does not create a new address space, so that still all restrictions apply like in a simple line architecture. There are no restrictions on where, or how many splitters are installed.

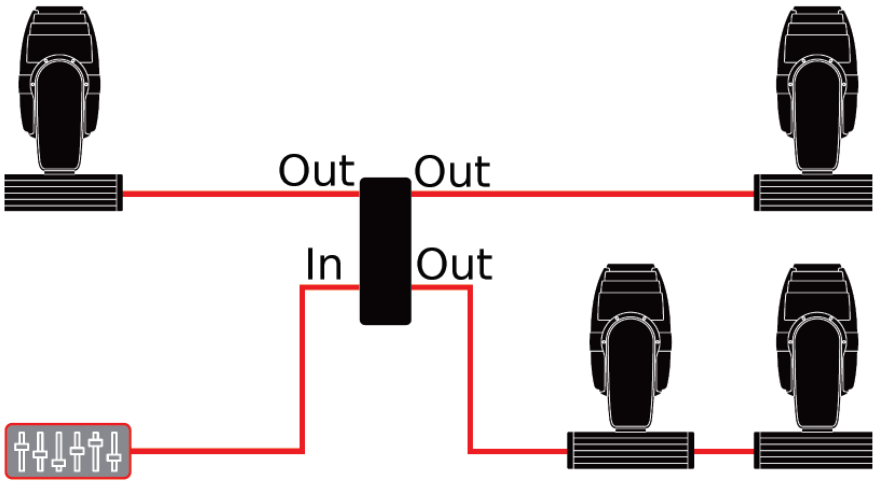

Figure 2.2: Sample DMX network with a splitter (black box) in the centre. Star architecture

Pin	Description
1	Ground (GND)
2	Data channel 1, negative potential (D1-)
3	Data channel 1, positive potential (D1+)
4	Data channel 2, negative potential (D2-)
5	Data channel 2, positive potential (D2+)

Table 2.1: Connections of a 5 pin XLR plug.

2.1.3 Physical Layer

DMX relies on the EIA-485 standard, also known as RS-485 [8]. It describes how the data are transmitted within the wire and what physical properties the wire has to fulfil. Without going to deep into the specifications of RS-485, it shall only be mentioned that the signal is transmitted in two data lines, where the signal is transmitted on one line and the inverted signal on the other line. The advantage of that is that even if the voltage drops at any point in cable, the potential between the two lines will stay the same. This makes the physical layer more robust. In a DMX network at least 3 lines must be used in a cable. Beside the two data line just mentioned there exists a common ground signal. The standard also defines a second pair signal lines, but they are marked as optional [4, table 2]. Table 2.1 gives an overview of the signals used in a DMX wire.

Beside the wire a common connector is described in [4]. It is usually called an XLR connector and is available with 3 and 5 pins. It must be mentioned here, that even if 3 pin connectors are available and often used in stage craft, they are not officially allowed. [4] explicitly asks for 5 pin plugs.

Because of the relation to RS-485 some restriction occur that must be fulfilled in any DMX application. This is that on a single DMX line as seen in figure 2.1 a maximum length of 1,200 m cannot be exceeded [4, sec. 4.2.2.5]. If the application needs longer control lines, a repeater must be included. The same applies for the amount of DMX devices listening on a single control line. If more than 32 devices [4, sec. 4.2.2.6] are connected, the signal must be amplified in order to guarantee a proper function. A splitter can be used if no repeater is available.

2.1.4 Data

Data sent over DMX control line must agree on two points. First, they have to follow a certain format, which will be explained next. Second, all devices in a DMX network must have the same understanding of how long a bit is. The standard DIN 59630-2:2000 has defined the length of 1 bit as 4 μs [4, sec. 4.4.1]. This gives a data-rate of 250 kbit/s meaning that 250,000 bits can be sent per second.

The basic format of a DMX packet can seen in figure 2.3. By looking closer into this packet it can be seen that no channel numbers are transmitted. They are given implicitly by their position of the data in the bit-stream. However, a packet starts with the **BREAK** signal that clears the control line and gives the DMX devices the chance to prepare for the coming signals. The BREAK signal is sent as a logic zero and must be at least 88 μs

long (this is the length of two data words). An upper time limit is not defined. For DMX devices this signal means that every pending operation must be cancelled.

The **Mark After Break (MAB)** separates the BREAK from the start code (SC). It is not clearly defined how long this signal is, but it must be between 8 μs and 1 s. Devices that can handle a shorter 4 μs MAB signal are allowed to add "(4 μs)" to the signature, like "DMX512/DIN(4 μs)" [4, sec. 5].

The **Start Code (SC)** is the first information that is actually delivered (see figure 2.4). It consists like the channel data of one start bit, which is transmitted as logic zero, the actual data byte which is defined as 0x00 [4, sec. 4.3.4.1] and two stop bits, transmitted as logic one. The start code 0x00 identifies the following data as light values and shall be interpreted from a DMX device accordingly. The codes 0x01 to 0xFF have been declared for future use. If a DMX device receives any start code different than 0x00 it is not allowed to treat the following data as light values [4, sec. 4.3.4.2]. This shall prevent the lights from displaying data that are meant to be displayed[†]. Altogether 11 bits are transmitted, which makes it 44 μs long.

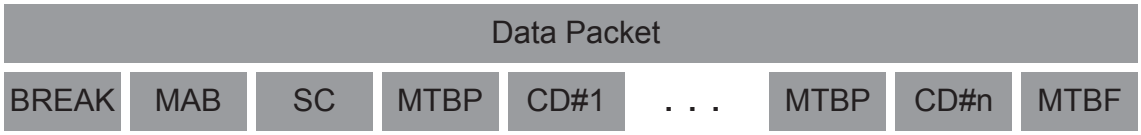

Figure 2.3: Structure of a DMX data packet. It is not mandatory to send all 512 channels so n can be chosen to be $1 < n \leq 512$ [4, img. 5]

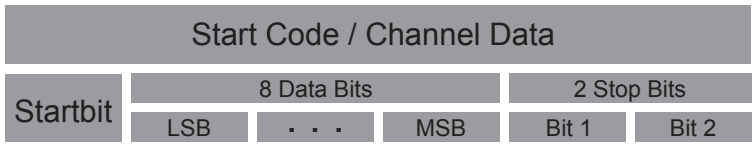

Figure 2.4: Structure of DMX start code (SC) and channel data (CD). Total of 11 bits: The actual data are represented by 8 bits.

Channel Data (CD) have the same structure as the start code, which is shown in figure 2.4. The difference lies in the transmitted data. These data are describing the value of a certain DMX channel and must be interpreted accordingly. Here it becomes visible that per channel only one value can be transmitted in a range between 0x00 and 0xFF, which is 256 values. If a greater range of values is needed it is possible to split the signal to two or more channels as described in the section DMX Devices. This, however, reduces the available channels.

Every data frame (start code or channel data) is followed by a **Mark Time Between Packets (MTBP)** signal (except the last frame, which is followed by MTBF). This signal

[†]Amendments of this standard define start codes different from 0x00, which are used, for example, for remote device management (RDM) and shall not be interpreted as light signals. RDM is not covered further throughout this report.

is very generic and can be transmitted as 0 up to 250 kbit, which leads to a delay between 0 and 1 s. If it is transmitted it must be sent as logic one. Usually this signal is not used [4, table 1].

For the **Mark Time Between Frames (MTBF)** applies the same as for the before mentioned MTBP. It can be sent as 0 up to 250 kbit. If the size is larger than 0, it is required to send it as logic one. However, the BREAK signal will be transmitted as logic zero so that the beginning of a new data packet can be identified.

Like figure 2.3 shows, the number of supported channels is limited to 512. A channel can be seen as the address of a DMX device. If more addresses are needed it is possible to set-up a second DMX-line (or even more). Since the control lines are independent, every line has its own address space, which is often called a DMX universe. The number of universes is obviously not limited because of the independence of each other.

Devices that comply to the standard DIN 56930:2000 are allowed to carry the signature "DMX512/DIN", "DMX512/1990" or "USITT DMX512/1990". If a device can handle short MAB signals the note "(4 μs)" can be added: "DMX512/DIN(4 μs)", "DMX512/1990(4 μs)", "USITT DMX512/1990(4 μs)".

2.1.5 Summary

DMX is a standardized protocol, designed for controlling stage devices such as plain lamps, scanners, moving heads and so on. Every DMX device is connected to a control line which fulfils the requirements of RS-485. The line starts at a DMX controller and is looped through all stage devices. The end of the control line is terminated with an end-resistor. The line itself cannot exceed a length of 1,200 m and cannot host more than 32 light units. If the number of devices or the length of the line is exceeded an amplifier has to be included. It is possible to split the signal line into several lines. This is useful in some applications and might save wire. The architecture of a DMX network is star based, but in can be reduced to a line architecture. The data send over a control line are introduced with a start code (0x00) and followed with a maximum of 512 bytes representing the data of the individual channels. One start-bit and two stop-bits are transmitted to signalize the start and end of a data byte and start code, respectively. The data rate of the bus is set to 250 kbit/s by the standard. A full data packet with 512 channels (assuming that MTBF and MTBP is as small as possible) has a length of 5667 bit[†]. This leads to a maximum of 44.1 DMX packets per second. In other words a refresh rate of 44.1 Hz. Beside the actual data no bits for error correction are transmitted, which can lead to corrupted data. Since only one device can send data to the network there is no need for collision avoidance/collision detection. A DMX network is restricted to 512 channels. If a greater amount of channels is needed a second, independent network can be set-up. A network is also known as a DMX universe.

[†]BREAK + MAB + SC + 512·CD = 22 bit + 2 bit + 11 bit + 512 · 11 bit = 5667 bit

2.2 W-DMX

W-DMX is a proprietary protocol maintained by Wireless Solution Sweden AB. The company had been founded in 2003 by top professionals from the lighting industry and wireless communication engineers [13]. Due to the proprietaryness, it is not possible to get technical details of the network protocol, others than they offered through their marketing brochures and on-line. However, Wireless Solution calls W-DMX itself an "unofficial standard" [13], which shows a need for discovering the technology in more detail. The following details will show what principles are used in order to get a good picture of the technology. Details cannot be provided as long as they are not freely available. Even if the marketing material has a big focus on hardware like connectors, etc. The focus of the coming paragraphs are on the technical details, meaning working principles, and not on physical restrictions like connector plugs or similar.

2.2.1 Network Devices

The term W-DMX covers two aspects. The first one is the W-DMX network protocol which is used for transmission and of most interest. The second one is the special hardware that is sold by Wireless Solutions. Both aspects belong together since the network protocol has to take physical restrictions into account. To get an image of what W-DMX can do both sides will be shown but the focus will be set to the upper layers of the protocol. However, this section will give a brief overview of the devices in a W-DMX network.

W-DMX Transmitter
The W-DMX Transmitter receives a DMX signal from a DMX line and sends them wirelessly to its connected receivers. The incoming data must be processed to form the desired package. One part of this processing is called marshalling, which orders the received data regarding to a certain structure. It will be discovered later in this chapter that the incoming data are processed in a way that an exact copy of the incoming signal can be regenerated at the receiver. Some additional data will be included to allow error correction in the receiver. The transmitter is displayed as a black-box with an antenna, labelled as "Tx" in the following images.

W-DMX Receiver
A W-DMX Receiver is the counterpart to the W-DMX Transmitter. One or more receivers will be connected to one transmitter in order to receive DMX data over air. Note here that multiple receivers can be attached to one transmitter but every receiver can only belong to one transmitter. This is why the network principally looks like a star-network with the controller/the transmitter as centre-point. The received data are processed in order to retrieve the DMX values and corrects them if necessary. From the received data a DMX signal is created and send over a DMX line to the light fixtures. A receiver is displayed as a black–box with an antenna, labelled "Rx" in the following images.

W-DMX Repeater
W-DMX Repeaters are a combination of the two device mentioned above. It is connected to a W-DMX Transmitter like any other receiver. The difference is that no DMX signal

will be created, hence the received data will be sent to another receiver or transmitter. This allows the bridging of longer distances or obstacles. Because a repeater contains a usual transmitter multiple receivers can be attached. repeaters will not be used further in this chapter, therefore no image description will be given.

2.2.2 Network Architecture

W-DMX is designed in a way that it is compatible to DMX-512. The main advantage is the wireless communication that replaces (at least) one cable connection as shown in figure 2.5 and called **Point to Point operation mode**. This wireless transmission is very useful in concerts where the light operator with the light controller is located in front of the stage and has to bridge the crowd of people between him and the stage somehow. Like the image shows, the light controller is connected to a W-DMX transmitter (using a standard DMX connection) and the receiver on the other side is connected to the light fixtures (again with DMX). This leads to the first observation that W-DMX just acts as a wrapper for raw DMX signals. It could be seen as an alternative physical layer for the data.

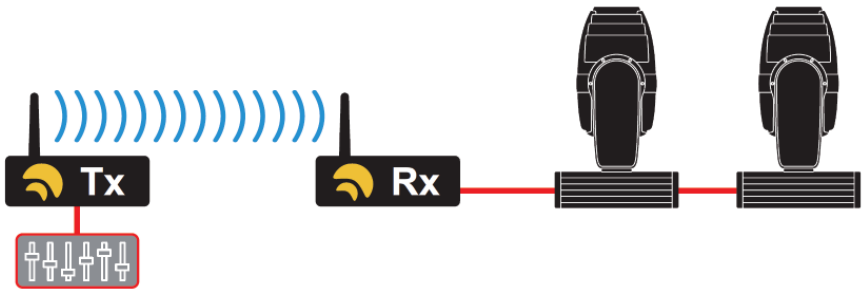

Figure 2.5: W-DMX Point to Point Operation [14]

Beside from this, a **Point to Multipoint operation mode** is available as seen in figure 2.6. The set-up is similar to the one before, but a second receiver is listening to the same radio channel. In [13] this behaviour is described as a “big, virtual DMX splitter”, where the number of receivers is not limited. This means that light fixtures can be freely distributed, as long as they are in range of the radio signal. The wired connection of the individual lamps is from that point on bound to the restrictions of DMX. This network architecture is star based, meaning that all the communication is initiated from the controller and than forwarded through W-DMX to its destination. This is generally nothing new, since in DMX has the same network set-up. However, in DMX a special Y-connector, also called splitter, is needed to perform a similar operation. The observation here is that again W-DMX is not offering new technology, but just replacing the wired connection. The physical properties of a wireless system, however, allows for less network devices.

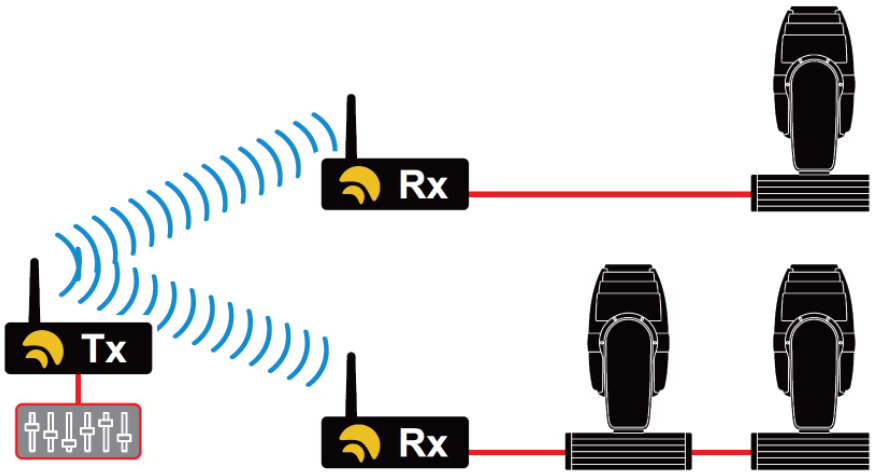

Figure 2.6: W–DMX Point to Multipoint Operation [14]

The third way W-DMX can operate is in **Multipoint to Multipoint mode**. Figure 2.7 is showing the working principle. It can be easily seen that this is simply an application of the Point to Multipoint operation mode and more a marketing gimmick than a technical detail. By using this kind of operation mode two controllers are used, meaning that two DMX universes are operating simultaneously, which increases the number of available channels for every transmitter by 512. On the other hand this set-up allows to infer that W-DMX is actually just transmitting the 512 channels of DMX and does not add any further data (except error detection, which will be described next).

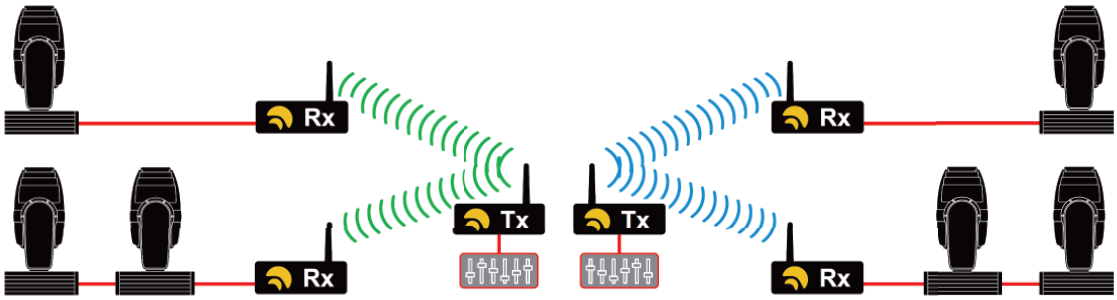

Figure 2.7: W-DMX Multipoint to Multipoint Operation (base image from [14])

2.2.3 Robust Data

Wireless Solutions promotes W-DMX with the term “Data-Safe” [13] to allow the correction of broken data in a receiver. The actual method is not described, only that it is patented [13]. It is also mentioned that any corrupted data will be corrected before forwarded. This seems like they are using an error correction algorithm, meaning that beside the DMX data additional information must be provided to do error correction. Doing this seems like a very good idea and should be considered in the proposed protocol that

will be described later in this report. However, error correction works always just up to a certain limit. If data is totally destroyed or a packet gets lost this technique is not longer providing the desired result.

2.2.4 Wireless Data

Another feature used for promotion is called "Invisi-Wire", described as the "industry's true wireless replacement for a wire" [13]. It is mentioned that all DMX signals are recreated. That is not just the actual light-data, but also all the DMX marks like "break, mark after break, interslot timings and slot count parameters" [13]. This gives another insight into the W-DMX system, namely that an exact copy of the incoming DMX signal is created, transmitted and recreated. This information is essential in a way that the actual network protocol that they use (which is not published!) can be reverse-engineered.

2.2.5 Wireless Communication

For wireless communication a radio frequency must be chosen and hardware must be developed around this. W-DMX uses the 2.4 GHz and 5.8 GHz band [13]. These bands are freely available for industrial, scientific and medical use and are therefore refereed to as ISM-band [2]. In [14] the frequency of the 2.4 GHz band is given in more detail, saying that "2402 to 2479 MHz" are used for transmission. The same source mentions that the output power is restricted to 100 mW up to 300 mW[†]. More precise information about the 5.8 GHz band that W-DMX uses had not been found. The advantage of using the ISM-band is that it can be used worldwide without any license[‡]. On both bands (2.4 and 5.8 GHz) "adaptive frequency hopping (AFHSS)" [13] is used. It is further described as "interference free" meaning that "W-DMX will not be interfered with, or cause interference with other radio systems [...]". It might be questioned how true this statement is, since wireless-routers and mobile phones are using the same frequency spectrum. This observation is also made in [13], saying that the 5.8 GHz band is preferred in indoor applications[§]. 5.8 GHz might be not available everywhere due to national restrictions.

The technical information given here are interesting, since they comply, at least in parts, with the IEEE 802.11-standard. However, Wireless Solution does not say that they are actually using this technology standard or not.

2.2.6 Communication Delay

The DMX signal that is created at any receiver will comply with the specifications of DMX-512. This includes a full 512 channel packet, a refresh rate of 44 Hz and a maximum number of 32 devices on a DMX line [13]. The number of W-DMX Receivers for a single W-DMX Transmitter is not limited [13, 14]. It seems obvious that a wireless connection has some kind of delay. Just from a physical point of view it is not possible to send anything over a distance without any time delay. The brochures are talking a little bit around this topic and goes from "without any delay" down to "virtually no delay" [13].

[†] "Only in FCC output power mode." [14]

[‡] Restriction may be made in some countries.

[§] "[...] when additional [...] protection against interference is required, the 5.8 GHz band is the ideal choice. 5.8 GHz is also the best choice when operating inside TV Studios or Live Recording environments [...]." [13, p. 42]

How long "virtually no delay" is, in terms of seconds, is not mentioned. However, this value might vary depending on the set-up infrastructure. Especially when it comes to large distance fixtures where a lot of repeaters are needed this delay can sum up to an observable time. The example in figure 2.8 will show a sample application with a high level of delay caused by W-DMX. This set-up is legal, because a transmitter does not care about who creates the DMX signal, it just sends it to its connected receiver. The delay which is generated during transmission, as well as for marshalling/un-marshalling accumulates over time. The time needed until all data are received by the first lamp can be expressed as

$$\begin{aligned}\Delta t_1 &= t_{\text{transmission}} + t_{\text{marshalling}} + t_{\text{unmarshalling}} + 2 \cdot t_{\text{dmx}} \\ &= 1 \cdot (t_{\text{transmission}} + t_{\text{marshalling}} + t_{\text{unmarshalling}} + t_{\text{dmx}}) + t_{\text{dmx}}\end{aligned} \tag{2.1}$$

where $t_{\text{transmission}}$ is the time needed to send the data from a W-DMX transmitter to its connected receiver(s), and t_{dmx} is the time needed for transmitting the data over a DMX line. The latter has to be used two times, simply because the two DMX lines are used (controller to transmitter and receiver to lamp). $t_{\text{marshalling}}$ and $t_{\text{unmarshalling}}$ are the times needed for marshalling and un-marshalling the data in a transmitter/receiver. This value could be equal to zero, if the received bits are directly put in the right position and not buffered before. However, a delay can occur here and should therefore be mentioned. The calculation of the delay time is an assumption and can be seen as worst-case. It is used anyway to demonstrate how the delay could accumulate since no better formulae is provided by wireless solutions. So it might be possible that Δt_1 is actually less or equal to the time calculated here.

The interesting thing here is that the time of the DMX packet has to be counted twice. This is because it is sent from the DMX controller through the wire to the transmitter, which cannot operate before the full packet is available, what can be justified by the assumption mentioned right before. The same counts for the receiver, who has to wait for the whole packet to arrive to construct a new DMX packet. By adding a light to the control network from figure 2.8, the time for transmission is increased by $\Delta t_1 - t_{\text{dmx}}$. The term $-t_{\text{dmx}}$ comes in because of the assumption that DMX line is looped through the lamp and no additional delay occurs here. With this in mind a general form n lights can be given as

$$\begin{aligned}\Delta t_n &= \Delta t_{n-1} + \Delta t_1 - t_{\text{dmx}} \\ &= n \cdot (t_{\text{transmission}} + t_{\text{marshalling}} + t_{\text{unmarshalling}} + t_{\text{dmx}}) + t_{\text{dmx}}\end{aligned} \tag{2.2}$$

The most interesting question is how large the delay between the first and the last light fixture is. This can be easily calculated by

$$\begin{aligned}\Delta t_{\text{large}} &= \Delta t_n - \Delta t_1 \\ &= nT + t_{\text{dmx}} - (T + t_{\text{dmx}}) \\ &= nT + t_{\text{dmx}} - T - t_{\text{dmx}} \\ &= nT - T = (n-1)T\end{aligned} \tag{2.3}$$

where $T = t_{\text{transmission}} + t_{\text{marshalling}} + t_{\text{unmarshalling}} + t_{\text{dmx}}$. Since there are no information available about the actual times needed for marshalling and un-marshalling, as well as

for the transmission (in the above calculations it had been assumed that the $t_{\text{transmission}}$ is equal for every transmission), only this abstract description of the delay can be given. However, it can be seen that the delay between the first and last light is depending on the number of lights. This is not a problem as long as it can be guaranteed that $\Delta t_{\text{large}} < \varepsilon$, where ε is the time that a human being is able to recognize as a delay. However, there is no official limit for n and therefore one could potentially create a network where the delay sums up to a recognizable time, without having the intention to do this.

Even if this situation seems to be a little bit theoretic, it might be become a real problem when lighting a larger area like a whole city or a long ski slope observed from distance.

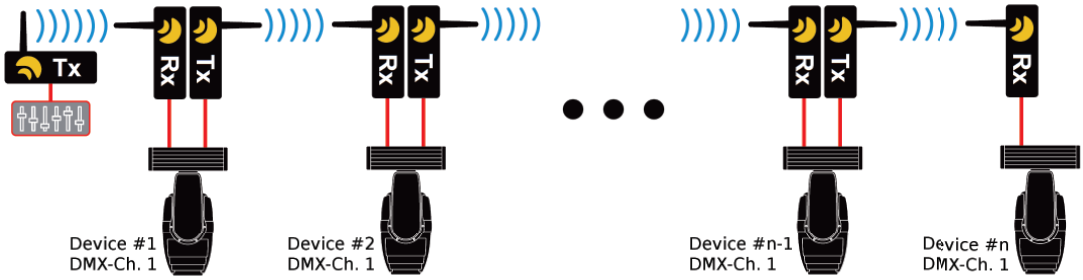

Figure 2.8: A case where W-DMX might generate high delays. The number of moving heads is not explicitly limited by DMX. Every lamp is set to channel #1. The DMX data must therefore be forwarded up to the last device.

2.2.7 Summary

W-DMX is a proprietary network protocol to transmit DMX values wirelessly. Beside the software special hardware is needed, which is sold by Wireless Solutions Sweden AB and partners.

The specialized hardware consists of transmitters, receivers and repeaters. Transmitters are usually connected to a light controller which will provide the data in DMX format. The data will be processed and prepared to be sent wirelessly to the attached receivers or repeaters. Receivers are connected to one transmitter or repeater and via a DMX control line to a number of lights. The received data is used to reconstruct the DMX data and once this has been done it is sent to the lights. Repeaters are middle devices and used to increase the operating range of W-DMX. All the time one receiver can be connected to one transmitter/repeater, but one transmitter/repeater can have multiple receivers attached to it. This way a W-DMX network always has a star architecture; In some special cases this can reduce to a line architecture.

The actual sending is done in the 2.4 GHz or 5.8 GHz band, also referred to as ISM-band. The output power is restricted to 100 mW (300 mW[†]). Adaptive Frequency Hopping Spread Spectrum is used. W-DMX supports a 44 Hz refresh rate, which is the same as in DMX.

[†]"Only in FCC output power mode." [14]

The delay in a W-DMX network is not given explicitly anywhere, but some small calculations shows that the actual delay will be dependant on the number of wireless links and can potentially sum up to a visible amount (assuming a worst-case implementation).

Concluding: W-DMX comes close to what this project wants to cover, but it still lacks some important features that might be needed to be future proof. This includes a better handling of delays, a complete removal of control lines and an address space that is not as restricted as DMX.

2.3 Art-Net

Art-Net is a UDP-based network protocol for controlling stage devices. In its current version it is open, meaning that all details of the protocol is published. Art-Net had been developed and is maintained by Artistic Licence Holdings Limited. The technical description can be downloaded[†], but is also included as version 1.4bf in appendix C. Unlike DMX-512 or W-DMX, Art-Net does not define a physical layer and relies completely on ready to use network hardware.

Art-Net had been developed to "transfer [...] large amounts of DMX512 data over a wide area using standard networking technology" [1]. This will be examined by looking into the network topology of an Art-Net network and its devices. The most important Art-Net packages will presented briefly. Addressing of the Art-Net devices differs significantly from DMX-512 and will be covered in section 2.3.2.

2.3.1 Devices in Art-Net

Art-Net relies on a standard computer network (usually a local area network) and on its designated hardware. It is not restricted to a specific hardware or bandwidth, but it is mentioned that dependant on the hardware the number of addressable universes changes [1, p. 1]. Only in a 1000BaseT network the full range of 2^{15} universes can be used. Obviously, this is because the faster a network can deliver data packets, the more packets can be potentially sent.

Network Devices

Since the basis of Art-Net is a classical computer network, all devices that are commonly used can be included in an Art-Net set-up. This might include

- router
- switch
- hub
- bridge
- wireless access points
- repeater
- ...

It must be said clearly, that those devices can be used, but do not have to be included. The actual situation shows a demand for a specific network.

Controller

The Art-Net standard describes a controller as a device whose main task is to generate control signals. It can be DMX controller with an DMX to Art-Net converter attached or a special lighting console with the capability to output Art-Net data. A controller can

[†] `http://www.artisticlicence.com/WebSiteMaster/User%20Guides/art-net.pdf` [03-2014]

also be PC/laptop with a specific software running on it. Since a computer is usually equipped with some kind of network card it is the preferred machine to use as a controller. The huge amount of universes and therefore channels might be controlled better from a computer than from a console.

Node
A node is any device whose main task is to receive Art-Net control signals. This is often a small device converting Art-Net packages to DMX. However, a lamp that is able to receive Art-Net signal immediately is also called a node in Art-Net.

Media Server
Media Servers are the third kind of device used in the Art-Net standard. It is described as a device that is capable of generating control "signal based on the 'mx' Media Extensions to Art-Net"[1]. However, this device will not play any role in the following paragraphs and shall only included for completeness.

DMX Devices
Beside the computer network n plain DMX networks are needed in Art-Net (see figure 2.9). n is a value between 1 and the maximum number of universes described in [1, p. 1]. A plain DMX network is one as described in section 2.1. No additional restrictions for these devices are introduced here.

2.3.2 Topology

Art-Net is defined as a protocol working on top of UDP. This allows to use standard networking devices and network connections. The advantage of this is that already established network technology can be used and does not need to be constructed individually. Figure 2.9 shows a sample architecture for an Art-Net network. The heart of the set-up is a common computer network. Within this figure it is assumed that the controller is able to output an Art-Net signal. If not, for example a DMX controller is used, another converter is needed to wrap the DMX signal into an Art-Net package. All packages are forwarded trough the network to its destination node, which is an Art-Net to DMX converter, shown as a small black box in the image. Art-Net knows two kinds to forward packages. Peer to Peer and Controller to Peer. This is called a topology in the Art-Net standard and is described in the following paragraph.

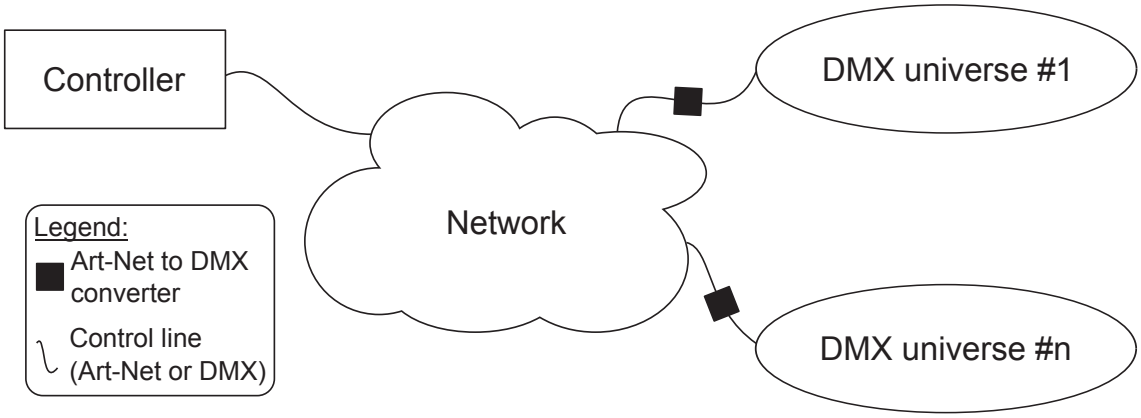

Figure 2.9: Overview of an Art-Net network. Devices in the cloud are casual networking devices such as routers, switches, and others. Devices in a DMX universe are plain DMX devices (W-DMX might also be possible). A converter from Art-Net to DMX is necessary. The controller outputs an Art-Net signal. If a DMX controller is used, another converter must be included. n $\leq$ max. number of universes in Art-Net.

Art-Net defines two topologies for their networks:

1. **Peer to Peer** is called an unmanaged network, where all devices are able to send and receive data. For communication only ArtDmx are used which are always sent to the directed broadcast address 2.255.255.255 (or 10.255.255.255 depending on the network switch[†]). This means that all devices on the same local network will receive the packet. Beside the direct broadcast a limited broadcast is known by Art-Net, which has the address 255.255.255.255. Art-Net Packets shall not be sent to this address [1, p. 2: Limited Broadcast].

2. The second topology described is called **Controller to Peer**. In this set-up one or more controllers are used to manage the network and the respective traffic. Communication occurs only between controllers and Art-Net devices. In this operation mode up to $2^{15} = 32,768$ universes can be managed. However, the number of universes is also dependant on the bandwidth of the network. To use all 32,768 universes a bandwidth of 1000BaseT is needed.

Since UDP is used to sent and receive data, an IP address must be provided to allow the Internet Protocol to work properly. This can be done dynamically by using DHCP[‡] or statically. If used the latter one, two possible modes must be considered:

1. Custom IP address (IP address and subnet mask can be programmed to own rules)

2. Default IP address (IP address chosen according to table 2.2, subnet mask is set to 255.0.0.0)

[†]The network switch is located at the actual Art-Net device and can be set to on or off. It is not explained where the network switch is used for other than changing the first octet of the IP address from 2 to 10. Maybe this is introduced to avoid IP address conflicts.

[‡]DHCP: Dynamic Host Configuration Protocol, assigns network configurations (like an IP address) to a client from a server.

Product Switch Settings	A	B	C	D	Subnet Mask
Custom IP	As Programmed				As Programmed
Network Switch Off	2	x + OEM	y	z	255.0.0.0
Network Swith On	10	x + OEM	y	z	255.0.0.0

Table 2.2: Definition of IP address for Art-Net devices.

To arrange an IP address for an Art-Net device, the MAC address and a special OEM code is used. The MAC address is a 6 byte number in the format u:v:w:x:y:z and is globally unique. The first three bytes (u, v, w) are assigned to the vendor, the last three bytes (x, y, z) form a unique number for every device[1]. OEM is a special 4 byte code provided by Artistic Licence. A list of sample codes can be found in appendix D. The fields C and D are taken directly from the MAC address. Field B is computed as B = x + OEM(high byte) + OEM(low byte) to make it more robust against IP address conflicts.

2.3.3 Art-Net Packets

UDP uses ports to deliver the package to the right application. Art-Net uses port number 6454. However, the port is not officially registered at IANA[†]. Registered or not, potentially every application could send datagrams to this port. To avoid messy data only packets that comply with the specification can be accepted by a node. This includes a common Art-Net header, shown in table 2.3. “Any other packets are ignored”[1].

Offset (bytes)	0	1	2	3
0	‘A’	‘r’	‘t’	‘-’
4	‘N’	‘e’	‘t’	‘\0’
8	*OpCode*		*Protocol Version* (14)	

Table 2.3: Characters in quote signs are ASCII encoded. ‘\0’ is the null character. *OpCode* describes the purpose of the packet numerically. DMX values will be sent in a packet with the code OpDmx (0x5000). A list of OpCodes can be found in appendix C, table 1. Current protocol version is 14.

Art-Net defines 16 packets for discovery, operation and management of the network. Two packets shall be shown in more detail, namely that is ArtPoll (and ArtPollReply) and ArtDmx. The first one is interesting, since it discovers the network and gets information from the devices back and the second one because it is directly related to the 2 chapters covered before. The other packets have their right to exist but do not influence the operation of the Art-Net network other than by producing traffic. However, this is not the focus of this analysis.

ArtPoll / ArtPollReply

When a network set-up is started the first time none of the units in the network know anything about the other devices. To change that a controller sends out an ArtPoll request to the directed broadcast address 2.255.255.255 (subnet mask 255.0.0.0, Art-Net port 6454). The intention of this message to spread out the information about the different

[†]`http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?search=6454` [accessed 14-03-2014]

devices into the network. This will be told the devices with the data field *OpCode* in the Art-Net header (OpCode OpPoll (0x2000) has to be used). Only two more fields are included in this packet. *TalkToMe* which has three flags, indicating if the sender of the message wants to receive diagnostic data, if these data are sent as broadcast or unicast or if only ArtPollReply messages shall be sent in response to an ArtPoll or ArtAddress packet. Lastly, the priority of this message can be set by a user to low (0x10), medium (0x40), high (0x80) or critical (0xE0) in the *Priority* field.

Figure 2.10 shows a sequence chart of the ArtPoll request. It can be seen that it is broadcast and because of this received by the sender itself. Every unit receiving this request has to reply with an ArtPollReply message to the directed broadcast address, even the sender. This allows that after a single request all Art-Net devices have sent their information to the network and no further service discovery is needed.

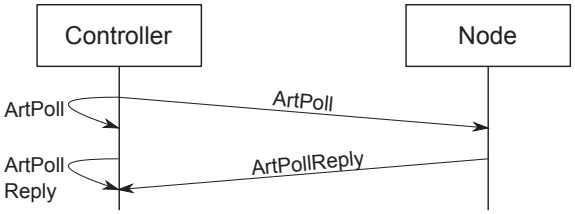

Figure 2.10: Packet flow of an ArtPoll request

The Response to an ArtPoll packet is called ArtPollReply. It is sent to notify the network about important information about this unit. The packet itself has 41 named data fields, many of them are used for general purposes such as status, firmware version or name[†]. Four fields are of major interest: *IP Address[4]*, *Port*, *MAC* and Port-Address, which is not a single field, but split to the three individual fields: *NetSwitch*, *SubSwitch* and *SwIn[4]/SwOut[4]*.

IP Address[4] contains the IP address of the device sending this message, *Port* is the port that it uses for listening to Art-Net packets. The port is fixed to 6454. These information are very important for further execution. This way a command for a specific universe can be sent directly to an Art-Net node without a need for broadcast, which will reduce traffic and increase performance. This operation mode must be used if more than 40 universes shall be hosted in an Art-Net network [1, p. 1]. To address a universe, the scheme in table 2.4 is used. *Net* (bit 14 – 8 from table 2.4) is transmitted in the 7 least significant bits of the *NetSwitch* field in the ArtPollReply packet. The same applies for bit 7 to 4 (named *Sub-Net* in table 2.4), which is transmitted in the *SubSwitch* field. The *Universe* (bit 3 to 0 in table 2.4) is sent in two fields identically: *SwIn[4]* and *SwOut[4]*. Since this field is an array of length 4, up to four different universe addresses can be sent per output-, input connector. This is a smart option, since a single Art-Net node can have more than one DMX output/input. This way all gateways can be addressed by this packet. By looking closer to table 2.4, it can be seen that 15 bits are used to encode a DMX universe, what

[†]It seems not necessary to describe every single field in detail here. For a full list of fields please see appendix C

brings a total of $2^{15} = 32,768$ universes that can be addressed. On a device this address can be set and reset.

Bit 15	Bit 14 – 8	Bit 7 – 4	Bit 3 – 0
0	Net	Sub-Net	Universe
Port Address			

Table 2.4: Definition of Port Address for Art-Net universes. 2^{15} universes can be addressed.

ArtDmx

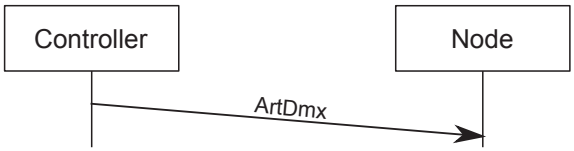

Figure 2.11: Packet flow of ArtDmx

The ArtDmx packet is described in detail by [1, p. 20ff.]. The most important fields are *OpCode*, *Length* and *Data[Length]*. The OpCode for this kind of packet is 0x5000. *Length* describes the number of values transmitted in the *Data* field, which contains the actual DMX values. Since up to 2^{15} universes can be addressed and a node can potentially send data to more than one universe, the port address described in table 2.4 is included in the packet in the *Net* and *SubUni* fields.

Image 2.11 show the data flow of an ArtDmx packet. It is sent from a controller to a node. This can be done by unicast or broadcast, where unicast is generally recommended. It is mandatory if more than 30 universes shall be driven in one network.

To say that an ArtDmx packet is always sent from a controller to an Art-Net node is a little abstraction. It is possible to use a DMX controller which outputs a DMX signal, input it to an Art-Net node which forwards it over a network to the specified node to un-marshal the packet and create a new DMX signal. In this case the communication would be node to node communication. However, the behaviour is the same. One node/controller sends an ArtDmx packet and the other one receives it. No acknowledgement will be sent.

It must be mentioned here, that the ArtDmx packet does not transport any information for error correction, neither does the underlying UDP packet (The packet gets dropped if an error is detected). This might speed up the communication, but it also makes it unreliable. Especially in an environment where wireless paths are included and a lot of traffic occurs this might have an impact on the quality of data. This might be a drawback of Art-Net.

2.3.4 Summary

Art-Net is an application layer protocol that is based on UDP to control stage devices. The promising advantage is that it relies completely on standard network techniques as known from local area networks. The protocol itself describes 16 packets all starting

with a standard Art-Net header. Since Art-Net relies directly on an computer network, it is simple to write computer applications that can control stage lights. Almost every programming language offers direct support for creating and sending UDP packets. This might be a big advantage because it can be abstracted from the actual value to more descriptive methods like a GUI on a computer screen. For example this can be a colour wheel instead of three numeric values.

The communication between the devices is strictly regulated by the standard and an example of ArtPoll and ArtDmx had been given. One observation here is that ArtDmx has no methods included for error correction, neither has the underlying UDP protocol.

Art-Net has the largest range of available addresses for DMX. Even if the DMX standard is not touched, the introduction of 15 bits for describing the address of the universe gives a total of $2^{15} \cdot 512 = 2^{15} \cdot 2^9 = 2^{24} = 16,777,216$ DMX addresses in theory. However, this is only available if the network fulfils a certain bandwidth and unicast is used. But because Art-Net is connected to DMX control lines, it has to take the restrictions of DMX into account.

2.4 Conclusion

This chapter gave an overview of the DMX-512, W-DMX and Art-Net protocol. DMX-512 is the only protocol of the three that had been standardized through an official authority. As a result DMX-512 has a high influence on other protocols. W-DMX and Art-Net are coming from the business sector and follow own goals. While W-DMX had been created to replace one or more wires, Art-Net wanted to get rid of the restrictions of addresses that is given by DMX-512. Both reach their aim, but none of them brings in fundamentally new concepts. Both rely on the principles of DMX-512 and its restrictions though. To increase the number of addresses both introduce a technique to drive separate universes side by side.

W-DMX introduces error correction, which seems to be a very good idea in order to avoid corrupted data. This is an advantage over pure DMX-512. However, once the data packets are corrected they are transmitted as plain DMX packets, which makes W-DMX applicable for a wide range of devices already on the market, but also less safe for data. W-DMX forms a first step towards a completely wireless stage lighting but it is still restricted to limits of DMX-512.

	DMX-512	W-DMX	Art-Net
Topology	line, star	line, star	depends on network
Physical Layer	RS-485	RS-485 + wireless	RS-485 + network
Devices	- Controller - Splitter - Repeater - Lamp	- Controller - Transmitter - Receiver - Repeater - Lamp	- Controller - Lamp - Art-Net to DMX converter (node) - network devices
Addresses	512 per universe, unlimited universes	512 per universe, unlimited universes	512 per universe, max. 2^{15} universes
Data Transmission	1 to 512 channels, position = address	like DMX	like DMX
Resolution	8 bit/channel	8 bit/channel	8 bit/channel
Reliable Data?	no	in wireless part	no
Reliable Transmission	no	wireless: unknown DMX: no	network: no (UDP) DMX: no
Synchronization	within universe[†]	within universe[†]	within universe[†]
max. distance	$\leq 1,200$ m ≤ 32 units	like DMX plus wireless path	like DMX plus network

Table 2.5: Comparison of DMX-512, W-DMX and Art-Net

Art-Net is based on UDP, which allows to use standard network devices such as routers, switches and so on. This is a clever idea since these devices are relatively cheap to buy

[†]Signals for multiple universes can be externally synchronized by sending data to the universe at the same time. After sending it is not possible to synchronize data any further, since communication between universes is not possible.

and maintain them. However, Art-Net has nothing like error correction and has to follow the restrictions from DMX-512.

The most important features are compared in table 2.5. It can be seen, that W-DMX and Art-Net are dependant on DMX-512, but also that both protocols introduce new ideas to extend the existing DMX-512 standard.

3 Problem Formulation

Chapter 2 gave a detailed overview of three existing protocols in the area of stage lighting. This chapter will use the discovered results to address problems which might occur in practical and theoretical applications. The given problems will be listed and prioritized. This list of problems will then be used to formulate the main problem(s) that this master thesis aims to cover. Therefore this chapter is of utmost interest because it will indicate an overall direction of the following parts of this report.

3.1 Address-Space

DMX has a maximum of 512 addresses per universe. A universe is an independent network and no communication to other universes is possible. The same restriction applies for W-DMX. Art-Net, however, includes in its packets a port address, which addresses a universe. This allows for one controller to send signals to multiple universes. Some light control manufacturer developed control panels for DMX that allows for direct connection to a second (third, fourth, ...) universe. This allows for more channels.

A drawback of current systems is that values have an 8-bit resolution. This makes 256 possible values per channel. Once more values are needed, like for moving heads or coloured lights, a second channel needs to be used.

A single network with more addresses would allow for simpler maintainability and better synchronization of lights. By providing a higher resolution of date, for example 16-bit, the number of necessary channels drops and more units can be driven in a network.

3.2 Reliable Communication

The term reliable communication means that a value is definitely be delivered over a network and delivered correctly. None of the mentioned protocols include mechanisms to provide this. Only W-DMX has an error correction in its wireless part. This is good, but once it had been received the signal is forwarded over a DMX line which is again unreliable.

A mechanism that acknowledges transmissions and checks data for correctness would provide a more reliable network protocol. It could than potentially be used for applications that directly affects safety and health, such as pyrotechnics.

3.3 Topology

DMX and W-DMX use a line/star topology. This is because of the physical cable of DMX, which has to be looped through each lamp. Art-Net uses a computer network which is not further described. It can be potentially anything, including the internet.

A star topology with the control panel in the centre, seems to be well suited for most applications. Only in special cases, such as shown in figure 1.3 this topology is not optimal and a mesh network is better suited. Section 4.2 will deal with different topologies and bring up the best one for general purposes.

3.4 Transmission

This is the point where all three protocols are using a different approach. All three will be studied at their physical and application layer. Layers in-between are of less interest.

3.4.1 Physical Layer

DMX uses a physical cable as defined in RS-485. It has four data lines, one ground line and a robust XLR-plug. The cable path starts at the controller, going to the first lamp, to the second and so on until all lamps are connected. Lamps do not have to be in any specific order.

W-DMX uses DMX plus a wireless connection to remove one part of the wire. The cable starts here at a controller and ends in the W-DMX transmitter. On the other side a W-DMX receiver gets the signal and forms a new DMX signal from it which is then forwarded, like DMX, through each individual lamp.

Art-Net replaces like W-DMX one part of the control line. However, this part can be more or less any computer network that is able to send UDP packages. It can be wired, wireless, fibre or anything else. Usually the signal comes from a controller and is inserted into the network. At the end of the network an Art-Net node converts the signal into a DMX signal and sends it to the individual lamps over a DMX control line.

For modern applications with a large number of light devices and possibly multiply universes, a lot of cables are needed. A wireless solution would reduce the cost of transportation and helps providing a better work environment for technicians setting up and down stages.

3.4.2 Application Layer

All three protocols follow the same approach on the application layer. This is collecting all data for a universe, from a light console usually, and sending them in a bulk to the lamps. This way every lamp receive 512 values, many of them are not needed. However, this makes it easy to change an address of a lamp or even to set more than one lamp to the same address. On the other hand the length of a DMX packet as shown in figure 2.3 together with the data-rate of 250 kbit/s results in refresh rate of around 44 Hz. By sending more channels in the packet, the refresh rate would drop.

In order to keep a refresh-rate of 44 Hz, a new application layer must be developed. Especially when the data resolution is going to increase, this bulk approach must be reconsidered.

3.5 Synchronization

Within a DMX network data synchronisation is done physically. Since all lights listen to the same control line, all lights will receive the same data at the same time. The largest delay here is between the first channel, which receives the light value at first and the last channel. This delay is $t_d < 1/44 \text{ s} < 22,73 \text{ ms}$ and therefore small enough to be ignored. Appendix B provides a small script that shows the impact of 23 ms delay. Manufacturers of stage lamps might include a small waiting period, so that the first and last lamp will display at the same time.

As good as the synchronization is within a universe, the bad it is between universes. Since no communication between universes can exist, no synchronization can be performed. It is still possible to send signal to multiple networks immediately, but it cannot be guaranteed that lamps will display light values at the same time. Additional delays could occur (e. g. a W-DMX or Art-Net path in a second universe) that do not affect the universe itself, but the synchronization of the different universes.

Synchronization is today and still will be in the future an important part of a network protocol designed for light applications.

3.6 Conclusion

The problems that are caused by current light control protocols have been shown. This following listing will show these problems point by point:

- Only 512 channels (addresses) available
- More channels are available by introducing another independent network
- No synchronization between networks (universes)
- Data sent over DMX, W-DMX or Art-Net are not reliable, nor can it be ensured that they are delivered
- All data is sent at the same time to all lamps
- Data is sent over a physical cable

This master thesis will work on these problems and will propose a new network protocol to control stage lights. That is mainly

- the increase of possible addresses in a network to allow a bigger amount of lamps,

- and wireless communication between lights to save costs for transportation and connection.

It will be studied if the transmission of data will be still performed, as DMX does, in a bulk, or if an individual addressing might be more suitable. The same applies for the network topology. It must be worked on the question, if a new topology must be introduced, or if existing ones are still suitable for the changing application layer.

4 Requirement Analysis

This section will analyse the problems from chapter 3 in more detail and offers solutions. These solutions will then be used for further development in the following chapters. This chapter is of much interest, because decisions made here will have impact on the rest of the project.

4.1 Address-Space and Data Resolution

In the presented protocols, the addresses are limited to 512 per universe. Simply increasing the number of channels that are transmitted will result in a lower refresh rate. As a reminder: the data-rate of the DMX bus is set to 250 kbit/s and cannot be changed. The size of a DMX packet is a function of the number of channles:

$$\begin{aligned} f_{\text{dmx}}(n) &= \text{BREAK} + \text{MAB} + \text{SC} + n \cdot \text{CD} \\ &= 22 \text{ bit} + 2 \text{ bit} + 11 \text{ bit} + n \cdot 11 \text{ bit} \\ &= 35 \text{ bit} + n \cdot 11 \text{ bit} \end{aligned} \tag{4.1}$$

By sending all 512 channels the size of the packet is $f_{\text{dmx}}(512) = 5667$ bit. The refresh rate ρ can be expressed as a function of the number of channels

$$\rho(n) = \frac{250 \text{ kbit/s}}{f_{\text{dmx}}(n)} = \frac{250 \text{ kbit/s}}{35 \text{ bit} + n \cdot 11 \text{ bit}} \tag{4.2}$$

For 512 channels a refresh rate of $\rho(512) \approx 44.1$ Hz can be calculated. This is the same as mentioned in [4]. A simple boundary value analysis shows that ρ will become smaller and smaller the larger n gets.

$$\lim_{n \to \infty} \rho(n) = 0 \tag{4.3}$$

This shows that it is not feasible to send more channels (and therefore more data) via DMX, since this causes the refresh rate to drop. To be compatible with existing DMX devices, a refresh rate of 44 Hz should be kept.

The solution to the problem of not enough addresses could be solved by addressing individual lamps and not just the features within a lamp. This way a RGB-lamp with six channels needs only one address. However, this is a major change and needs beside clearly defined packages another, faster physical layer. This can be seen by equations 4.2 and

4.3, where f_{dmx} has to be exchanged by another function $\hat{f}$, describing the size of the new packages. However, this new function is still dependant on the number of units in the network (also the type of units, since different units request different packets) and will increase with the number of devices in the network. Therefore the equation 4.3 will yield for the same result.

The advantage of this new approach is that features of a light within a package are not longer restricted to an 8-bit resolution. Features can than have individual numerical data types such as BOOLEAN, INT-8, INT-16 or even higher. This will provide a more suitable data format. Certain features can be turned on and off, just by setting a specific flag and other, more complex parts can retrieve data in an 16 or 32-bit resolution. However, this approach will only work on a physical layer that has a much higher data-rate, since this will be a fixed value for calculating the maximum number of devices for the wanted 44 Hz refresh rate.

4.2 Topology and 802.11

Section 4.1 already showed that another physical layer is needed when more addresses shall be supported in a light control network. Chapter 3 also mentioned that a cable connection must be transported and set-up/teared-down for every event. A wireless connection would solve these problems. On the other hand a topology must be chosen for a certain physical layer.

Since the controlled stage lights have to be connected through a network, it seems a small step to use standard wireless network devices. It is assumed that each lamp has a wireless network card, which will not be the case for most device today. That standard network devices works in principle is shown by Art-Net, who are using this technology already. By going over to a completely wireless network, a topology must be chosen to operate on. IEEE 802.11 defines two operating modes:

1. Infrastructure networks
2. Ad-Hoc

Infrastructure networks rely on a central device that organizes the network and performs routing. This device is known as an access point [2, p. 11]. **Ad-Hoc networks** do not need this central device, since it organises itself. This makes the network more secure (one device less used means one device less that can malfunction), but also more communication is needed for maintaining the network. However, in ad-hoc networks routing usually takes place in layer 3. To improve the performance the IEEE standard 802.11s for mesh networks had been developed. It basically defines another MAC-layer that is capable of routing on layer 2.

4.2.1 Perfomance in 802.11s

For mesh networks using 802.11s a physical layer must be chosen. [15] compares the performance of 802.11b/g and 802.11n in an 802.11s mesh network. The results can be

seen in figure 4.1. Two results can be extracted from that image. First, 802.11n performs better, meaning a higher throughput could be measured compared to 802.11b/g. Second, the throughput drops dramatically with an increasing number of hops. In [15, sec. 5.6] it is therefore recommended to not exceed four hops. This limitation is a major point to reject the idea of a mesh network of stage lights. In a practical application, like shown in figure 1.1, much more than 4 lamps exist close together. In a worst case scenario the 802.11s routing protocol (HWMP[†]) could decide to hop over every light, which would result in a unreliable and unpredictable low throughput.

Even in a set-up where lights are distributed sparsely (like shown in figure 1.3), an 802.11s mesh network is not optimal. By taking the 802.11n physical layer, a distance of 300 m can be covered, in theory. By building a chain of four lights, a maximum distance of 1,200 m could be reached. However, 300 m is quite a far distance for lights. More realistic would be a distance of less than 50 m. This would result in maximum length of less than 200 m, which is far less than DMX can potentially do today (1,200 m) without amplification of the signal. For these reasons a mesh network is not any longer considered.

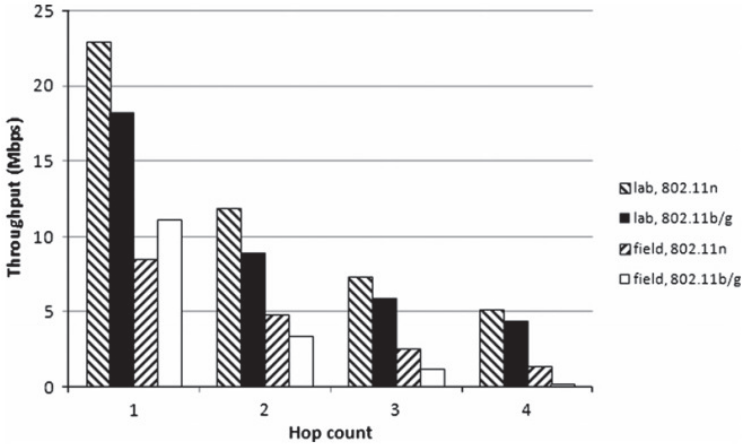

Figure 4.1: Comparison of TCP throughput of 802.11b/g/n physical layers. Copied from [15, fig. 4b]

4.2.2 Performance in 802.11b/g/n

802.11b/g/n networks are popular today and the mostly known ones for wireless local area networking. The performance varies through a large amount of parameters, such as the modulation (FHSS, DSSS, OFDM, ...), the collision avoidance/detection scheme and others. In [3] these parameters are listed and compared for 802.11a/b/g, with the aim to show the theoretical maximum throughput of 802.11 networks. Unfortunately 802.11n is missing in this paper. However, the two most important results are that the actual throughput that can be performed will be less than the net data-rate. Much less. Secondly, the throughput is dependant on the amount of data sent in a package (see figure 4.2). This makes sense, since every new data frame has an overhead and by sending many

[†]HWMP is the satndard routing algorithm used in 802.11s. It is hybrid, so that it can work on static and mobile networks. HWMP stands for Hybrid Wireless Mesh Protocol.

small data packages more overhead than data is transmitted, which causes the throughput to drop.

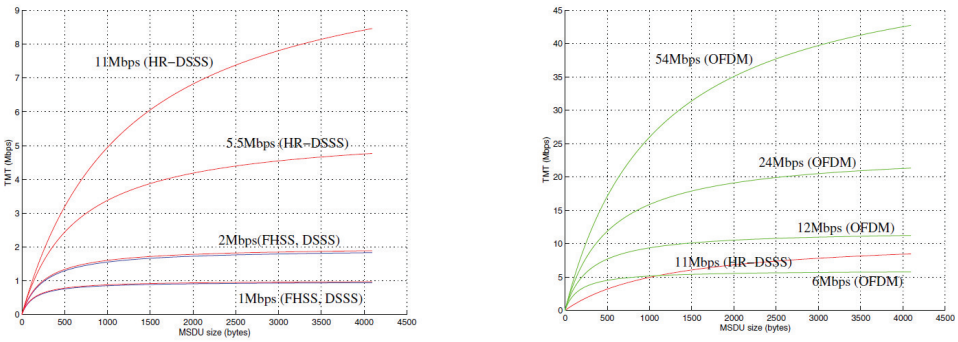

Figure 4.2: Theoretical Maximum Throughput (TMT) on 802.11 networks with FHSS, DsSS, HR-DSSS all with CSMA/CA (left) and HR-DSSS, OFDM with CSMA/CA (right). Copied from [3, fig. 4 and 6]

An infrastructure mode must be chosen to complete the topology. On a first glance ad-hoc mode seems to be the best approach, since no additional hardware is needed. This is true. But an infrastructure network offers some nice features that can make it more generic for general purposes. One of this feature is that one is not restricted to wireless, as it would be in an ad-hoc network. This way a fast backbone network could be established that helps to increase the range, without causing interference. Additionally network devices can be used which are not capable of ad-hoc networking, such as Android smartphones/tablets, or similar.

4.2.3 Summary

Mesh networks do not offer the degree of freedom as it seemed in the beginning. “Normal” 802.11 wireless networks perform in most cases better. The choice between 802.11b, g and n will go to the g standard. It might be sufficiently fast to support a large number of devices and good assumptions can be made upon the results from [3]. It is further assumed that every lamp and other device that shall be controlled, is able to receive wireless control signals, meaning that it has a radio receiver built-in. If this is not the case, an adapter as shown in chapter 7.2 can be used.

4.3 Application Layer

The application layer of DMX is held very simple. Every channel has only one value in the range of $[0, 255]$. By changing the addressing from feature wise to unit wise, the application layer has to be changed accordingly.

Since a standard wireless network will be created between the lamps, UDP will be used to deliver packets. UDP seems to be a good choice, since it delivers fast (compared to TCP) and has only a small overhead. The data inside the datagram will be the actual values for the lamp(s). Two basic ideas must be considered when it comes to sending the data.

1. Send data for each lamp in an individual packet
2. Send data for multiple lamps in one packet

The first approach seems to be more sophisticated and draws a clear line between itself and DMX. The second one is closer related to DMX. With the results from [3] both approaches can be compared. In [3, eq. 5] is shown that the maximum throughput depends on the amount of bytes transmitted per packet (here equation 4.4). For the formula it is chosen $a = 0.14815, b = 159.94$ (OFDM, CSMA/CA @ 54 Mbps).

$$TMT(x) = \frac{8x}{ax + b} \cdot 10^6 \ bps \tag{4.4}$$

The following calculations want to show how many dimmers can be operated at a data rate of 54 Mbps. Equation 4.5 will show the size of a single packet. This includes the overhead of the lower layers up to MAC layer. It is assumed that the value for the dimmer is 1 byte.

$$\begin{aligned} x &= \text{IP Header} + \text{UDP Header} + \text{Light data} \\ &= 20 \text{ byte} + 8 \text{ byte} + 1 \text{ byte} = 29 \text{ byte} \end{aligned} \tag{4.5}$$

With this as a starting point, the theoretical maximum throughput can be calculated next.

$$\begin{aligned} TMT(29) &= \frac{8 \cdot 29}{0.14815 \cdot 29 + 159.94} \cdot 10^6 \ bps \\ &\approx 1.4126 \ Mbps \end{aligned} \tag{4.6}$$

Since a refresh rate of 44 Hz should be kept, the throughput per 1/44 second is of more interest. The time of 1/44 second, which is approximately 23 ms, will be called in interval and represented with the unit i in the following equations.

$$\frac{1.4126 \ Mbps}{44} = 32104.55 \ bpi \tag{4.7}$$

From this value the maximum amount of packages that can be send will be computed in equation 4.8. This is done by dividing by the size of a single packet.

$$\frac{32104.55 \ bpi}{29 \ b/Packet} \approx 1107 \ Packets/interval \tag{4.8}$$

The result of 1107 Packets maps directly to 1107 lamps that could be driven. Compared to DMX that is 1107 channels or 2.16 universes.

Computing the maximum amount of lamps with equation 4.4 by sending every time 1500 byte, the throughput will change to:

$$\begin{aligned} TMT(1500) &= \frac{8 \cdot 1500}{0.14815 \cdot 1500 + 159.94} \cdot 10^6 \ bps \\ &\approx 31.4 \ Mbps \end{aligned} \tag{4.9}$$

Again the throughput per interval is of major interest:

$$\frac{31.4\ Mbps}{44} = 713.\overline{63}\ kbpi = 713636.3\overline{63}\ bpi \tag{4.10}$$

The maximum amount of packets can be computed similar to equation 4.8. However, since the number of packets is not mapped directly to number of lights, it must be calculated how many lights can be driven with each packet. To distinguish between the light data, a new light header is introduced. It contains the address of the device and is 2 byte for this calculation[†]. Because the nature of

$$\begin{aligned} x &= \text{IP Header} + \text{UDP Header} + n \cdot (\text{Light Header} + \text{Light data}) \\ &= 20\ \text{byte} + 8\ \text{byte} + n \cdot (2\ \text{byte} + 1\ \text{byte}) \leq 1500\ \text{byte}; n \in \mathbb{N} \\ \Rightarrow n &= 490 \end{aligned} \tag{4.11}$$

With the knowledge that a single packet can drive up to 490 dimmers, the total amount of lights that could potentially be driven can be calculated. Equation 4.12 uses the theoretical maximum throughput per 1/44 second divided by 1500 bytes, the size of the packet.

$$\frac{713636.3\overline{63}\ bpi}{1500\ b/Packet} \approx 475\ Packets/interval \tag{4.12}$$

By combining the results from equations 4.11 and 4.12, a maximum amount of 232,750 units can be driven. Compared to DMX this is 454.5 universes. The results shows that the second approach is better. However, a practical problem arises here: Only 2 bytes had been used in the example to address devices. This is a maximum of $2^{16} = 65,536$ addresses. To give every of the 232,750 potential units a unique identifier, the address field must be at least 18 bits long.

Combining the results to each other, it shows clearly that the latter approach gives a much higher throughput and therefore a much larger number of potential units in a network. Since these calculations are for demonstration purposes only, the numbers itself are of small significance. The calculations will be repeated, once the packets are designed. Even if the second approach uses a little bit more space for addressing, the result is clearly better. The application layer of the proposed solution will therefore take the second approach, sending multiple values in a single packet, into account. Packets containing multiple light values will be sent as broadcast. This way all network units will receive the message and pick the information that is intended for them (indicated by the address field).

Synchronization is another important feature that the protocol must consider. Two points are of interest though. First, all lights have to display the new value at the same time. Second, packets must arrive within a certain time. Any Packet that arrives after that time should be refused. The term “weak real time” can be used here. This condition is necessary, because light values in a show are strictly bound to a certain time. If a packet arrives delayed, it is not longer part of the show and must be ignored. However, in some cases it is more important for a packet to arrive, rather than arrive in time. Think of the end of a concert where from one moment to the other all lights are blacked out. If in this

[†]This is only for comparison. The definition of the protocol will redefine most of these values!

case a packet would be refused, the lamp stays switched on. So accepting packets with a 'small' delay is acceptable in some cases.

To achieve the first constrained, all lights must have the same internal time. Synchronizing the time of all lamps is a task of the controller, since this is the central device that can talk to any other unit in the network. When a packet with light data is sent, a display time is included. The time can be potentially any time in the future, but it must be kept in mind that the further to the future it is set, the larger a buffer has to be designed. This is necessary because packets that arrive after the first packet had been received and before it is displayed, must be buffered. On the other hand, a large delay makes the system unreliable for an operator. He assumes that an action happens immediately after a button is pressed and not at some point in the future. The maximum delay should never be greater than 450 ms (see [5]).

Figure 4.3 shows how this synchronization of display will be achieved. The controller sends two packets to the lamps (to all lamps, since it is all broadcast). All lamps will check if the packet contains information for itself and if not discard the packet. If the packet contains data, they will be buffered and displayed once the time reaches the display time. In figure 4.3 Lamp #1 and 2 represents each a bunch of lamps that retrieve their information from the same packet. The important thing here is that the ordering of the packets is not important. Even if the packets arrive in a different order each time, the correct values will be retrieved and displayed. Note that the packets are not forwarded by lamp #1, it is a single packet that will be received by different units.

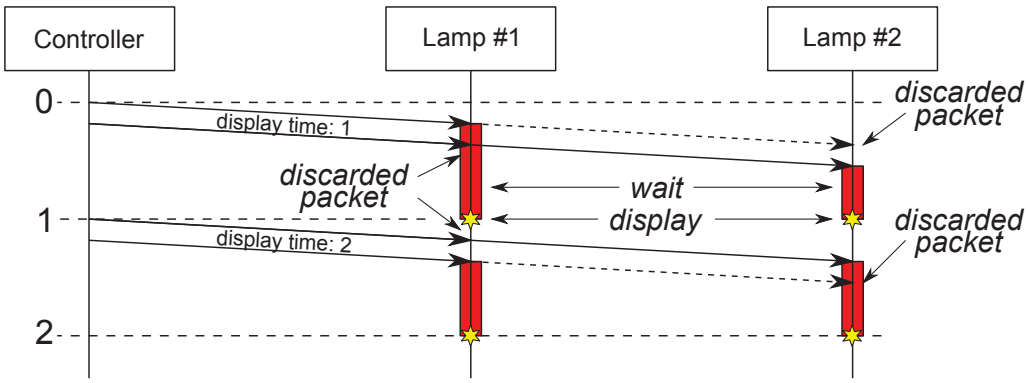

Figure 4.3: Synchronisation of display time. Red: waiting after packet had been received. Star: display

To take the real time constraint into account the controller has an upper limit of messages that can be sent. If the amount of messages stays below this value, it is guaranteed by the lower layers that packets are delivered in time. For important messages the controller can set a flag that indicates this packet as important. A so called importance factor δ can be computed as the difference between the old and new light value (see equation 4.13). This function is an indicator function, returning one if the value is important (major change in the value) and zero if not (only a small change occurred). The function is a proposal and had been introduced for completeness. There might be other functions that does this job

better or faster.

$$\mathbf{1}_A\left(\frac{|v_{k-1} - v_k|}{\max} > \theta\right) \tag{4.13}$$

It is assumed that the parameters v_{k-1} and v_k are from the interval $[0, \max]$. θ is a threshold value and from the interval $[0, 1]$.It can be set accordingly to different requirements. So for example, if $\theta = 0.2$ all changes in the light values that are larger than 20% will be handled as important.

4.4 Conclusion

This chapter dealt with a first analysis of the problems stated in the problem formulation. It investigated the address space in more detail and showed that addressing of lamps is more efficient than addressing features within a device explicitly. Beside from that it had been shown that RS-485 is not capable of transmitting more data than it does today, if the refresh rate of 44 Hz shall be kept. To overcome this and remove annoying cables, a standard wireless network is suitable as a new basis. It will be driven in infrastructure mode. Packets in the network will always be filled up as much as possible (1,500 bytes) to achieve a throughput as high as possible. This allows to install and drive more lamps than DMX could do now. Lastly the network had been bound to real time constraints. To achieve a reliable delivery, a maximum number of packets, and therefore light units, has to be defined. If packets contain an important message they might be accepted by the lamp even when they arrive to late. To show the significance of a packet an importance factor is computed and a flag in the packet is set accordingly.

5 Network Light Control Protocol

5.1 Network Devices

The Network Light Control Protocol is a new protocol based on the ideas of DMX, W-DMX and Art-Net. It is designed to perform better in most scenarios. This is done by using standard network technology and wireless connections between controller and lamps. This section will give an overview of the devices used in a NLCP-network and what hardware requirements they have to fulfil. The requirements are made with a look forward to the application layer and the network topology.

5.1.1 Controller

Like the three other protocols, a controller is needed. This device will take a user input and send it to the lamps. A controller can be a specific hardware console that had been designed just for this task or a computer program that is providing these functions. Anyway, a controller needs a wireless network interface, which connects to an access point which needs to be set up. The access point is the central device in the network and is necessary in infrastructure mode. Manufacturers can include an access point into the device's case, so that controller and access point appear as one physical unit.

Since a packet that is sent from the controller to the lamps includes usually multiple data packets for different receivers (compare to the calculations in section 4.3 ff.), the controller has to transmit its packets to the broadcast address. Every lamp will receive the packet and picks the data that is intended for it. If the amount of devices is to huge to be sent in one package, multiple packets will be sent out. This would result in some messages that do not contain any data for a specific lamp. To overcome this problem multicast can be considered in the future. However, this makes the application more difficult to set up and to maintain for operators. Broadcast will be preferred by now.

5.1.2 Lamps

Beside the controller, lamps are the most important devices. Their task seems to be clear and must not be described further. In all three protocols studied before, no special interest was put on the lamp. Connections had been done by the DMX-interface, which was the only requirement for a lamp to be controlled.

In the proposed protocol, a cable connection is no longer preferred. This is because a wireless network interface is required. The lamps will act as a server, waiting for signals from the controller. The lamps are connected to the access point. At least for the set-up

period a bi-lateral communication is required. In this phase the controller discovers the network and scans for controllable stage units. After this point an uni-lateral communication from the access point to lamp is allowed. This way the range can be increased by transmitting with more power from the access point to the lamps. The lamps do not need this feature of changing transmission power, since no backward communication is designed in operation phase.

A lamp will receive the message from the controller as a broadcast message. It then has to open the message and search for the data that is intended for this device. To make the search faster within the packet, a hash table with addresses and positions will be included in the packet (see section 5.2).

5.1.3 Network Devices

The defined network is wireless and based on the 802.11g standard. The network interfaces from the controller and the lamps have to follow this standard. At least one access point must be included in the network. This is because of the infrastructure mode that had been chosen. However, this allows to create a wired backbone network that can connect to multiple access points. This way the covered range can be increased to a practically unlimited size. By building the backbone network, which will not be covered any further, the transmission speed must be much higher, so that it does not affect the light show. Basically any standard network device can be used here. This is all the devices that also Art-Net uses.

5.2 Packet Definition

The packets defined in this section are forming the application-layer and contain therefore the most important data for the stage devices. Packets had been defined generically for the three most common types of lamps. In a real world more packets might be needed for different types of lamps. Throughout this report only the proposed packets will be used.

5.2.1 Overview

The defined packets are based on UDP. Like Art-Net, nodes are only allowed to process packets that comply with the following defined packets. Since different vendors have a large variety of function built-in the units, it is not an easy task to find a general format that fits generically for every lamp. The following packets are based upon devices from EuroLite[†]. An overview of the designed packets is given in figure 5.2.

The general structure of a multi-packet is shown in figure 5.1. It contains a header, that is unique for this kind of packets and some global information, which are of interest of all included single-packets. Because the different kinds of lamps could be driven with one multi-packet, an address scheme must be introduced to find the data that are of interest for a lamp. This is done by a table that is placed directly after the header. It contains the address of a lamp and its location in the packet. Lastly the single-packets are included, which contain the actual light data.

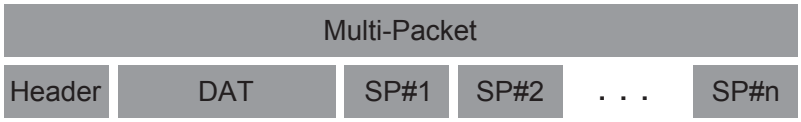

Figure 5.1: Structure of a multi-packet designed to control lights. The total size is always $\leq$ 1472 bytes. DAT: Data Allocation Table; SP#n: single-packet number n.

5.2.2 Packet Header

Every packet has the same header, which includes a preamble, the Version number and the purpose of the packet. It is similar to the one Art-Net uses, but not identical. In figure 5.2 it can be found in the abstract class NLCP-Header.

Offset (bytes)	1	2	3	4
0	'N'	'L'	'C'	'P'
4	*VersionMajor*	*VersionMinor*	*DescCode*	
8	*displayTime*			

Table 5.1: Characters in quote signs are ASCII encoded. *DescCode* describes the purpose of the packet numerically. Current protocol version is 1.0.

DescCode is one of table 5.2 and used to describe the purpose of the packet. Throughout this report only 0x00 will be used, which is the code for single-packets containing light

[†]EuroLite is a vendor of stage lights. `http://www.eurolite.de`

values defined in the next sections. It is not possible to send single-packets for different purposes in a multi-packet. Since every lamp knows how to interpret a packet with a given *DescCode* a further, finer description is not necessary. In all packet descriptions an italic font indicates that this is just a place holder for a numeric value. Normal font is used when exactly this value should be transmitted. Letters in quotation marks shall be transmitted as the numeric value of this letter, ASCII encoded.

The size of the header is 12 bytes.

Value	Description
0x00	Packet containing light values as specified here
0x01	Packet for network discovery
0x11	Reply packet of network discovery
0x02	Packet for time synchronization

Table 5.2: Values for the field *DescCode*

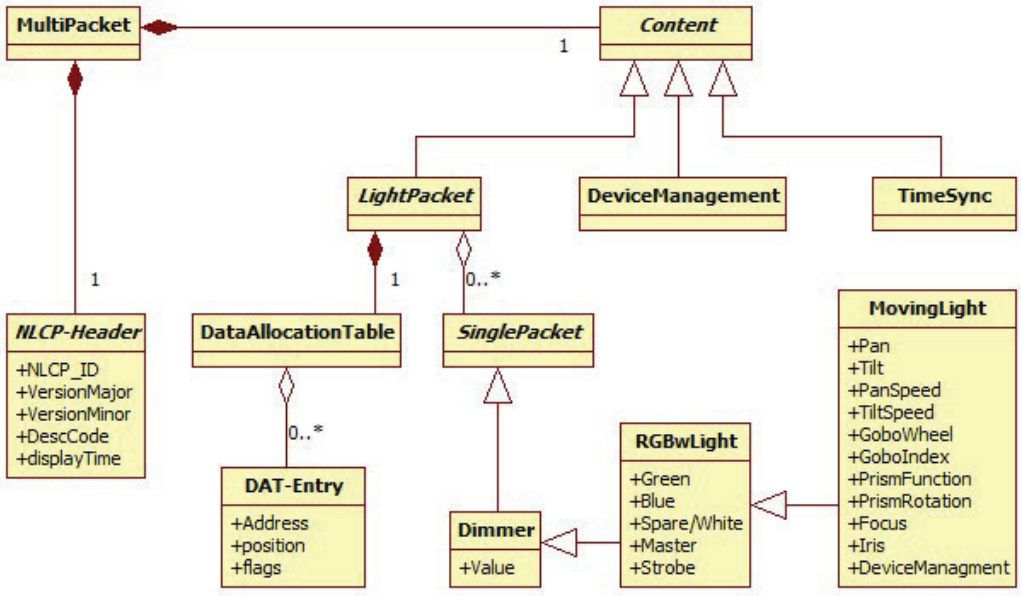

Figure 5.2: Definition of packets as class-diagram. Details can be found in the related sections. Device Management is not defined, yet. Thew field *Value* of the Dimmer class is the Red value for derived packets

5.2.3 Device Allocation Table

Since multiple single-packets are included in one multi-packet, a way to address the individual packets must be found. Inspired by the *File Allocation Table, FAT*, an table will be included directly after the header. The structure is shown in table 5.3. This table provides a fast way to locate the interested single-packet within the multi-packet.

The *flags*-field is used to store some additional information for the device, which are not

Offset (bytes)	1	2	3	4	5	6
0	*tableSize*		*modulo*		*spare*	
6	*Address of Lamp #1*			*absolute Position in bytes*		*flags*
12	*Address of Lamp #2*			*absolute Position in bytes*		*flags*
...	...			...		...
$6 \cdot n$	*Address of Lamp #n*			*absolute Position in bytes*		*flags*

Table 5.3: Characters in quote signs are ASCII encoded. '\0' is the null character. *DescCode* describes the purpose of the packet numerically. Current protocol version is 1.0.

directly related to the light values. To position them in the DAT allows for a semantically clearer definition of the single-packets.

Bit	Description
7 – 4	not set
3	important packet yes/no
2	Light values are interpreted as 0: 8-bit/1: 16-bit resolution
1	Interpret *displayTime* as timestamp or frame number
0	Synchronize lights on/off

Table 5.4: Specification of the bits in the *Flags*-field

If bit 2 in table 5.4 is set, the light values are transmitted as 16-bit. This is the normal case. To make the system downward compatible to DMX, the flag can be set to zero, what indicates an 8-bit resolution. In this case only the LSB[†] are set and the MSB[‡] are all zero and not used. For a converter it is no difference if it should use the first or second byte, but for a human it might be more intuitive to set the higher byte to zero.

Bit 3 can be set by the controller (sender of the packet) to show that this packet is of high importance. This can be the case if the value changes dramatically or this is the last packet for a longer time. In both cases the packet will be accepted even if it arrives out of time and no newer packet had been already received.

To store and retrieve data in the multi-packet, an entry in the device allocation table must be generated. To do that a hash function is introduced which maps the address of the device to a field in the table. A field is shown as a row (containing the address, the position and flags) in table 5.3 and as class DAT-Entry in figure 5.2. A row is addressed by a multiple of 6, starting at zero. A good hash function should fulfil the following criteria [11]:

- small probability of collisions
- chaos, similar elements should result in very different hash values
- surjective
- efficient, fast computation with small memory footprint

[†]LSB: Least significant Bit
[‡]MSB: Most significant Bit

Based on this criteria, a suitable hash function would be 5.1.

$$h(x) = x \bmod m \tag{5.1}$$

To function in the best way, m should be prime and equal to the number of entries in the hash table. Since only in special cases both requirements can be fulfilled at the same time, m should be chosen larger or equal than the maximum amount of entries in the table. If a value is computed that is larger than the maximum number of entries, it should be treated as collision.

Collisions can be easily and reliably handled by linear probing. So if the field is already used, take the next one that is free. This way all elements will be stored in the table. Usually the element can be found directly by calculating $h(x)$. In some cases it becomes necessary to search for the right entry. This way the complexity for storing and receiving is $\mathcal{O}(n)$, but if no collisions occur, the best case is $\Omega(1)$, constant time. In practice the complexity will be in-between the two boundaries, but often tending to the lower one.

Yielding forward to the definition of single-packets, the smallest one will be the one for a Dimmer. It contains only a single value and its size is 2 bytes. Since it is known that the multi-packet will be filled up to 1,472 bytes, an upper limit for m can be determined. The size of the Packet Header is 12 bytes, which is counted only once. The same applies for the first row of the DAT, which is another 6 bytes. The size for a single DAT entry is six bytes and put inside the parenthesis, such as the size for the light value itself (2 bytes).

$$\begin{aligned} n(2+6) + 12 + 6 &\leq 1472 \\ n &\leq \frac{1454}{8} \\ n &\leq 181.5 \end{aligned} \tag{5.2}$$

$n = 181$ is the maximum number of single-packets in a multi-packet. This allows to set an upper limit of $m = 181$. This is suitable, since 181 is by itself a prime number. Taking a number that is prime, allows for a greater chaos in the generated hash value. The controller might compute a suitable prime number before sending the packet. It will be included, so that the receiver knows which number had been used for calculating the hashes. Additionally the size of the table will be included, so that the receiver knows when to handle a a collision due to overflow. In order to save computational time at the sender, a list of prime number might be calculated before a light show begins. Alternatively, prime numbers can be stored in a database and just loaded on start-up. Since the maximum number is known (181), this should be an easy task.

The size of the DAT is $6(n + 1)$ bytes, where n is the number of entries in the table.

5.2.4 Dimmer

Dimmers are the most basic device and require only a single value. The packet is defined as shown in table 5.5. The field *Value* contains a 16-bit unsigned integer value. The 16-bit resolution allows for a much finer control of the light as DMX-512 does. However, a flag in the devcie allocation table can be set that allows to send 8-bit encoded values instead.

Total size of this single-packet is 2 bytes.

Offset (bytes)	1	2
0	*Value*	

Table 5.5: Dimmer Packet with a single field.

5.2.5 RGB(W)-Lights

A device commonly used in stage lighting is a RGB-light. The 'W' stands for white, which could be displayed by RGB-values alone, but sometimes a white LED is used instead. This allows for a much clearer white light. RGB(W)-lights have the ability to change its colour, but cannot move or do anything else than output light. Table 5.6 shows a packet for this kind of lamp. As In every packet, not supported fields shall be transmitted as zero.

Offset (bytes)	1	2	3	4
0	*Red*		*Green*	
4	*Blue*		*Spare/White*	
8	*Master*		*Strobe*	

Table 5.6: Packet for a RGB(W)-light

Total size of this single-packet is 12 bytes.

5.2.6 Moving Lights

Moving lights are commonly used and had already been discussed in the Introduction and Motivation chapter. Scanners and Moving Heads are the most commonly known ones and will be covered by this packet[†]. Since from a protocol point of view they have a lot of common features, only one packet had been designed to support both kind of devices. Details can be seen in table 5.7 and figure 5.2 (class MovingLight).

Offset (bytes)	1	2	3	4
0	*Red/ColourWheel**		*Green*	
4	*Blue*		*Spare/White*	
8	*Master*		*Strobe*	
12	*Pan*		*Tilt*	
16	*PanSpeed*	*TiltSpeed*	*GoboWheel***	*GoboIndexing*
20	*PrismFunction*		*PrismRotation*	
24	*Focus*		*Iris*	
28	*DeviceManagement*			

Table 5.7: Packet for a Scanner or Moving Head.

The fields *Red*, *Green*, *Blue* and *Spare/White* are used in a dual way. This is because some scanners use only a single light source together with a colour wheel to display a certain colour. In this case only the *Red* value is used and the other three values are ignored. *Red* indicates the position of the colour wheel. However, if the light has a multicolour

[†]The packet is based on the features of scanner 'Futurelight DSC-60 LED-Scan' and moving head 'Futurelight DMH-100 RGBW LED'

light source, all fields can be used to set a specific colour. The lamp itself knows how to interpret the fields. However, the operator needs to know how the lamps react before starting a light show.

Pan and *Tilt* fields are used to control the motion of the head/mirror. *PanSpeed* and *TiltSpeed* indicate how fast the light beam moves from one point to another. The following fields are there to modify the light beam. This can be done with a gobo, a prism, the focus and the iris. A gobo is a metal or glasplate that goes between the light source and the opening of the case (The word gobo is short for 'GO Between light and Opening'). It covers a part of the light beam, so that a special shape is displayed instead of a round beam. The shapes are often stars, arrows, triangles and similar. The gobos are mounted on a gobo wheel and to activate a specific one, the field *GoboWheel* will be used. The field is explained in detail in table 5.8 where it can be seen that only the first nibble[†] is used for addressing of a gobo. The last four bits are flags for the gobo wheel.*GoboIndexing* instead can be turned on and off and used to position the gobo correctly. This way an arrow, or somthing else, can be made to point to a certain position.

A prism is a function that duplicates a light beam. Depending on the actual implementation, one or more prism exist, which can rotate forward or backwards. These settings can be made in the field *PrismFunction* and explained in more detail in table 5.9. *Frost* is a flag to show a blur effect.

Focus makes the field to focus and *Iris* can be used to set a diameter of the beam. Many devices might not support all functions mentioned in this packet. Those values should than be set to zero.

Bit	Description
7	GoboIndex on/off
6	GoboShake on/off
5	Rotate gobo wheel forward/backward
4	Normal gobo change/black-out at change
3	Position of Gobo Wheel
2	
1	
0	

Table 5.8: Description of the bits used in the field GoboWheel in table 5.7. 15 positions can be addressed.

This study of Scanners and Moving Heads showed how difficult it is to describe generic packets which are capable of all functions for specific kind of light device. Since the main purpose of this report is not to define packets, the three presented packets will be used as a basis for the further work. The task of specifying packets is left over for further research.

The total size of this packet is 30 bytes.

[†]A nibble is half of a byte. That is four bits.

Bit	Description
15 . . . 8	Not used
7	Rotate on/off
6	Rotate prism forward/backward
5	Frost mode on/off
4	Prism on/off
3	Prism Address
2	
1	
0	

Table 5.9: Description of the bits used in the field PrismFunction in table 5.7. 15 different prisms can be addressed. Bit 8 – 15 are not used, yet.

5.2.7 OEM-Packets

Manufactures of light devices might want to design own packets that are optimized for a certain light series. This can be done by derive packets from the SinglePacket class in figure 5.2. All vendors shall use fields with 16 bit resolution, as long as there are no technical reasons to avoid this.

5.2.8 Network Discovery

Packets for network discovery do not need to carry any other data, than the information that this message is for network discovery. This can be encoded in a single bit and is done by setting the *descCode* to the value 0x01. No additional DAT or any other data is transmitted. The *displayTime* field is not used can carry any data; all zero is preferred. When a lamp receives a network discovery message, it shall reply with a Network Discovery Reply, described next.

The size is therefore the same as for the header, 12 bytes. Is is the only packet that will not be transmitted as a 1472 byte packet. Since this and the Network Discovery Reply are sent before the light show is started, no real-time constraints are set here.

5.2.9 Network Discovery Reply

The Reply to a network discovery is a very small packet, comparable to a dimmer. Basically only the kind of the device is transmitted. Table 5.10 shows it in detail. The IP-address, which is used as address can be collected from the IP-layer immediately.

Offset (bytes)	1	2
0	*Kind*	

Table 5.10: Network Discovery Reply Packet with a single field.

The kind of the packet is to describe the functionality of the device, so that a controller knows how to structure the data. Table 5.11 shows the details of this field.

Value	Description
0x00	Dimmer
0x01	RGB(W)-Light
0x02	Moving-Light

Table 5.11: Details of the *Kind* field of the Network Discovery Reply packet.

6 Evaluation

The protocol designed in the previous chapter is by itself just a combination of bytes. The real-time requirements that had been discussed have to be proved by other techniques. Two tests have been done to achieve this. First a simulation in the network simulator OMNeT++ had been performed to see how many packets can bee sent until the network is overloaded. Second a field test with the prototype from section 7.2 had been used to see which area can be covered by the system until the radio signal becomes too weak to be received.

6.1 Simulation

The simulation of the network is a crucial part of the evaluation process. It gives an understanding of how the network behaves in different situations without building it physically. Main idea of this simulation is to send more and more data, until the network is overloaded and cannot process any more packets. Finding this point, allows for telling until which point the network operates successfully and when it starts to get unpredictable.

The tool of choice was the OMNeT++[†] network simulator (version 4.4.1) together with the INET framework.

6.1.1 System Design

The simulated system is close to the network described in the previous chapter, but not identical. It consists of a controller, a wireless access point but only one Lamp (see figure 6.1). This is an abstraction to the real world, but it does not change the results. Since all packets are sent as broadcast messages, all lamps would receive every packet. The data that is extracted from the packet (sending and receiving time) will be identical on every device. With this in mind the abstraction to use only a single lamp in the simulated network is a feasible abstraction.

The simulation does not cover all possible communication scenarios, since many of them are only of minor interest. For operating the lights, it is of major interest that the data arrive in time, meaning that an operator can rely on the network. This real-time constraint must be fulfilled at any time. The simulation is intended to show when this constraint will break, so that a maximum number of light that can de driven can be determined.

[†] `http://omnetpp.org/`

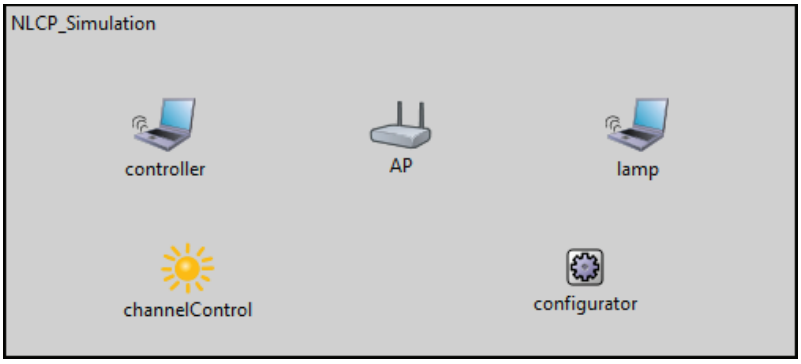

Figure 6.1: Simulated network

The **controller** sends packets, via the access point to the lamp. The behaviour of the controller is similar to an actual controller. Every 23 ms a number of packets are sent. The number of packets is increasing with every run starting with a single packet. The simulation is executed as long as the network can process the packets. Every packet has a size of 1,500 bytes, as they would have in a real application. Figure 6.2 shows the behaviour of the simulation. From the application layer of the controller n packets are sent ($n \geq 1$). The packets are created at the same time and pushed to the MAC-layer, which is queueing the packets. One packet at a time is then sent to the lamp. If the number of packets (n) gets too large, the MAC-layer is not able to send all packets before new packet arrive from the application layer. This will cause packets to arrive too late at the lamp (indicated by the red arrow). Another problem is the limited size of the queue. Packets might get discarded when the queue is full, which will cause packets to be dropped. To find this point, the simulation is executed several times with an increasing number of packets (n) per 23 ms interval.

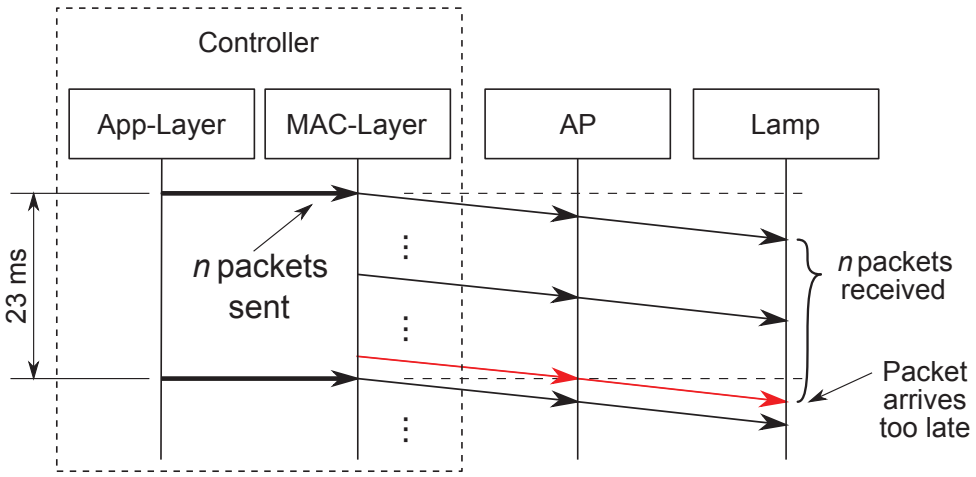

Figure 6.2: Behaviour of the simulated network

The data-rate of the wireless network might have an effect of the number of packets that can be delivered. To take this into account, the simulation had been run several times with

changing data-rates. The data-rates selected to simulate are 6 Mbps, 18 Mbps, 36 Mbps and 54 Mbps to get an understanding of how the network performs when changing this parameter. The expectation is that if the data-rate increases, the number of packets in the network will increase, too.

The **lamp** is the second important part of the simulation. It acts as a sink that retrieves all packets sent by the controller. After receiving the data packet it will be recorded when it had been created and when it had been received. This way it can be analysed if the packet had been arrived in time (if $t_{received} \leq t_{created} + 23$) or too late. As shown in figure 6.2, the creation time ($t_{created}$) is the same for all packets in an interval.

The **access point** is a necessary device in an infrastructure network. Its only task is to receive and forward messages from the controller to the lamp. No recording is done here, since it is not of any interest by now to see how many packets are received/sent by an access point. However, this could be of interest in future improvements and can easily be implemented if desired.

channelControl is used to control the channel. Settings like the carrier frequency (2.4GHz), number of channels (1) and the propagation model (Free Space Model) can be set here. The default values had been usually used.

configurator is a module that is used to make global settings to the network. In this simulation it just be used to set the subnet-mask of all devices to 255.255.255.0 and the address of the network to 10.0.1.0.

The INET framework does provide easy access to the UDP protocol and offers a lot of examples. The controller is a modified UDPBasicApp module, the lamp a modified UDPSink module. The modifications in the controller influence how many packets are sent in an interval (the sending procedure had been wrapped in a for-loop, so that multiple packets are sent). For the lamp only the recording had been added. All values are left as default, except the data-rate.

Because INET does not implement broadcasts by itself, an RTS/CTS scheme had been chosen, where the retry limit is set to one, meaning that every packet is only sent once. The acknowledgement frame had to be sent, in order to fulfil the requirements of RTS/CTS scheme. To drop the effect of this frame, the transmission duration had been set to zero. This way the simulation is close to real broadcast.

6.1.2 Expected Results

As mentioned before there are two parameters that might have an impact on the performance of the network. The number of packet sent in a 23 ms interval and the data-rate that is used for transmission.

To determine the maximum number of packets, the theoretical maximum throughput (TMT) as shown in [3] can be calculated and from that how many packet possibly could be sent. Assuming OFDM modulation, RTS/CTS-scheme (a function for broadcast does not exist in this paper) and a data-rate of 54 Mbps, the TMT can be calculated as:

$$TMT(x) = \frac{8x}{0.14815x + 225.94} \times 10^6 bps \tag{6.1}$$

For a given packet size of 1,500 bytes, the $TMT(1,500) = 26.7759\ Mbps$. Hence in NLCP data must be retransmitted every 1/44 s $\approx$ 0.023 s = 1 interval, a maximum throughput of $26.7759/44 = 0.6085\ Mbpi$ can be achieved. This maps directly to a maximum of 405.7 packets per interval.

For a 6 Mbps data-rate, a maximum of 175 packets could be sent.

However, these results are just theoretical, not taking the topology of the network into account. Because of the access point, which is forwarding every single packet, and therefore blocking the channel, any packet must be counted twice (first when it is transmitted to the access point and second when it is forwarded from there to the lamp). Also this approach does not take any other packets into account that might occupy the channel. So this result is a maximum value, but in reality this might be much smaller.

6.1.3 Simulation Results

The whole simulation takes 120 seconds, which is not very much, but taking into account that 44 intervals are generated per second, this number seems to be efficient. Because the network is doing some initial set-up in the beginning, the sending of light data packets starts at $t = 20\ s$. This results in a recording time of 100 seconds, which is 4348 intervals per simulation.

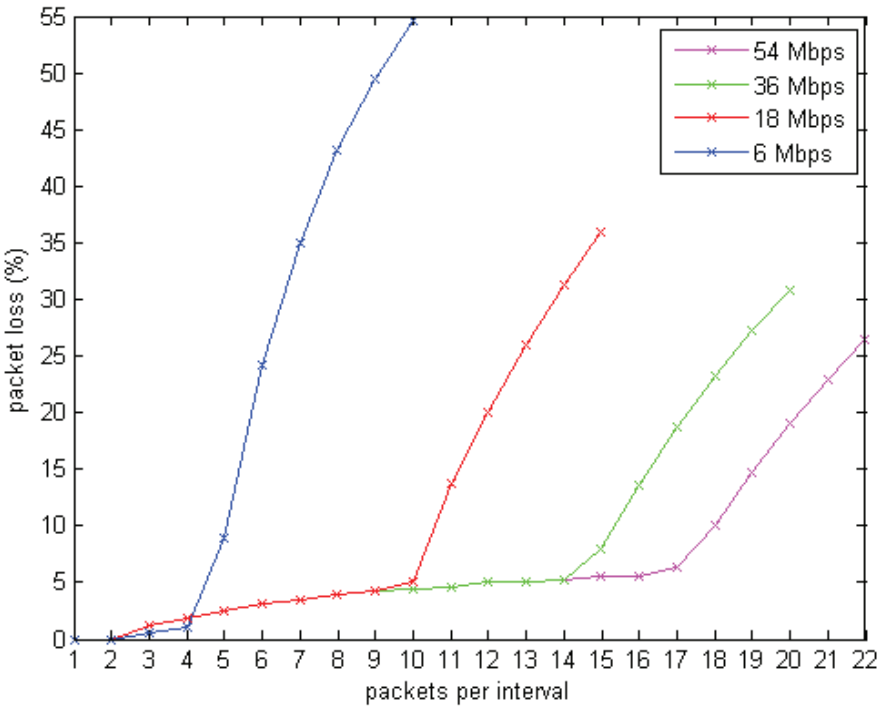

Figure 6.3: Packet loss in the simulation in percent

On a first glance the packets that had been sent can be compared to the packets that

had been received on the other side. This gives a rough estimation on how the network behaves. Figure 6.3 shows the packet loss in percent of the total amount of packets that had been sent. It can be seen that the there is no packet loss if one or two packets are sent in an interval. After that the packet loss increases slowly. Depending on the data-rate, the packet loss increases heavily at one point. This point marks the upper maximum limit. In a real application, one does not want to go beyond this point.

A view to the absolute values in figure 6.4 shows clearly why the packet loss increases so strongly. All simulations behave very similar in the beginning. They show differences in their maximum capacity. Of course the higher the data-rate is, the more packets can be handled. All curves go flat after a while. This is the point where the network is overloaded and any additional packet will be dropped.

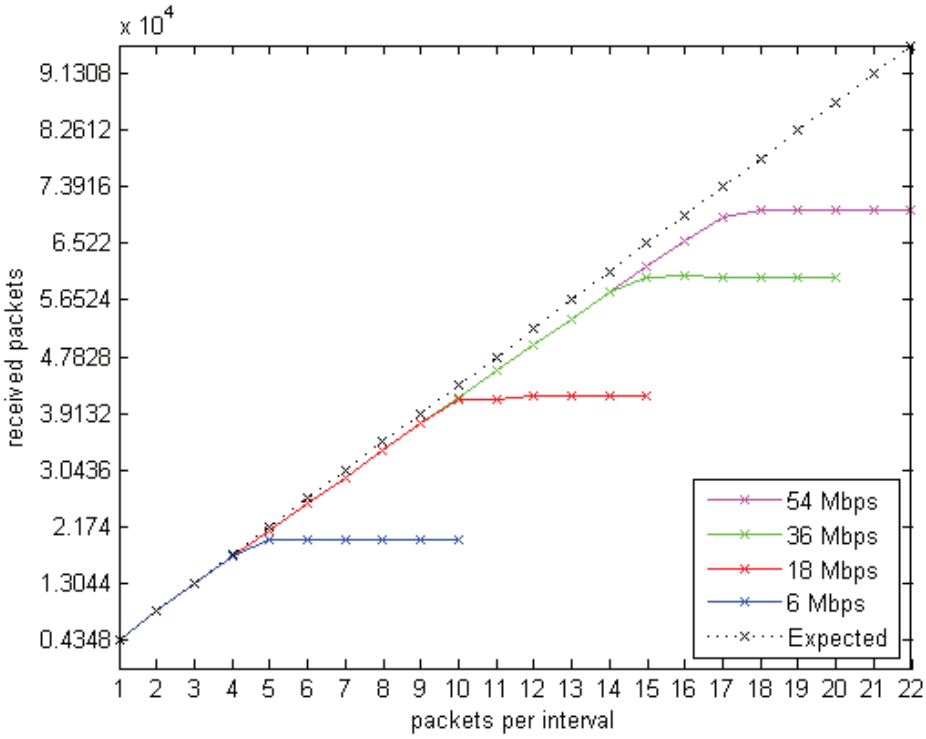

Figure 6.4: Packet loss in the simulation (absolute values)

However, this is only a very rough estimation. Just because a packet arrives does not mean that it arrives in time. Figure 6.5 shows how many packets arrived too late, based on the total amount of packets sent in the according simulation run. Packets that had been lost are counted also as too late. This way the plot provides a correct image of the performance of the network

The interesting thing to see here is that the number of delayed packets increases heavily after a single point. The same had been observed in figure 6.3. The point where the graph changes its direction is identical in both diagrams.

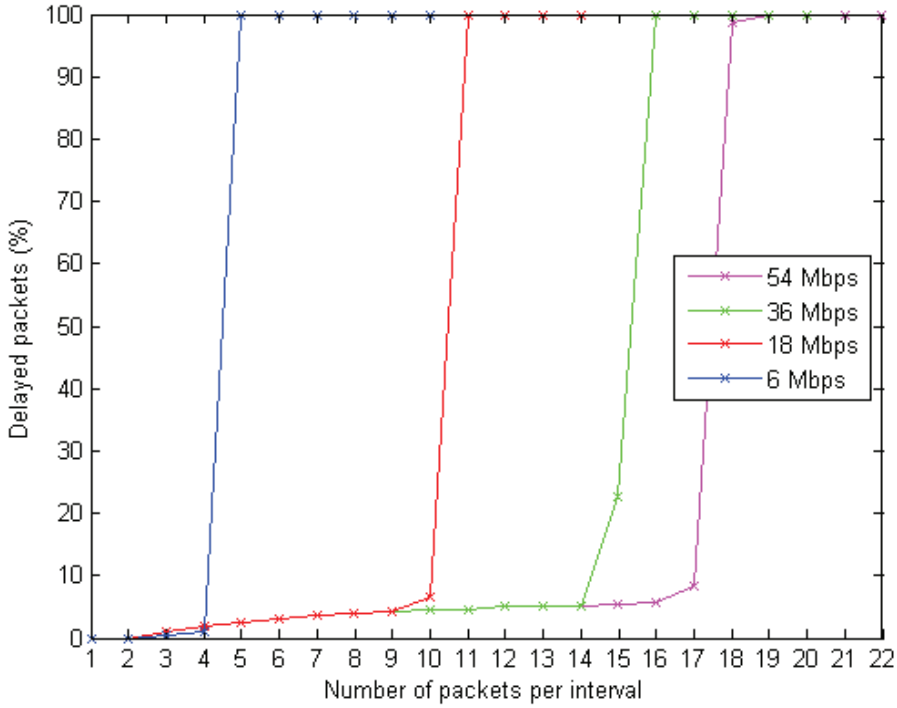

Figure 6.5: Delayed packets (in %)

Of course there must be a relation between the graphs from figure 6.3 and 6.5. Because all the packets that had been dropped are counted as delayed, the number of packets that actually arrived, but delayed, is greater or equal to the number of dropped packets. Therefore, if the number number of dropped packets increases, the number of delayed packets must increase, too. The question is: why is the number of delayed packets going to 100%? Why isn't there at least a small amount of packet that arrive in time? One answer to this might be in the structure of the queue in the MAC-layer. It works after the First In – First Out (FIFO) principle. When more packets arrive, than there can be sent, the queue will overflow after a while. The new packets are dropped and the packets in the queue are already outdated. At the receiver they will be counted as 'too late'.

To overcome this problem, it must be possible to send a signal to the MAC-layer that the queue must be cleared and all data that are still there must be discarded. However, such a signal is not defined and for software developers everything below the transport-layer is a black-box and no access granted.

Up to now, the packets had been only sorted into the categories in-time and too late. This might be legal since, this is the most important measurement. However, it will be interesting to see when they actually arrive. This will provide a better picture of the behaviour of the network. Figures 6.6 and 6.7 display the end-to-end delay of the packets that arrived at the receiver as histograms. Dropped packets are not counted. The plot in the centre shows the simulation where most of the packets still arrived in time, the one

above is the simulation with one packet less per interval, the one below is with one packet more per interval. The solid red line shows the mean delay in this simulation, the dotted line shows the maximum delay of 23 ms.

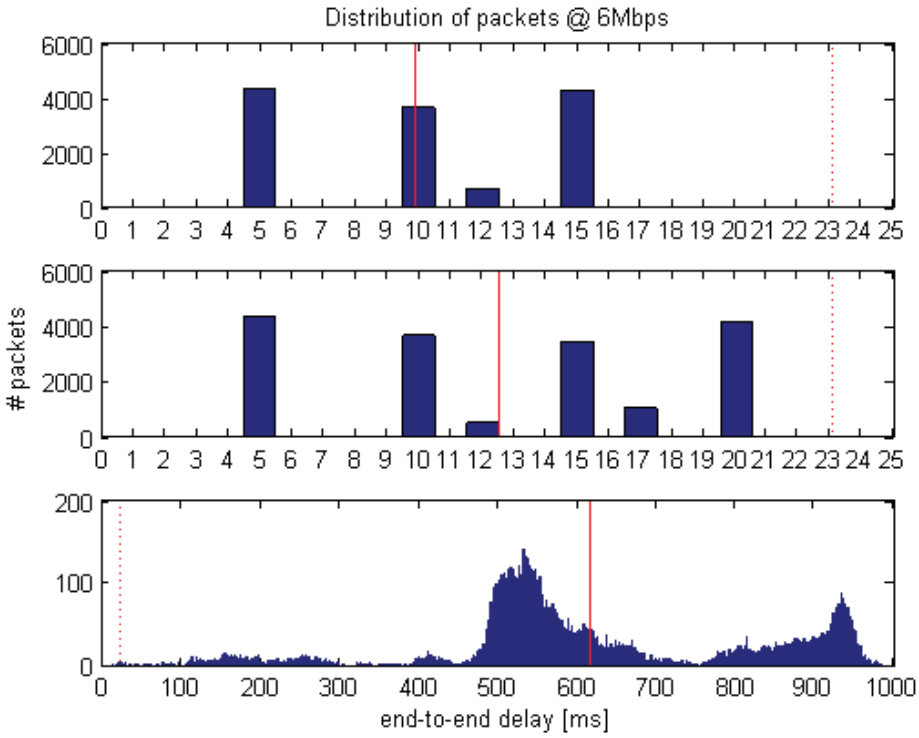

Figure 6.6: Detail view on the distribution of packets at 6 Mbps. Top: 3 packets/interval, Centre: 4 packets/interval, Bottom: 5 packets/interval. Solid red line: mean; dotted red line: 23 ms

Comparing the diagrams of figure 6.6 shows that the one at the top and at the centre are very equal, except that in the centre there is a fourth (large) bar. In this two diagrams the number of the highest bars is the same as the number of packet per interval. The three/four high bars represent each around 4000 packets. An interpretation of this result could be done as followed. All three/four packets are sent at $t = 0$. The MAC-layer buffers them and send them one by one to their destination. This is supported by the minimum delay of 4.60 ms in both simulations.

The diagram at the bottom is the first one where large delays occurred, regarding to figure 6.5. It can be seen that only a very small amount of packets arrive within the 23 ms time frame. Most of the packets have delays of 500 - 600 ms. The delay goes up to 996 ms. An explanation of this might be, that packets gets queued in the sender, before they are actually sent and in the access point, which has to handle them. The size of a queue is 100, so that 100 packets can be stored there. The minimum delay of 4.6 ms and the queue size of 100 leads to maximum waiting time in the queue of 460 ms, which is exactly the point where the histogram starts to show a larger amount of packets. Since packets can get queued twice, the maximum delay of 920 ms can occur. This is the second peak in the

plot.

Figure 6.7, recorded with 54 Mbps, draw a similar picture. In the upper two diagrams, most packets arrive within time. Packets are distributed uniformly here. This changes in the diagram at the bottom. Again, only a minority of the packets arrive within the 23 ms time frame. Most of the packets arrive at $t \approx 120$. The huge difference here is that the packets are normally distributed and not as chaotic as in figure 6.6 – bottom. This could be explained by the fact that the data-rate is much higher in this application and one of the queues does not fill completely. In this case the total delay for a packet is the sum of both queueing times, but cannot exceed a certain limit.

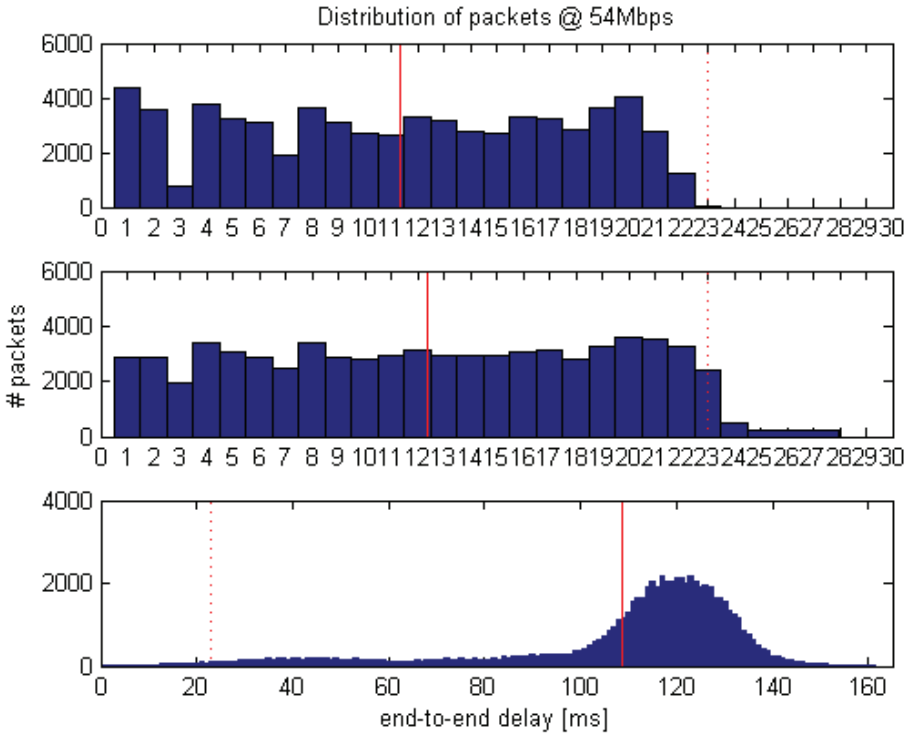

Figure 6.7: Detail view on the distribution of packets at 54 Mbps. Top: 16 packets/interval, Centre: 17 packets/interval, Bottom: 18 packets/interval. Solid red line: Mean; dotted red line: 23 ms

6.1.4 Summary

This section showed the expected results and the one observed by the simulation. These data shall give an upper bound after which the network becomes unpredictable. The expected results had been calculated with the formula of the theoretical maximum throughput. It had been shown that, depending on the data-rate of the network, a maximum of 175 to 405 packets could be sent per 23 ms interval. The number had already been mentioned to be too high and the simulation of the network proved this expectation. In the case that not more than 10 % of the packets are allowed to be too late, a maximum 4 to 17 packets can be sent securely.

What does this mean compared to DMX? One packet sent with this new protocol is 1,500 bytes of size. Minus 28 bytes headers for the IP- and UDP-header means that 1478 bytes are left for light data. The NLCP-header needs another 12 bytes plus 6 bytes overhead for the DAT. Every entry needs another 6 bytes. Table 6.1 shows how many lights can be driven, if all lamps are of the same kind and shows how many DMX channels would be needed to drive the same amount of lamps in a standard DMX-network

	Dimmer	RGB(w)-Light	Moving-Light
Size (byte), Single Packet + DAT-Entry	2 + 6	12 + 6	30 + 6
Max. Devices per Packet (1460 bytes)	182	81	40
DMX-Channels	1	6	33
Max. Devices @ 6 Mbps (4 Pkt/Interval)	728	324	160
DMX-Channels	728	1944	5280
Max. Devices @ 18 Mbps (9 Pkt/Interval)	1638	729	360
DMX-Channels	1638	4374	11880
Max. Devices @ 36 Mbps (14 Pkt/Interval)	2548	1134	560
DMX-Channels	2548	6804	18480
Max. Devices @ 54 Mbps (16 Pkt/Interval)	2912	1296	640
DMX-Channels	2912	7776	21120

Table 6.1: Maximum number of devices that can be driven by NLCP compared to DMX

Taking the lowest value of 728 devices, that could be sent at 6 Mbps, maps to 728 DMX-channels. That is 1.42 universes. Doing the same for the highest value, 640 Moving Lights, that would need 21,120 DMX-channels. That is 41.25 universes. Always keep in mind that delayed packets will occur when driving this many lights. But the number of delayed packets will be much smaller than 10 %. If a higher lower probability of packet loss is necessary, less lamps must be driven.

6.2 Field-Test

It is described by the pathloss models that the strength of radio signal decreases the further away the receiver is placed. The question is how large the distance between the sender and receiver can be without loosing to much information. On the other hand the previous section had shown that the maximum amount of lights in a network is dependant on the data-rate that is used for transmission. In order to answer this question, a field test had been performed. The target was to measure the packet loss at the receiver at different data-rates and distances.

6.2.1 Test Network

The network that had been used for the test is similar to the one from the simulation. One controller (a laptop) is connected to an wireless access point. The receiver, a prototype implementation with an Arduino (see chapter 7.2), had been placed on different positions, increasing the distance to the access point every time.

The **controller/laptop** is a Dell XPS L502X with an Intel i7-2670QM CPU (2 2.2 GHz 64-bit cores), 6 GB RAM and Windows 7 SP1. The program used for the test was a Java application, written with Netbeans. Its task is to send 2000 UDP-packets to the broadcast address. The size of the packet had to be limited to 90 bytes due to limitations on the Arduino[†]. Between the sending of packets a 2 ms delay had been built in, so that the packets have a real chance to be sent through the network. The source code of the program can be found in appendix A.1.

The **receiver** is an Arduino with an Adafruit wireless shield attached to it. It had been programmed to count the packets that it receives. The counter is stored in the EEPROM of the microprocessor, so that it can be read later. The Arduino had been reset after every test run, so that a new, clean connection is available every time. The code written for the Arduino can be found in appendix A.2. Details of the hardware implementation of this device will be covered in section 7.2. Its initial purpose was not to count packet, instead it should act as a converter from NLCP to DMX. But the only difference between the two application scenarios is just the software running on it.

The central device of this network is the **access point**. A CISCO 871W wireless router had been used for this task. It had been configured to act as a root device in the network that allows others to connect to it. DHCP had been activated. The two most important settings are the transmission power, which had been set to 3 dBi (2 mW) and the data-rate, which had been changed regularly to 6 Mbps, 18 Mbps, 36 Mbps or 54 Mbps.

The transmission power had to be set to such a low level so that the measurements could be performed with the given equipment. The limit here was the length of the power cord (45 m) that was available and the length of the footpath. A first test run had been cancelled after the first measurement showed no significant packet loss at the end of the

[†]The RAM of the Arduino UNO is limited. Changes to allow packets > 95 bytes must be made in one of the imported libraries, which might have unpredictable side effects. The effect of this reduced packet size might be a slightly smaller packet drop, since a smaller packet has less bits that can be potentially toggle.

road. It is assumed that this parameter will have an effect only on the distance and the other result will stay the same.

6.2.2 Test Location

Searching for a suitable location is not an easy task. A lot of things must be considered. First, the location must be close to a building, to be able to have access to electricity. Second, the location should be free of disturbing W-LAN and radio signals. Lastly, the location should be similar to the one that might host the system in the future. This will allow for good assumptions about how the network behaviour in reality.

The problem is that those points are often contrary and not all can be fulfilled at the same time. Usually every house has a wireless access point installed nowadays. But the closeness of a building is important to have electricity. On the other hand the test shall be close to reality, but in reality most people on a concert are having smartphones, which are using the same frequencies as NLCP does.

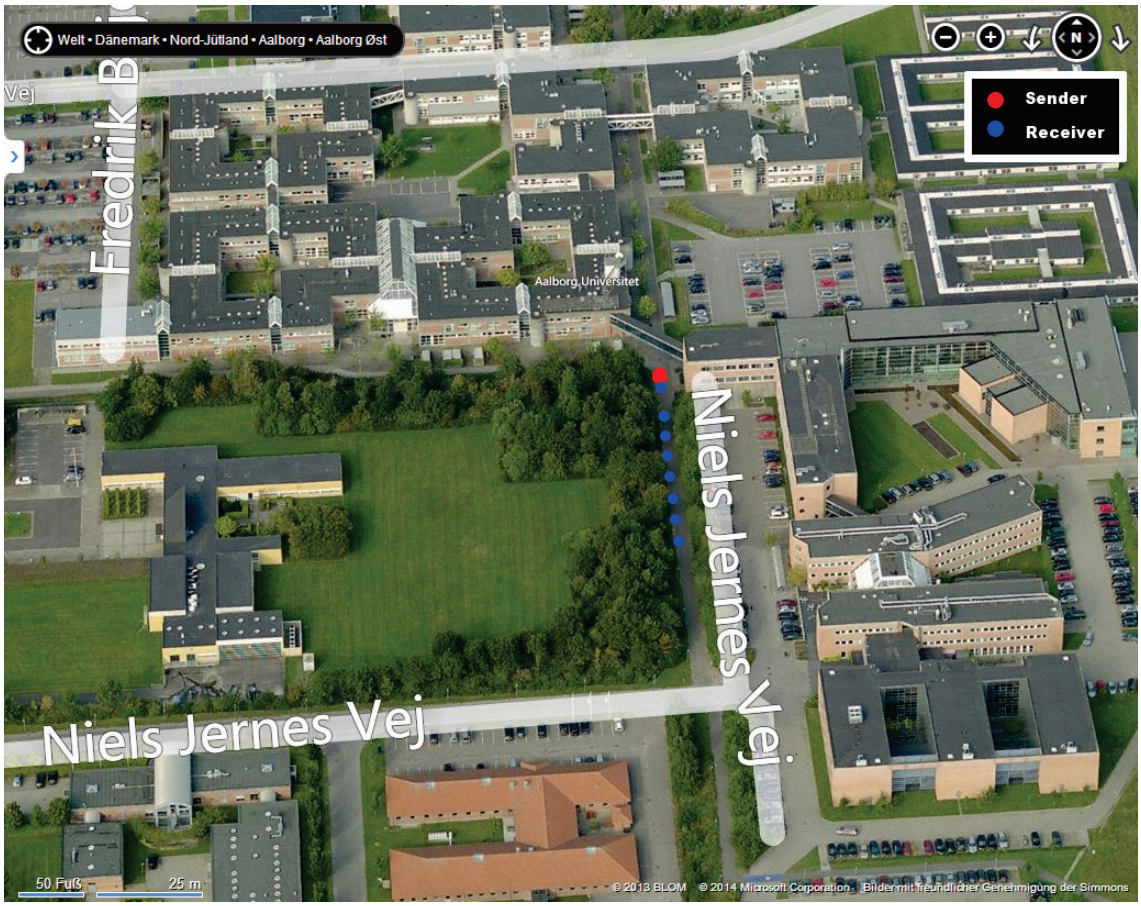

Figure 6.8: Positions of the sender and receiver. The position of the receiver changed during the test. The positions of the blue and red dots are rough estimations.

The decision had been made to stay close to the university, but a bit hidden behind trees, so that the signal from the access point is higher than the other ones. Secondly, the access

point had been configured to use the least congested channel, so that collisions become more unlikely. The exact location can be seen in figure 6.8. The red dot is indicating the position of the laptop and the access point. Both are placed on a table, around 80 cm above ground. The distance between the access point is circa 40 cm. The blue dots on the map approximates the location of the receiving module. The exact distances will be described in the following paragraphs.

6.2.3 Test Execution

Figure 6.9: Images of the field test. Top: close-up of the antenna positions. Bottom: Measurement of packet loss in 1 m distance. Right: Location of the measurements. The receiver had been placed on different spots on the path, always on the right side to allow traffic to pass by.

The test had been performed on Saturday, 17th of May 2014 between 10 am and 4 pm. The measurements had been made at distances of 1, 15, 20, 25, 30, 35, 40 and 45 m from the access point for all data-rates. After a certain point the receiver could not connect to the access point any more, in this case no more measurements had been performed on further away distances and the packet loss had been registered as 100 %. But to get a better picture, few measurements had been done between the last possible distance and here. This way measurements at distances of 22, 27, 32, 42 and 46 m had been made, but only for single data-rates. While the measurements at the main distances had been made all at the same time (one after another), the tests in between those distances had been performed at the end of the field test.

The execution of a test was always the same. At first the receiver had been started. After a while a red LED flashed three times and than turned permanently on. This is the signal that the receiver had been successfully connected to the access point. The program on the laptop had been started than. After the first packet that had been received, the LED

switched off and a countdown of 35 s had been started. For every packet that had been received a counter had been incremented. After the countdown had been reached zero, the counter was written to the EEPROM, so that the results are not lost after the receiver had been reset for the next test run.

This procedure had been performed five times for every data-rate and distance. If the receiver could not connect to the access point within five minutes, the test had been cancelled and a packet loss of 100 % had been counted.

6.2.4 Test Results

During the test a set of data had been collected. Since for every distance/data-rate test 5 results had been produced, the mean value over those values had been calculated. Only for a single measurement, namely the one with 36 Mbps at 1 m distance, curious results had been observed (386, 403, 391, 395, 1996 received packets, out of 2000). On every other data-rate the amount of received packets is close to 100 %. In this case the first four measurements had been treated as measurement error and not used for calculating the mean. The mean values are plotted as dots in figure 6.10.

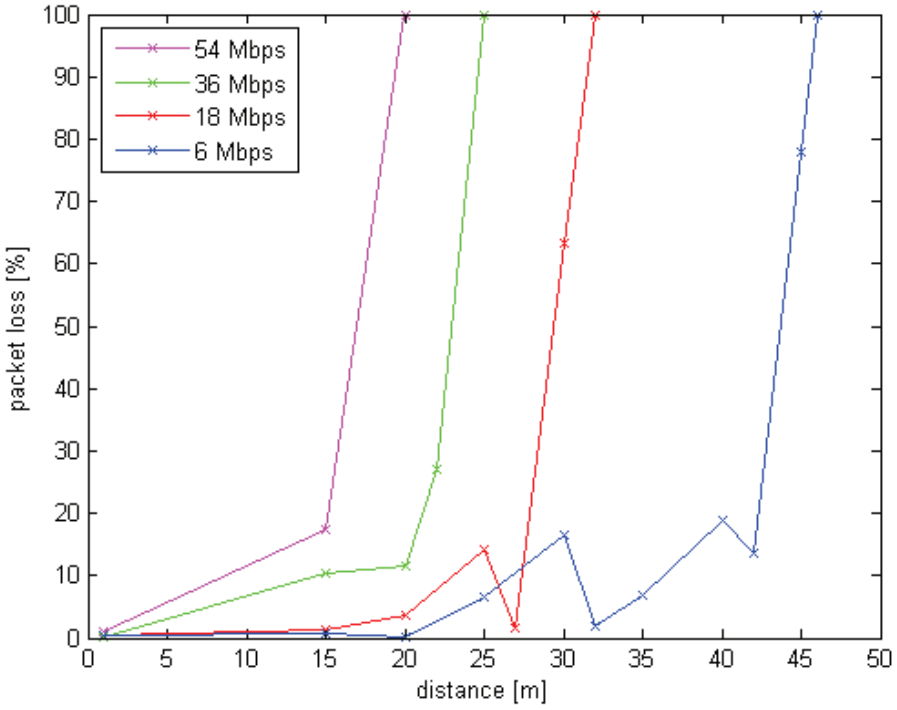

Figure 6.10: Results of the field test. The measurements are marked and connected.

The test results show a relation between the packet loss and the distance between sender and receiver. They also show that the packet loss depends on the data-rate. The larger the distance between sender and receiver, the higher the packet loss will be. Also the slower the sender transmits data, the further it can send while keeping the packet loss constant.

6.2.5 Pathloss

By looking towards the 6 and 18 Mbps graph at figure 6.10, it seems odd that the measured packet loss increases first and drops again immediately after that. This section will try to give an explanation for this behaviour. The first thing that can be realized is that this phenomenon happens not at very close distances, but on larger distances. A first thought is that reflection, as displayed in figure 6.11, could be responsible for the fast increase of packet loss at a certain distance.

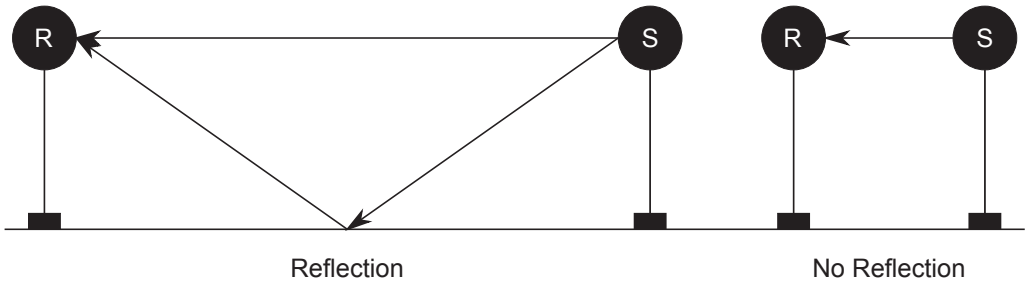

Figure 6.11: If the distance between sender and receiver gets too larger, the signal can reflect on the ground.

To see when reflection happens, a so called Fresnel-Zone can be determined. Fresnel-Zones describe the propagation of radio waves between a sender and receiver. The zones are layered, meaning that there exist many of them. However, higher zones are wrapping lower ones, so that the first Fresnel-Zone is the smallest possible. Zone 1 is also the most important one, since the most energy is transmitted in this zone [9]. Obstacles in this zone have an heavy impact on the transmission. The other effect is that once a Fresnel-Zone reaches a size where it touches an obstacle, the radio waves reflect from it. In this test, the ground is the most important obstacle and might reflect from the point on where the first Fresnel-Zone touches the ground. A sample Fresnel-Zone with the letters used in the formula is given in figure 6.12.

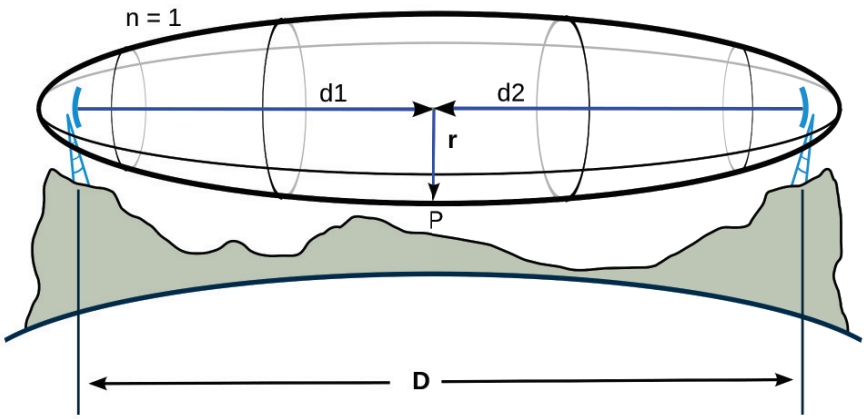

Figure 6.12: Sample Fresnel-Zone between sender and receiver over an hilly area.[9]

The radius r can be calculated with the formula 6.2 for the first Fresnel-Zone. For this test r can be seen as given by the height of the table (0.7 m); λ is the wave length and can be computed by equation 6.3.

$$r = \sqrt{\frac{\lambda \cdot d^2}{2 \cdot d}} \tag{6.2}$$

$$\lambda = \frac{c}{f} \tag{6.3}$$

To find the proper wave length, it is assumed that c (the phase speed) is equal to the speed of light ($c = 299,792,458\ m/s$) and the frequency is 2.448 GHz, which is not defined as any channel, but is exactly the centre value between the first and last channel. With these values the wave length can be calculated as 0.122464 m.

To find the maximum distance between sender and receiver, where the first Fresnel-Zone will not touch the ground, equation 6.2 must be re-ordered to compute $D = 2d$.

$$\begin{aligned}
r &= \sqrt{\frac{\lambda \cdot d^2}{2 \cdot d}} \Downarrow \\
r^2 &= \frac{\lambda \cdot d}{2} \Downarrow \\
\frac{2 \cdot r^2}{\lambda} &= d \Downarrow \\
2d &= \frac{4 \cdot r^2}{\lambda} = D \Downarrow \\
D &= \frac{4 \cdot (0.7\text{ m})^2}{0.122464\text{ m}} \approx 16.0\ m
\end{aligned}$$

In figure 6.10, it can be seen, that the packet loss increases after $\approx$ 15 m. Both results match.

To understand how the signal behaves after this point, the Two-Ray Model can be used. It is a simple model that takes reflection from the ground into account. The principle is already shown in figure 6.11. The received power (P_r) can be calculated by equation 6.4. h_t, h_r are the heights of the transmitter and receiver ($h_t = h_r = 0.7$ m), G_t, G_r are their antenna gains ($G_t = 2.2$ dBi, $G_r = 0.5$ dBi). P_t is the transmission power of the transmitter ($P_t = 3$ dB). An plot of the equation is given by figure 6.13. It can be seen that the received power drops, as the distance becomes larger. This can be directly related to an increasing packet loss. Unfortunately, the digram does not show any deep gaps, as they appear below 10 m, after the 16 m mark. The idea of this diagram was to show that the increasing packet loss at 25 and 30 m could had been explained by such a gap.

$$P_r = \frac{\lambda^2}{(4\pi D)^2} \cdot 4 \cdot \sin^2\left(\frac{2\pi}{\lambda}\frac{h_t h_r}{D}\right) G_t P_t G_r \tag{6.4}$$

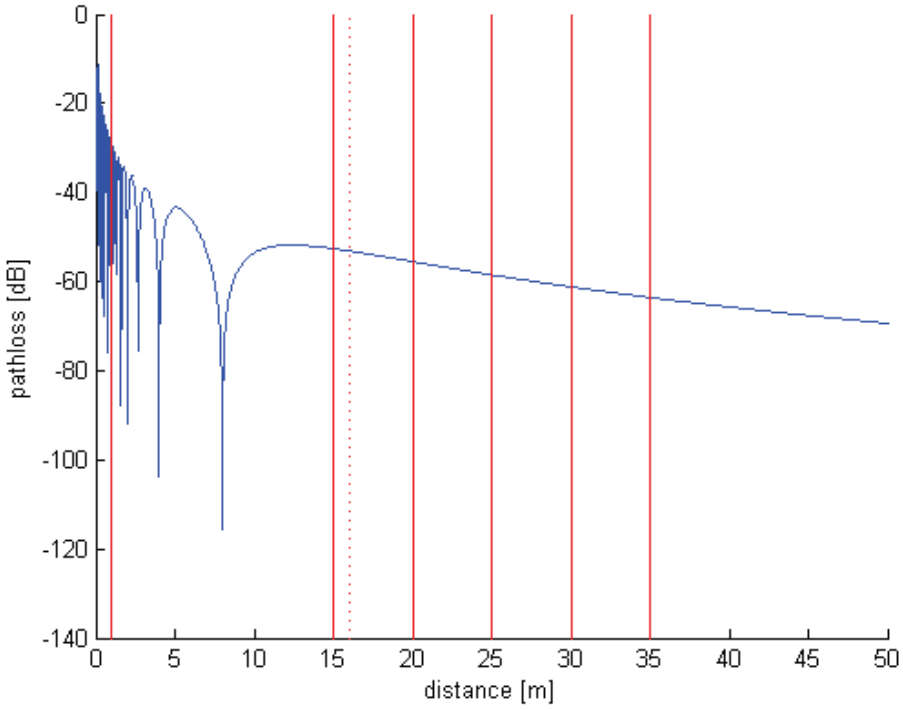

Figure 6.13: Pathloss based on the Two-Ray Model. Solid red lines: Measurement points; dotted red line: 16 m, the point where the first Fresnel-Zone touches the ground.

All test data are presented in appendix A.3.

6.2.6 Summary

The field test gave a good picture of the behaviour of the radio signal over distance and the packet loss that results from it. The further away a receiver is, the more likely is it to lose packets. Also the higher the data-rate is that is used for transmission, the more likely it is to lose packets. Unfortunately the test could not be executed with the same packet size as defined in the protocol. It will be more likely to lose packets the larger they are. However, the results displayed in figure 6.10 show a general behaviour of a wireless network.

With the path loss model and the Fresnel-Zone an idea of the packet drop had been introduced. It had been shown that the packet loss might be not just related to the distance (which had been shown by the physical test), but also from the position of the antenna. By placing the antenna on another, higher, spot the transmission will be affected less by obstacles, including the ground.

Another point to mention is the transmission power. The test had been done with the lowest possible power, which is 3 dBi (2 mW) at the used access point. If this parameter will be changed, the observed distances will change, which is the reason why the distances are shown in figure 6.10, but not taken as a result here. The result is the observed pattern,

saying that the higher the data rate is, the smaller is the range of the application. Also the choice of the antenna will affect the results. A directed antenna might be able to sent much further than the one used in this experiment. However, the pattern stays the same.

6.3 Conclusion

This chapter showed the simulation for the Network Light Control Protocol, trying to find the maximum number of devices that can be driven in an application. The simulation had been made in the OMNeT++ simulator with the INET framework. Unfortunately, INET does not support broadcast, this is why RTS/CTS had been used instead. To come close to broadcast, the retry limit had been set to 1, meaning that every packet can only be sent once and the duration of an acknowledgements had been set to zero, so that this frame does not have an effect to the simulation. The results show that the higher the data-rate of the radio signal is, the more packets can be delivered in time.

To get an understanding of the range that can be potentially be covered a real world field test had been performed. As a receiver a self-made device had been used. It consists of an Arduino board and an Adafruit CC3000 wireless shield. The program was written in C. The task was to count the amount of packets that had been received. This can be directly translated to the number of packet that had been lost. In order to provide a stable network, the packet loss must be kept minimal, which results in a certain distance that cannot be exceeded. Because there are a lot of parameters that can change (position of antenna, kind of antenna, transmission power, interference with other radio signals), the main result is that the higher the data-rate of the radio is, the lower is the distance that can be used.

The different results had been collected and presented in figure 6.14. For the distance, a maximum packet loss of 5 % had been chosen, whereas less than 10 % was the criteria for the maximum amount of delayed packets. All values are rounded.

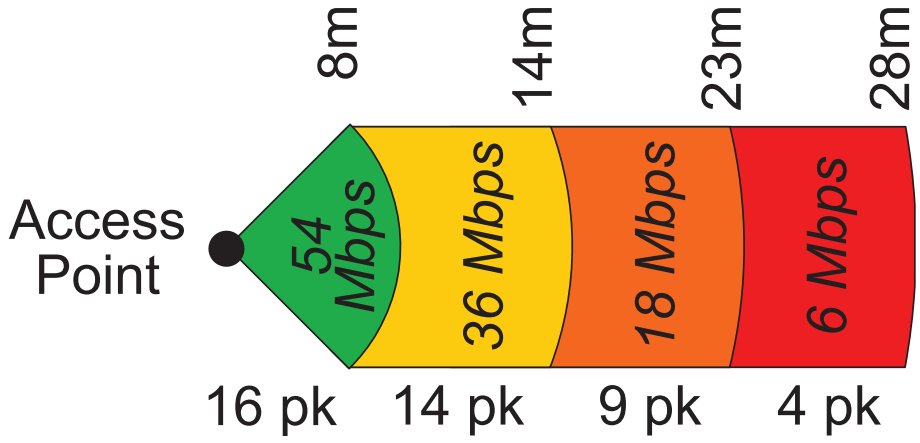

Figure 6.14: Maximum values for distance and number of packets, for a given data rate

7 Implementation

The NLCP protocol was designed to drive lamps and overcome current problems. Simulation can produce data that belongs to a certain scenario, but it does not cover anything that is related to usability or any practical restrictions. To see how the protocol performs in a real application, a controller had been programmed and a converter from NLCP to DMX had been built. This section will show these two parts and show what had been learned by building them.

7.1 Controller

The task of a controller is to provide a user interface, that allows a user to 'insert' data and to send these data in a specified interval to the lamps. Since this project relies completely on standard networking technologies, it was a small step to decide for programming a controller and running it from a laptop. During implementation, some thoughts came up that shall be described here. One is the choice of the programming language, how to structure the packet and a fundamental change in the user interface, compared to existing solutions.

As a programming language, Java had been chosen. Java offers good support for networking. It is easy to send and receive packets1 and its object oriented programming paradigm allows for easy implementation of the NLCP packets. For the user interface Swing was used, but instead of programming every piece manually, the Netbeans window builder was used. One main window had been made, that offers support for connecting to the lamps. This is done by sending a Network Discovery packet to the broadcast address. The received Network Discovery Reply packets are buffered. After a time out (here 4 seconds had been used. This is enough to send one and receive two packet. See chapter 6.1.3) the buffered packets are evaluated and the controller is built upon these responses. The 4 seconds must be change to a higher value, if a lot of lamps are in a network.

All packets had been implemented regarding the class diagram from figure 5.2. To transform the fields of the packets to bytes, a function `public byte[ ] toByteArray();` was included in every class. This way the class will handle the correct order of its fields. Additionally a method `public int getPacketSize();` was included in every packet. This was necessary because it is not always clear how many space must be allocated for a certain packet. Listing 7.1 shows these two methods implemented for the Dimmer packet. The complete source code can be found on the enclosed CD.

```
@Override
public byte[] toByteArray() {
    ByteBuffer buffer = ByteBuffer.allocate(2);
    buffer.putShort(value);

    return buffer.array();
}

@Override
public int getPacketSize() {
    return 2;
}
```

Listing 7.1: toByteArray() and getPacketSize()-method from the Dimmer packet

Packets are created based on the values set at the graphical user interface. However, the mapping from a fader to a packet is not an easy task. To overcome this problem, the controller had been programmed in a way that it creates fader based on the type of a lamp.

7.1.1 Responsive User Interface

Classic control panels provide a number of faders, one for each channel, see figure 7.1. The controller logic will create the DMX signal based on the position of the fader. This works perfectly good, because every feature of a lamp is mapped to a channel. In this approach the controller has no information about the network at all. A signal is generated, even if there is not a single receiver connected to it. This way a DMX controller can be called a 'dumb' unit.

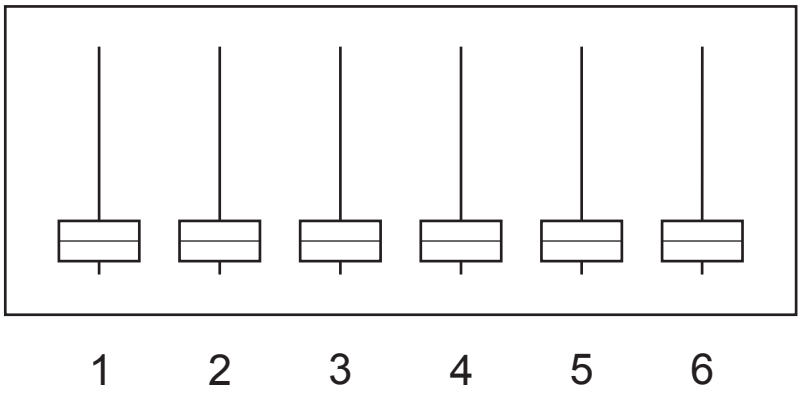

Figure 7.1: A simple DMX control panel, each fader controls exactly one DMX channel.

NLCP follows a different approach. A lamp is addressed by a single packet, not a feature as DMX. DMX addresses channels, that can represent any feature of a lamp. To map a fader to this feature, the controller needs to know which feature is meant in which lamp. This means that a certain amount of knowledge must be given to the controller. This is done by the Network Discovery Response packet sent by the lamp to the controller. Based on this information, the controller can do this mapping. The question is still how

many fader should be created for the control panel? This question is difficult to answer, and maybe an answer is not even necessary. Since the controller is a piece of software, it can change its user interface due to user needs, or the number of lamps in the network. Figure 7.2 shows how the user interface changes when a dimmer and a RGB lamp had been detected in the network.

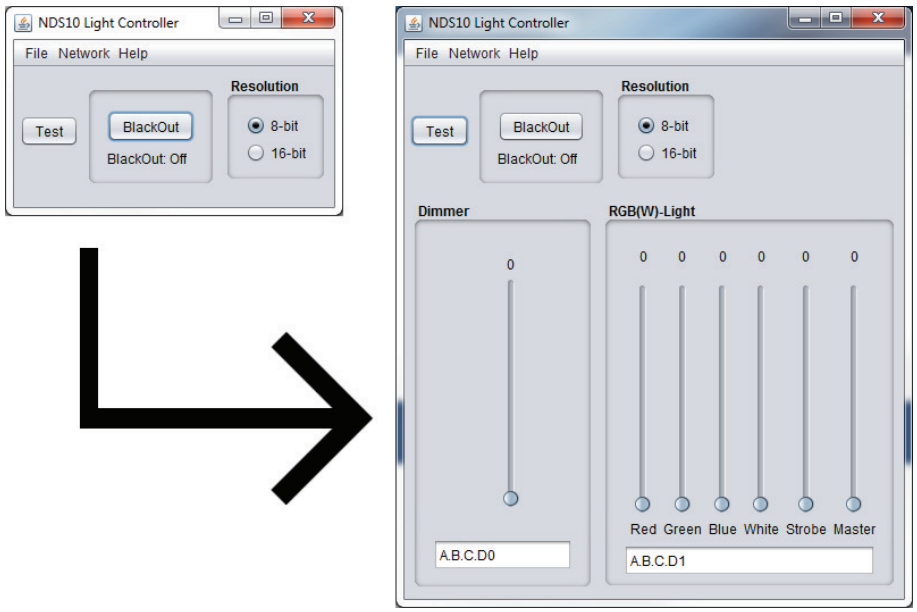

Figure 7.2: Empty controller after start (left) and with two dummy control areas (right)

So, where is the advantage of doing something like this? First, the number of faders will always fit the number of lamps, independent of their kind. Second, once a fader is created, it will always be mapped to the correct feature of the correct light. Third, lamps with a lot of features, like moving lights, have fields of type on/off. A fader is far from optimal for such a feature. Buttons or check boxes are much better for this task (unfortunately, this is not represented in figure 7.2).

7.1.2 Bin Packing Problem

The last section showed how the value of a fader can map to features of a lamp. By digging a bit deeper, the values must be set in a single packet and then be wrapped in a multi packet. There is no problem as long as all single packet can be placed in one multi packet, but if there are more single packets, a strategy must be chosen, so that the number of packets stays as small as possible. This can be expressed as the bin packing problem.

In the original bin packing problem, the task is to fill k bins, each of size $b \in \mathbb{N}$ with n objects, each of size $a_1, 1_2, ...a_n < b$, so that the none of the bins is overfilled. The bins in this scenario are the multi packets. Their number (k) is not defined, but should be as small as possible. The size of a multi packet is given as $b = 1460$ bytes. The single packets are the objects, that have to fit into the bins and are of size (2+6), (12+6) and (30+6) bytes. The number of single packets depends on the actual set-up.

Unfortunately, the bin packing problem is a NP-hard problem and as long as science cannot prove that P = NP, no best solution can be found for that problem that can be executed in polynomial time. However, there are ways to calculate a good solution, that comes close to a best solution. One way is to use a heuristic called »first fit decreasing«. What it does is to order the objects decreasingly (according to their size) and takes the largest object first. It continues until an object is too large to place it in the remaining space of the bin. In this case the next object is chosen. For the leftover objects, a new bin is created and they are placed in there. This procedure will be repeated until all objects are placed in bins. Listing 7.2 shows the pseudo code for this heuristic. By using a heuristic like this, the amount of multi packet can be kept small.

```
1  ALGORITHM ffd(objects) RETURNS bins
2  BEGIN
3      objects.sortDecreasing();
4
5      WHILE NOT objects.isEmpty()
6      DO BEGIN
7          size = objects.size();
8          bin;
9
10         FOR i = 0..size
11         DO BEGIN
12           IF bin.capacity > objects[i].size
13           THEN BEGIN
14               bin.add(objects[i]);
15               objects.remove(i);
16           END
17         END
18
19         bins.add(bin);
20     END
21
22     RETURN bins
23 END
```

Listing 7.2: Algorithm filling bins with the performing First Fit Decreasing Heuristic

7.1.3 Continuous Sending

The application was designed to send the NLCP packets in 23 ms interval. But the question is if this is really necessary. Of course data must be sent if it change, but what if it stays the same? In this case unnecessary data would be sent, which increases the network load and might increase the chance of packet loss.

To implement this feature, packets are created by the user interface and send downwards to the sender object. Once they have been sent, the packet is removed. The NetworkFaçade, which hides the functionality of the sender, will notify the sender when new data are available. The sender will collect the new packets and send them to the network. However, the sender keeps the 23 ms interval, so that changes that occur faster are omitted. If this would not be the case the network could be flooded by one fast changing packet and the refresh rate would not be 44 Hz. Listing 7.3 gives a programming example of this modification. Figure 7.3 shows this principle.

```
1  // imports are not included in this example, as well as getter, setter
       constructor...
2  public class Sender extends Thread {
3
4  private boolean changed = false;
5  private boolean stop = false;
6
7  @Override
8  public void run(){
9
10   while(!stop){
11     send();
12
13     synchronized(this){
14       long t1, t2;
15       t1 = t2 = System.currentTimeMillis(); // get the current timestamp
16
17       while((t2 - t1) < 23 && !stop) { // catches early notifications
18         try {
19           wait(23 - (t2 - t1)); // calculate new wait time
20         } catch(InterruptedException e){}
21         t2 = System.currentTimeMillis();
22       }
23     }
24
25     synchronized(this){
26       while(!changed) { // catches a false notification
27         try {
28           wait(); // Wait until notification
29         }catch(InterruptedException e){}
30       }
31     }
32
33     changed = false;
34   }
35
36 }
37
38 // notify the thread that new data are available
39 public void doNotify(){
40   changed = true;
41   synchronized(this){
42     notifyAll();
43   }
44 }
45
46 // stop the thread
47 public void killThread(){
48   stop = true;
49   doNotify();
50 }
51
52 }
```

Listing 7.3: Programming example that avoids continuous sending

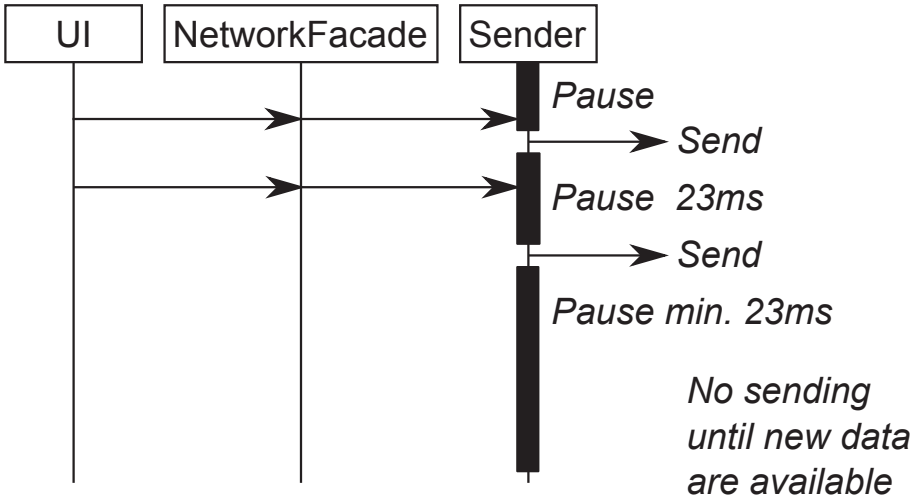

Figure 7.3: Packets are only send if the value has changed. But a minimum 23 ms interval is kept.

7.1.4 Summary

By programming the controller, some knowledge concerning practical problems had been earned. The problem of mapping a fader to a feature had been elaborated and implemented. To reduce the number of packets, the bin packing problem can be used and by going one step further, packets need only to be sent if a values for this lamp change. The bin packing problem had been implemented, since the test network hosts only two lamps, so that the result would be the same every time. The last approach had been implemented in the sender thread.

7.2 NLCP – DMX-Adapter

NLCP makes a very strong assumption, when it demands a wireless connection on every lamp. This request can be fulfilled only by a very small amount of light nowadays. To overcome this problem, an adapter had been built that converts NLCP signals to DMX. This way a common stage light can be driven by NLCP. Figure 7.4 shows the adapter.

The adapter is based on an Arduino Uno R3. It is a very convenient, open source, prototyping platform with an own IDE. The micro controller on the Arduino board is an ATmega 328. It can be programmed in C/C++, the IDE compiles the source code and allows for easy uploading to the board via USB. The structure of an Arduino program differs slightly from usual C-programs. The `main()`-function is not visible for a developer and all the work is done the functions `void setup()` and `void loop()`, where 'setup' is only called once in the beginning and 'loop' is called again and again in an endless loop.

To make the Arduino receive wireless signals, a wireless shield had been placed on top. A shield is an hardware extension to Arduino adding more functionality. The wireless shield is an Adafruit CC3000. Ready to use libraries are provided by Adafruit, allowing for a

fast development. Unfortunately, the needed functionality was not directly available as an example and some time had been put into socket programming in C.

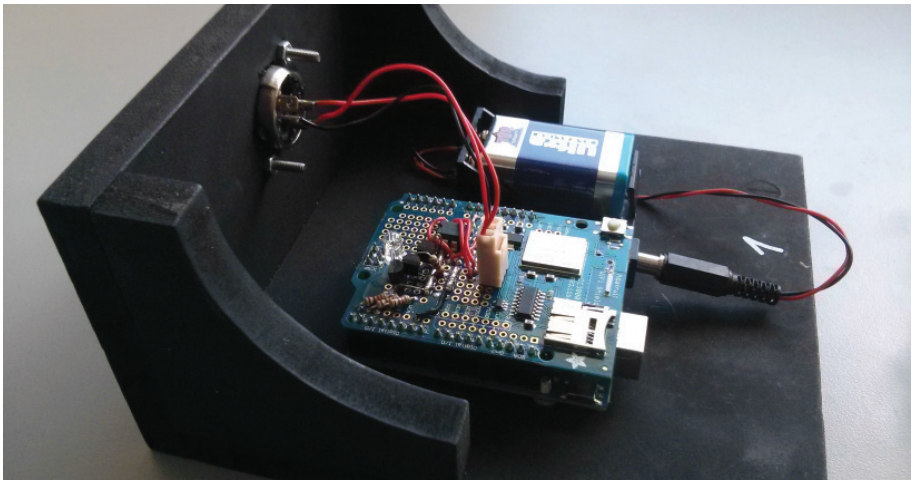

Figure 7.4: NLCP – DMX-Adapter

The Arduino is able to create a DMX signal (meaning the bit string), but this signal needs a little transformation so that it is compatible to RS-485, that is used by the DMX standard. To achieve this, a SN75176 RS-485 driver chip had been used. It is connected to one pin on the Arduino and to power. Depending on the level of the Arduino pin, the bus driver will generate an RS-485 suitable high or low signal. To generate a valid DMX signal, the DMXSimple[†] library had been used. It is really simple to use, only two settings must be made in order to get started. This is the pin that will output the signal and the number of channels that shall be created in a DMX packet. Additionally, a LED had been placed on the board, so that a visual information about connection or error can be displayed.

Unfortunately, this adapter is not able to receive full 1,500 byte packets. This is the problem of a setting in the CC3000 library. The maximum packet size is 95 bytes[‡]. However, for 2 lamps this is sufficient and for a prototype implementation this limitation can be made.

7.2.1 Function

The Arduino tries to connect to the specified access point, after this had been successfully performed, the NLCP server is started. Basically a socket is bound here to a specific port, so that all received packets on this port can be handled. A green LED shows that the the server is ready to receive packets.

The receive method from the server class is the first executed statement in the 'loop'. The execution of code is stopped until a packet arrives. Once the method returns, the packet can be found in the buffer. It is than checked if the message can be handled, that means

[†] `https://code.google.com/p/tinkerit/wiki/DmxSimple`
[‡] `https://forums.adafruit.com/viewtopic.php?f=31&t=53841`

if the first 6 bytes match the specification. If this check returns true, the message will be handled, which means that a Network Discovery Reply will be sent or that the light data will be forwarded to the lamp.

If the packet contains light data, the DAT is searched for the an entry that belongs to this device. If there is none, which can happen when more than one packet is send per interval, the code will return without doing anything. But if there is such an entry, the data are fetched and written to the DMX bus.

The full source code can be found on CD in appendix E

7.2.2 Summary

The NLCP – DMX adapter is a prototype implementation of a NLCP receiver. It does not act directly as a stage light, but provides a DMX interface, so that a common lamp can be driven by it. The adapter is based on an Arduino Uno and an Adafruit CC3000 wireless shield. It provides a basic set of features, so that it can work in an NLCP network. Unfortunately, the packet size must be limited dramatically when working with this device, meaning that not as many lights can be driven as it had been shown in the previous chapter. To overcome this, a micro controller with more RAM can be used, since this problem is just a problem of the available memeory, not the software running on it.

7.3 Conclusion

This chapter showed practical aspects of NLCP on the controller and receiver side. The programmed controller showed that a classical fader layout with a fixed amount of faders is far from optimal for NLCP. This is due to the changing addressing scheme. While DMX addressed every single feature in a network, NLCP addresses only the lights itself. This leads to the problem which fader maps to which feature on which lamp. To overcome this problem a responsive user interface had been proposed, that recognises the kind of a lamp and generates as many faders, button and check boxes as necessary. It keeps an eye on the values and creates packets for a specific lamp.

While implementing the controller the thought came up, how a multi packet must be structured in order to keep the amount of multi packets as small as possible. This problem can be expressed as the bin packing problem and cannot be solved optimal due to the NP-hardness of the problem. However, a heuristic can be used to find a good solution.

The amount of packets can be further reduced, and therefore the probability of packet loss, if packets are only sent when a value change. At least some of the lights might keep their data for a while, before getting new instructions. During this time no data needs to send to these lamps.

An adapter had been built to convert a NLCP packet into a DMX signal. The converter is based on an Arduino Uno and an Adafruit CC3000 wireless shield. Implementing this device showed that it is not always as easy as it seems. Memory constraints might be a problem when working with embedded systems. The packet size must therefore be limited to maximum 95 bytes when sending to these devices.

8 Future Work

This chapter will show points that should be explored further to improve this work. This can be amendments to proposed ideas and points that had been worked on due to time restrictions.

One important thing to consider is synchronization of lights in the NLCP network. Figure 4.3 shows how the displaying can be synchronized. However, this is only possible if the internal clock of the light is working very precisely. A problem here is that this internal clock might fade, when used in a long running application. Methods for synchronization of clocks must be developed and shown how these will affect the overall network performance.

The size of a multi packet had been set to 1,500 bytes to avoid fragmentation. It should be explored if this size can be increased. Regarding to figure 4.2, the throughput can be further increased if the size of the packet can be increased. This way the more lights can be operated in a NLCP network.

In this project only a small amount of packet had been defined. These packets are based on the functions of actual lamps out on the market today. More and more general packet should be defined in the future. This way more kinds of stage device can be driven, allowing the NLCP protocol to work in more scenarios.
Packets for backwards communication from the lamp to the controller should be defined an tested. This way notifications can be sent to the operator, informing him about malfunctions or the general status. However, this feature might influence the performance of the network dramatically, especially when all lamps implement this kind of communication. Defining a proper communication model and simulation seem to be very important before releasing this feature.

The field test should be repeated to test how different antennas, antenna positions and transmission power will affect the network performance. In some scenarios it might be sufficient to send a direct radio signal from the access point to the stage, covering a great distance, but only a small area. With these test information can be gained how the network behaves with different equipment.

The simulation should be further analysed, so that the packet loss can be described in more detail. It seems a bit too precise, that simulations on all 4 data-rates have exactly the same packet loss.
Throughout this project, it had been assumed that the controller is connected wireless to the access point and positioned close to it, while the lamp had been based on different positions. It would be very interesting to see how the network behaviour change when the

controller is a mobile device and moving around. The questions might be how will the range be affected by this and will the packet loss change?

Reliable Communication had been mentioned in the problem formulation. It would be a good idea to introduce methods for error correction in the NLCP protocol. This way errors can be detected and corrected, by now UDP props the packet when errors occur. Error correction would make the protocol more reliable, but also slower. It must be tested how error correction effects the delays.

9 Conclusion

This chapter is going to conclude the work that had been done in this project. It will review the problem formulation and compare it to the gotten results.

The problem formulation (chapter 3) formulated problems of current stage light protocols. That is:

- Only 512 addresses available per network
- No synchronization between networks
- Data are not reliable and no guarantee for delivery is given
- A physical cable connection is used for data transmission

Further on, the main focus of this work is set to increase the maximum amount of addresses per network and a wireless communication between stage lights and the controller.

To increase the number of addresses, the explicit address scheme of DMX (which gives every feature an individual address) had been dropped. By addressing always a complete lamp, the number of needed addresses will be reduced. To make all features still available for a controller, new packets had been defined.

The number of addresses is only one part of the solution, the physical layer must also be able to transmit these data. Since a wireless solution had been preferred, the 802.11 standard had been explored. Three approaches came to a topic:

- Wireless mesh network (802.11s)
- Wireless ad-hoc network
- Wireless network in infrastructure mode

An infrastructure network seemed to be most promising for this task. Due to restrictions on some hardware (i.e. Android), which do not allow ad-hoc mode, the small additional effort of setting up an access point might be justified. With this as a basis, it had been explored how the controller can structure the data packets. The initial idea of sending individual packets to each lamp had been discarded. Instead a multi packet of size 1,500 bytes had been introduced that wraps a number of individual (so called single-) packets. Multi packets are always sent to the broadcast address. This approach is similar to DMX. The advantage is that the theoretical maximum throughput is dependant on the packet size and increases with larger packets.

To see how the network behaves and to find the maximum number of devices that can be driven, a simulation had been programmed. The OMNeT++ simulator, together with the INET framework, did a good job for this scenario. The simulation had bee performed with four different data-rates (6, 18, 36 and 54 Mbps), because the assumption was that with a higher data-rate more packet can be sent. This had been proved. The results of the simulation is that a maximum of four packet can be handled at 6 Mbps, so that the packets arrive still in time. Nine packets at 18 Mbps, 14 at 36 Mbps and 16 packets at 54 Mbps.

To see the maximum range that can be covered, a field test had been performed. At different data-rates packets had been sent from a laptop, via an access point, to a receiver. The test should show how many packets get lost when the distance between the access point and the receiver increases. Different data-rates might have an impact on this. It had been showed that with a lower data-rate a larger area can be covered than with a high data-rate. A maximum of 8m at 54 Mbps can be accepted, 14m at 36 Mbps, 23m at 18 Mbps and 28m at 6 Mbps. However, the distances can be increased when a higher transmission power or different antenna is used.

All these test produced a lot of data, that abstract from the actual problem. To see how the protocol works in the real world, a controller and receiver had been implemented. The controller is a Java application and run from the laptop. It generated packets based on the NLCP specification. While implementing, it had been discovered, that the amount of packets can be reduced, and therefore the probability of packet loss, by sending only data to the lamps which actually changed. Also the ordering of the single packets (bin packing problem) can have an impact on the number of multi packets. The idea of a responsive user interface is not directly related to networking, but shows that a new protocol will affect the way a controller is designed.

For the receiver, a C-program had been written in order to receive and process NLCP packets. The device itself is based on an Arduino and a wireless shield. Implementing it, showed practical problems concerning the RAM can occur and must be considered when building similar units.

Concluding, the project was successful. A protocol had been defined and tested. These test showed that the address range increased and the communication is all wireless.

Bibliography

[1] Artistic Licence Holdings Ltd. Art-Net 3, Specification for the Art-Net 3 Ethernet Communication Protocol. Document Revision 1.4bf 3/3/2014, 03 2014.

[2] M. S. Gast. *802.11 Wireless Networks – The Definite Guide*. O'Reilly, first edition edition, 2002.

[3] Jangeun Jun, Pushkin Peddabachagari, Mihail Sichitiu. Theoretical Maximum Throughput of IEEE 802.11 and its Applications. In *Proceedings of the second IEEE International Symposium on Network Computing and Applications (NCA'03)*, 2003.

[4] Normenausschuß für Bühnentechnik in Theatern und Mehrzweckhallen (FNTh). *DIN–Taschenbuch 368, Veranstaltungstechnik 2 – Theater,– Studio– und Hallentechnik*, pages 489 – 497. Beuth, DIN Deutsches Institut für Normung e.V., Mar. 2000. Part 1: Begriffe und Anforderungen, Part 2: Steuersignale.

[5] Simon Thorpe, Denis Fize, Catherine Marlot. Speed of processing in the human visual system. *Nature*, 1996. volume 381.

[6] Wikipedia. Bühne (Theater). `http://de.wikipedia.org/wiki/B%C3%BChne_(Theater)`, 2014. [Online; accessed 11-March-2014].

[7] Wikipedia. DMX512. `http://en.wikipedia.org/wiki/DMX512`, 2014. [Online; accessed 10-March-2014].

[8] Wikipedia. EIA–485. `http://de.wikipedia.org/wiki/EIA-485`, 2014. [Online; accessed 11-March-2014].

[9] Wikipedia. Fresnelzone. `http://de.wikipedia.org/wiki/Fresnelzone`, 2014. Online; accessed 27-May-2014.

[10] Wikipedia. Glühlampe. `http://de.wikipedia.org/wiki/Gl%C3%BChbirne`, 2014. [Online; accessed 11-March-2014].

[11] Wikipedia. Hashfunktion. `http://de.wikipedia.org/wiki/Hashfunktion`, 2014. [Online; accessed 11-April-2014].

[12] Wikipedia. Ringtheater. `http://en.wikipedia.org/wiki/Ringtheater`, 2014. [Online; accessed 11-March-2014].

[13] Wireless Solution Sweden AB, Stureparksvägen 7, SE-45155 Uddevalla, Sweden. *W–DMX 2013, G4 CATALOG*, 2013. Revision 1.

[14] Wireless Solution Sweden AB, Stureparksvägen 7, 45155 Uddevalla, Sweden. *W–DMX^{TM} BlackBox, user manual*, 2014? Release v3.0g.

[15] Ying-Dar Lin, Shun-Lee Chang, Jui-Hung Yeh, Shau-Yu Cheng. Indoor deployment of 802.11s mesh networks: Lessons and guidlines. *Ad Hoc Networks*, 2011.