GHENT UNIVERSITY

Bachelor Thesis

VOP Openwifi

2020

Supervisors:
prof. dr. ir. Ingrid Moerman

Advisors:
prof. dr. Xianjun Jiao
dr. Wei Liu

Authors:
Jonas De Busscher
Lander Verloove
Lukas Demey

Project within the framework of the course

Cross-Course Project

in the

Bachelor of Science in Electrical Engineering

FACULTY OF ENGINEERING AND ARCHITECTURE

Contents

1 Introduction

In this report we will discuss the progress and the technical elaboration of our bachelor thesis project 'Openwifi: open-source IEEE802.11/Wi-Fi baseband chip/FPGA design'. In total we worked eleven weeks on the project.
The first weeks were used for an extensive research in order to have a good understanding of the open source project. This was mandatory for an optimal further progress of the project. The results of this research will be briefly recapitulated in the section 'Openwifi Chip' of the report. At the same time, this section will give the reader an orientation and global understanding of the project and the according chip design.
After these orientation weeks, we looked at several factors to determine the further direction of the project. Here, we laid together our knowledge thus far and our major interests. In consultation with our advisor Xianjun Jiao, we eventually came to the following task: the optimalization of Wi-Fi communication in emergency cases. Next to providing a useful application, this task shows, at the hand of an example, the relevancy of the Openwifi project as the application makes use of functionalities that are exclusive to the Openwifi chip and are not found in regular commercial chips. A detailed technical elaboration and implementation can be found in the section 'Application'. In short, we developed a protocol that senses the congestion grade of a network and implements a method to 'free up' the network in order that only the most important users are offered a secure and guaranteed communication.
Next to this technical learning journey, the cross-course project provided an environment to learn the art of planning, group communication and many other aspects that come with an independent group project. Our approach and experiences of our project management are elaborated in the section 'Project Management'.
At last, we would like to thank our supervisor, Ingrid Moerman, and advisors, Xianjun Jiao and Wei Liu, for the great learning environment provided and the received help during the whole trajectory of the project.

2 The Openwifi chip

In the first three weeks of the project, we focused on research in the study area where the open source project Openwifi is situated. As the project is a chip design according to the IEEE802.11/Wi-Fi protocol, a basic study of different protocols and aspects around the IEEE802.11 protocol was needed. After this global orientation, we dove into the open-source Openwifi Github repository. This knowledge combined with an introduction lecture, given by our advisor Xianjun Jiao, gave a good basic global idea of the chip design and its role in the internet stack. A summary/explanation can be found in what follows. At the same time, this section provides the reader the basic knowledge needed to understand the further report that is the 'emergency communication'-application. Sources used can be found in the bibliography of the report.

2.1 Basic required knowledge

The IEEE802.11/WiFi [1] is a set of standards for wireless local area networks created by the Institute of Electrical and Electronics Engineers. It provides restrictions and instructions on the specifications of the shared medium access and the physical layer. This shared medium acces is defined in 802.11 MAC protocol [2]. The physical layer is defined in the 802.11 PHY protocol. In Table 1, a graphical representation of the role and placement of the two protocols in the internet stack is given.

Application layer	
Transport layer	
Network layer	
Link layer	LLC 802 802.11 MAC
Physical layer	802.11 PHY

Table 1: Orientation of 802.11 in internet stack

The 802.11 MAC protocol handles the medium access and does this with the use of a random access protocol Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). The basic working method is represented in Figure 1. For the further report, it suffices to understand this figure. For a more thorough description, we refer to [3].

CSMA/CA protocol:

1. If station senses the channel idle, it transmits its frame after a short period of time known as the Distributed Inter-frame Space (DIFS).
2. Otherwise, the station chooses a random back-off value using binary exponential back-off and counts down this value after DIFS when the channel is idle. While the channel is sensed busy, the counter value remains frozen.
3. When the counter reaches zero (note that this can only occur while the channel is idle), the station transmits the entire frame and then waits for an acknowledgement.
4. If an acknowledgment is received, the transmitting station knows that its frame has been correctly received at the destination station. If the station has another frame to send, it begins the CSMA/CA protocol at step 2. If the acknowledgment is not received, the transmitting station reenters the back-off phase in step 2, with the random value chosen from a larger interval.

Figure 1: CSMA with Collision Avoidance protocol [3]

This medium access protocol makes it possible for different chips that happen to operate in same frequency range, to communicate properly without disturbing each others signal. This provides an ideal protocol but does not cover the collision of two clients that both see the access point but are not able to see each other. For this, the 802.11 also includes an optional reservation scheme with the use of 'request to send'-and 'clear to send'-packets (RTS/CTS). In this scheme, a client who wants to transmit a packet to the access point first has to send a short Request to Send control frame. The access point then broadcasts a Clear to Send packet to all its connected clients. This frame gives the sender permission to send and also warns other stations that they can not transmit. The RTS fragments are still subject to possible collisions but the length of these packets is short and thus such a collision does not have a significant influence on the throughput. We refer to [3] for a more detailed description.

Next to the link layer, Wi-Fi also defines the protocol 802.11 PHY that operates on the physical layer. This layer comprises the electrical hardware and software needed to send raw bits to each other. The medium that is used in WiFi are electromagnetic waves that are transmitted and received with antennas. These electromagnetic waves have a frequency ranging from $5.150GHz$ to $5.850GHz$ for $5GHz$ Wi-Fi, that is divided in different channels. Each channel can be used for communication between two chips and has its own center frequency. The specific channel used for communication depends on the role of the higher layers by which the chip is controlled. In an ad-hoc network, this is a network where clients connect with central access points, the channel is chosen as follows. When initialising a router access point, a channel on which it will operate has to be chosen manually. The 802.11 standard requires that access points then periodically send *beacon* frames on their channel, these contain the information needed to associate with this access point. A clients device scans all the channels for beacon frames in order to determine which access points are reachable. With the use of the SSID, and the other information that is contained in the frames, like the supported services and a time for synchronisation, a connection can be set up for wireless communication. If a client does no longer detect these beacons, it will assume that the access point is out of reach.

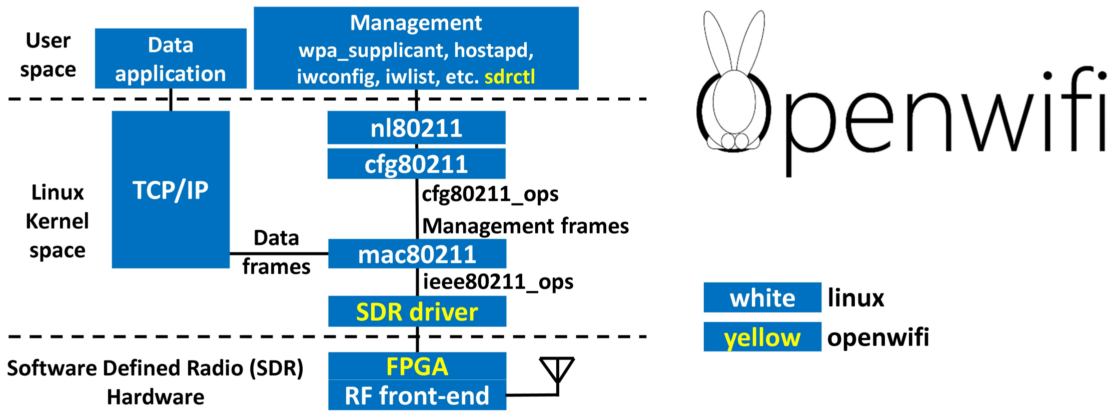


Figure 2: Openwifi architecture

2.2 Orientation of the open source project

A chip that conforms to the IEEE802.11 standards receives a Wi-Fi certificate and can be called a WiFi-chip. Every manufacturer can receive such a certificate as long as they conform to the constraints. This chip can then be used in devices that work according to the TCP/IP-protocol stack. In this way, devices could use the WiFi-chip as medium to send or receive data fragments. To pursue this, a hardware and software socket is needed to 'connect' the network layer with the data link layer. This connection point is called the driver of the WiFi-chip. This driver is created by the manufacturer of the WiFi-chip and offered to the developer of the higher layer application. The specific design and implementation of the WiFi-chip is most of the time not shared. They are black-boxes that only have an accessible input and output.

Now, most of the commercial available Wi-Fi chips are designed for day-to-day internet usage. Think of a user communicating with a wireless access point in a restaurant. Although these are the most common situations, WiFi can also be used in much more other applications. For a developer of such an application, it is a point of struggle that only the input and output is visible. As example, imagine two Wi-Fi chips implemented in a internet-of-things doorbell. The camera at the door sends its captured image to a screen in the living room and uses a direct Wi-Fi connection to the screen for this. If the developer would like to make an intelligent Wi-Fi chip that calculates the minimum of power needed to just reach the screen. This would of course lead to a more energy-friendly design. Nevertheless, this would be a very hard job when an ordinary Wi-Fi chip is used. Also researcher would have a great improvement of their arsenal if the internal parameters of the Wi-Fi chip would be available.

It is at this point of potential improvement that the Openwifi project finds its basis. With an open source Linux max80211 compatible full-stack IEEE802.11/Wi-Fi design based on software defined radio, developers and researches are offered a basis design that is fully accessible and can be used and adjusted according to their specific constraints.

2.3 Technical explanation of the design

The software and hardware making up Openwifi can be divided in 2 parts. A part that is designed specifically for Openwifi and a part implemented in Linux. The Openwifi-chip can also be divided in a 3 layer stack. The first layer in the stack is software defined radio. This layer is made up by the hardware. The second layer is the Linux kernel space and the third layer is the user space layer. These last 2 layers are completely implemented in software. In each of the 3 layers further distinguishes can be made. An illustration of this implementation is showed in figure 2. Another distinction that can be made is that a part of it is implemented in FPGA and the other part runs on an ARM processor.

2.3.1 User space

In the user space, software is implemented to facilitate the handling of the implemented functionality of applications and protocols lower down the stack. The software makes it possible to control this functionality by editing config files and executing commands in the Linux terminal. Hostapd is an example of such software and stands for Host Access Point Deamon. The sofware helps to make an access point from a network interface card and let it function as an authentication server. Simply by editing the config file, the SSID, channel, passphrase can be altered. Other examples of user space software are wpa_supplicant, iwconfig and iwlist. Another user space program is sdrctl which is created specifically for the Openwifi-chip. With this tool, the values of the registers can be changed. In this way it is possible to communicate with the driver.

2.3.2 Linux kernel space

The Linux Kernel space contains the mac layer up to the transport layer of the internet protocol stack. The kernel acts as an interface between both the Hardware and the user space. The interface between the hardware and the Linux is formed by the driver. The driver contains a set of APIs named *IEEE_80211_ops* that can be called from the Linux subsystem mac802.11. An example is the tx API, which is called when the upper-layer has a packet to send. The driver will process the packet and pass it to the FPGA. This also works the other way around. When a packet is received, the FPGA will pass the packet to the driver and the driver will deliver the packet to the Linux subsystem on its turn. The previously mentioned Linux subsystem together with the cfg80211 makes up a description of the physical layer that can be used by the Linux kernel to control the driver. The nl80211 forms the interface between cfg80211 and the user space. This layer and the user space both completely run on the ARM processor.

2.3.3 Software defined radio

When the driver has a packet ready for transmission, it gives it to the FPGA where it is stored in the buffer. The FPGA then decides when the packet should be transmitted based on the existing protocols. CSMA/CA in combination with Clear Channel Assessment is an example of protocols that have an influence on the time when a packet send. The FPGA is also used to control the software defined radio which is used to modulate, transmit and receive the packets over the antennas.

3 Application

After the orientation phase, we needed to determine the further trajectory of the project. For this we laid together our gained knowledge of the IEEE802.11 protocol next to the special specifications and implementations provided by the 0penWifi design. Eventually, in consultation with our advisor Xianjun Jiao, we decided to build a protocol to attack a problem around emergency communication that could benefit from the exclusive control possibilities offered by the Openwifi chip design. In this section we will discuss this problem statement, our proposed solution and its implementation thoroughly.

3.1 Problem statement

At medium to large sized events, like music festivals, there is always the posibility that an emergency will occur. In that case people must react quickly and contact the emergency services. Nowadays on these kinds of events, security and first aid can be contacted with Walkie Talkies.

This project will try to implement a Wi-Fi system that can be used by guests, but will give priority to the staff members in an emergency case. Using Wi-Fi to implement a highly reliable system comes with its own problems. If a lot of visitors are using the Wi-Fi, the network can get congested, there will be high latency and packets will be dropped.

It is no option to just retransmit all the packets for which no acknowledgement is received using a protocol like TCP, because if the packet loss or latency is very bad, it will not be possible to connect to a receiver in the first place. So a special protocol was developed to make sure that the data will reach the receiver, even if the network is congested.

Given that the usecase for this project is clearly defined, some assumptions can be made, which will justify the design choises:

- The network in which this will be used, can be classified as a multi-hop ad-hoc wireless network
- Given the size of a festival area and the range of cellphone Wi-Fi in an open field, the assumption can be made that the hop count will be limited, minimising the chance of packet drops

This last assumption needs some justification. Say that this project will be used on a festival like Rock Werchter, which is already a large one, the festival area is approximately $260.000m^2$ and the amount of routers that need to be placed will mainly be determined by the range of a commercial wiFi chip operating in $5GHz$ network, which is approximately $60m$ in an open space according to [4]. This means that the entire area can be covered with 30 routers accoring to a paper from the University of Technology of Helsinki[5]. Using the estimated optimal covering, a maximum hop count of 7 is obtained as can be seen on Figure 3.

3.2 Proposed solution

3.2.1 Concept 1

The first concept was pretty straightforward. The sender would just brute force its way into the network. When an emergency occurred, the sender would use the Openwifi chip to turn off CSMA and start spamming special reservation frames. When these frames reached a router, it would kick all other users and just communicate with the sender of that special packet as seen on Figure 4. The main advantage of this approach was that the packet would eventually reach the receiver, setting up a direct communication link between the emergency services and the staff.

As the other users having CSMA enabled, they would quickly stop sending data. Just turning of CSMA and not sending a reservation frame, is not an option due to the hidden terminal problem. When

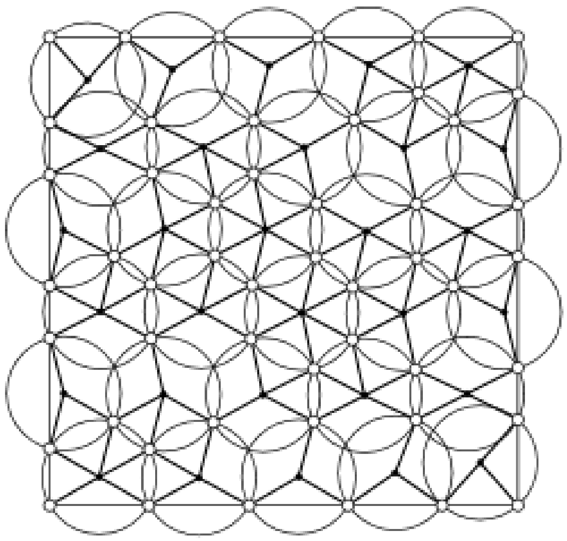


Figure 3: covering a square with 30 equal circles in an efficient way

another device is sending to the router, but the signal of the sender does not reach that device, they can both send and cause interference at the router. Also, this approach would be able to split certain network topologies, rendering the rest of the network useless. This approach was also deemed too aggressive.

3.2.2 Concept 2

Just like in the first solution, the intention is to reliably send the data in an emergency case. So the first improvement to the previous concept, is to probe the network first to see if there is high latency or packet loss. If the network is deemed reliable, no action will need to be taken and the data can just be sent while occasionally probing the network again to see if no problems rise up. If the network is estimated to be not reliable enough, the more aggressive approach will be taken. CSMA will be turned off and instead of spamming the reservation frame, a CTS-to-self package will be sent followed by the reservation frame until the router acknowledges that the channel is reserved as can be seen in 5. The technical approach of the probing and the action taken when network is congested, will be elaborated in the rest of the section.

The approach of this second concept is better because we eliminate the possibility that another user is able to send data before the new reservation packets arrives to reset its backoff timer as CTS_to_self will cause it to stop sending for a predetermined time or whenever an acknowledgement is received. Also the behaviour of the reservation packet was changed. It will no longer just kick its users. The time between beacons will be set very high and the reservation acknowledgement sends back the channel SSID, so that users associated with this router will no longer think it is there. Using that SSID, the staff member trying to contact the emergency services will still be able to connect to that router to send its data.

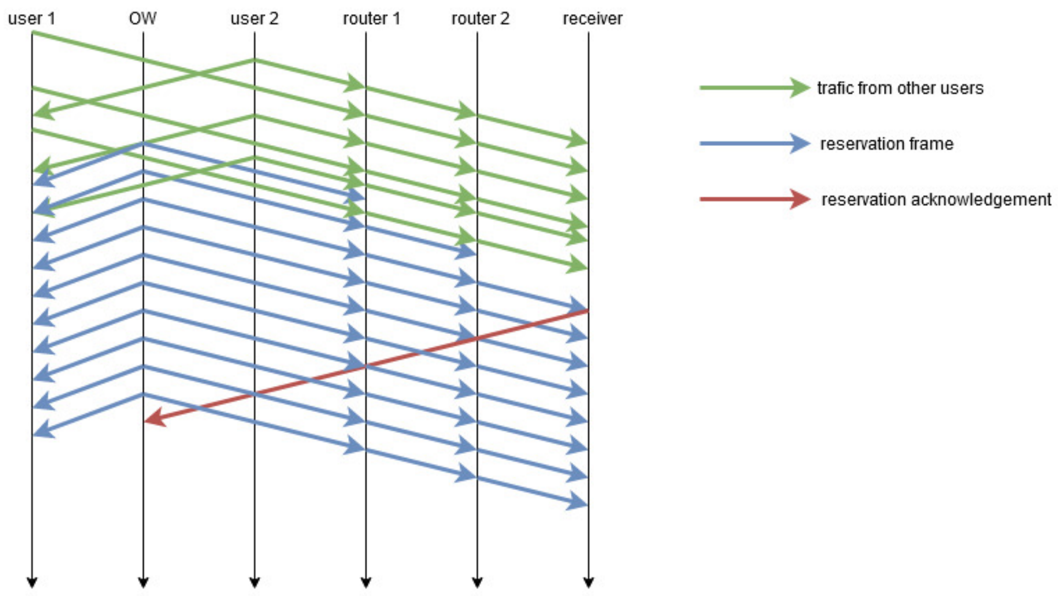


Figure 4: reserving of the channel as intended in concept 1

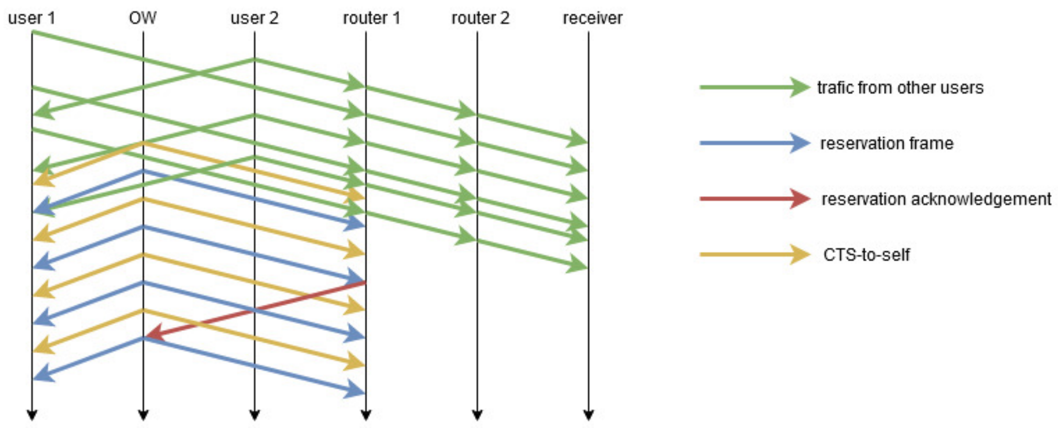


Figure 5: reserving of the channel as intended in concept 2

3.3 Overall principle of the protocol

As can be seen in figure 6, the first step is to probe the network and check its latency. For this ICMP was used to get an estimate of the uplink and downlink latency. Based on the estimated uplink latency, the protocol will decide whether or not to take over the network. Only uplink latency is considered because data has to reach the emergency services and not the other way around. For the actual communication UDP is used, with possibly an integrated network probe. Once the file is sent completely, CSMA will be turned back on.

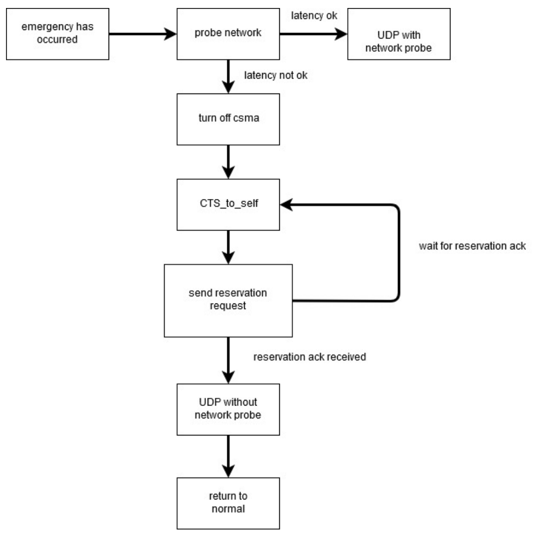


Figure 6: protocol flowchart

3.4 Technical implementation of protocol

3.4.1 Network probing algorithm

To probe the network, ICMP packets of type 1 and type 2 are used. Both ICMP types are not being used for any application at the moment, so they are free to use.

Just regular ping packets (type 0 and 8) or timestamp packets (type 13) would not suffice. Because ping only reports the round trip time and in this case the uplink latency is what is interesting. Timestamp does have the ability to measure uplink latency, but its packet size is rather small and a large packet must be used such that its probability of being dropped is higher than that of the actual data that needs to be sent.
As can be seen on Figure 8, the sender starts by sending 4 packets of type 1 where the ID field is 0 and the data field contains a timestamp and characters to fill it up such that the packet size is 1024 bytes, which is equal to the MTU of the Openwifi board. The receiver listens for those packets and records the time at the moment that the packet arrives. This time is chosen in favor of the time it was sent back because it is the uplink latency that is interesting. If the receiver receives 4 packets of type 1 within 2 seconds of the first arrival, it will use byte 17 to 34 of the data field to include the timestamp en send it back in a type 2 packet where the ID field is set to 0. However, if the receiver does not receive 4 packets within that time or the checksum indicates that something has gone wrong, it will assume that the network is not reliable. In response 4 packets of type 2 with ID field set to 1 and no data will be sent back to the sender, signaling that the network is not reliable enough for data transfer.

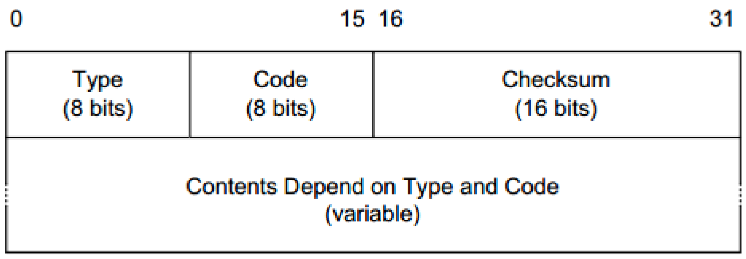


Figure 7: structure of an ICMP packet

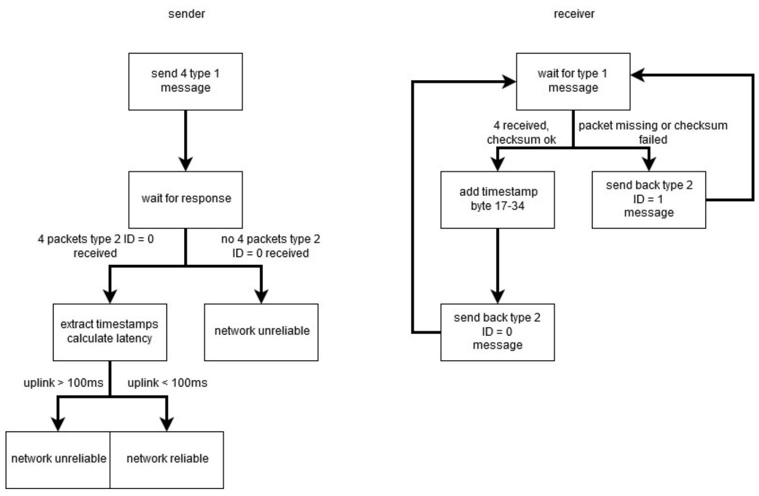


Figure 8: algorithm of sender and receiver of network probe

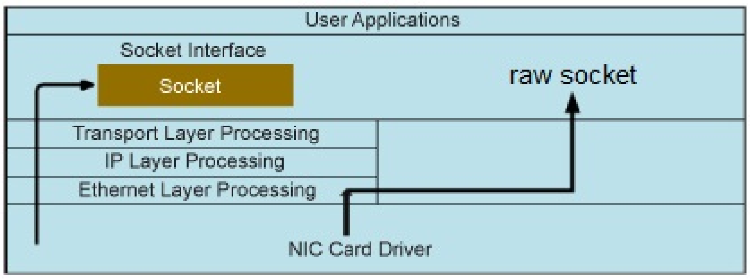


Figure 9: representation of raw socket in the OSI-model

3.4.2 Why ICMP?

There are a couple of ways to probe the network and check the latency, but here ICMP was chosen over UDP or TCP because of the following reasons:

- TCP requires a connection to measure latency, which is not possible in a congested network
- ICMP messages have lower priority than UDP for routers and will be dropped faster if there is a lot of traffic
- a ICMP socket should be programmed anyway if some kind of debug information must be printed at the terminal

That the ICMP is a lower priority protocol is an advantage in this case as the emergency data will be sent over UDP and if ICMP does not get dropped, UDP most likely won't either. This also means that the latency for UDP packets will be lower than expected, as ICMP has lower priority and thus will take longer to get processed. This gives an upper bound on the uplink latency.

3.4.3 Python implementation

All software was programmed in python, making use of raw sockets. Raw sockets are different from regular sockets as the received packets skip most of the network stack an get delivered straight to the application layer, as can be seen on Figure 9. This allows for lower level access to the packet and is the only way to send or receive ICMP messages from the application layer. This means that the IP of the sender can be extracted and the ICMP messages can be sorted based by type.

Python has a lot of different functions to time something, but time.time() is the only one that has a defined reference point. It returns a floating point value of the amount of seconds that have past since January 1, 1970, 00:00:00 (UTC) with a resolution of $239ns$ and since uplink latency will likely be measured in milliseconds in this application, this resolution is good enough. This function does not guarantee that the output will be rising monotonically. It is possible that the system has set back its clock between two calls, that the return value is lower than the value before that.

As the timestamp has to be put in the data field, it has to be converted to string and back to a float for calculations. If this method is used:

$$\text{float(string(timestamp))}$$

only 2 decimal places remain (not rounded), reducing the precision to $10ms$. So to mitigate this problem, the timestamp is first multiplied by 1000 and rounded to 2 decimal places, this way a precision of $10\mu s$ is achieved, which is good enough for the application.

3.4.4 Network is sensed congested by the probe

When the network is found under heavy load by the probing, the protocol will initiate a freeing up of the network as follows. First the CSMA/CA of the users Openwifi-chip will be turned off. In this way the chip will send packages straight away without sensing if the channel is idle. Next to this, the chip will operate the CTS-to-self method. In this, the chip will send a clear-to-send package before it wants to transmit its package with important information. Through this way it can reserve a channel and make sure that the other clients will not try to transmit something. By first turning off the CSMA/CA, the repeated CTS-to-self packet spam will eventually get through and the clients will stop transmitting for a while. In figure 5 one can see the working of this mechanism. The yellow packet is the CTS packet sent by the Openwifi client. Because of the neglecting of CSMA this packet will be transmitted even though it senses that User 2 is transmitting. As users 1 and 2 are still operating following the CSMA/CA protocol, they will sense the channel used as they notice Openwifi is transmitting. They will enter back-off. The first package send by Openwifi chip will not be received properly by the router and users, as the other users were also transmitting at the same moment. However, as the Openwifi chip is spamming these packages and the other users are quiet because of the CSMA/CA protocol, the CTS package will eventually be received properly by the users and router. At this point the users are instructed by this packet to be quite and the Openwifi chip has a higher chance at a proper communication with the router. An important connotation in this, is the fact this doesn't provide guaranteed communication with the router. This is because of the hidden terminal problem. The CTS-packet sent by the Openwifi user is only received by other clients that are in range of its chip and not necessarily by all the clients connected to the router.

The clear-to-send frame is followed by an ICMP message of type 44. This packet has the purpose of instructing the router to set its inter-beacon time to 30 seconds. A consequence of this will be that other users nearby the router will no longer receive beacons and think the router is not operable anymore, to avoid the kicking of the Openwifi chip itself, a special packet is sent from the router to the Openwifi chip with the essential information of the router to ensure a persistent connection despite it no longer receives beacons.

3.4.5 Sending data

At last the data can be sent to the receiver. This will be done using UDP, but if the network was deemed reliable before sending, it will be probed every n packets to see if it still satisfies the conditions. The value of n is determined based on the data efficiency that is requested by the user. Let data efficiency be:

$$\eta = \frac{n * information\_packet\_size}{n * information\_packet\_size + 4 * probe\_packet\_size}$$

solving for n yields:

$$n = \frac{4 * probe\_packet\_size * \eta}{n * information\_packet\_size * (1 - \eta)}$$

If the network is deemed unreliable this time, the emergency data is sent again without CSMA enabled and using CTS-to-self.

UDP is used because if the channel is reserved, there is no other traffic so it will be fairly certain

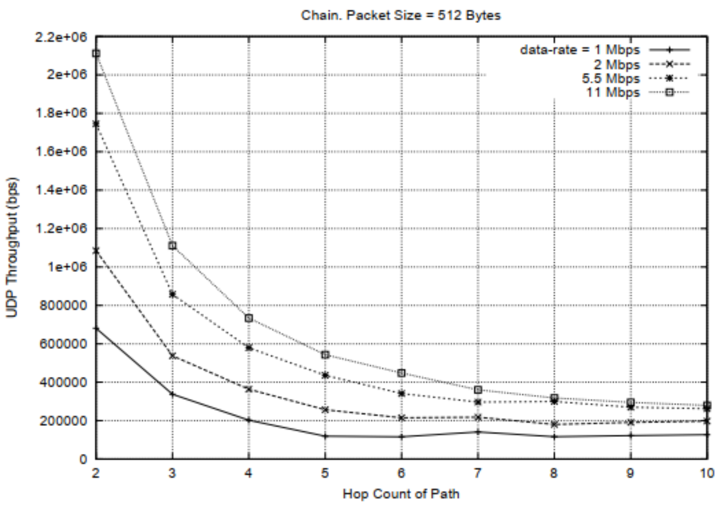


Figure 10: relation between hop count and udp throughput

that the packets will reach the destination, if the amount of hops is not too large. Even if the channel is not reserved, this project is intended to be used in a relatively small setting where the amount of hops between sender and receiver is minimal so that the chance of error is minimal as can be seen on Figure 10 from a study done on UDP performance in multi-hop wireless networks [6].

The file that has to be sent, is known to be really small. It's data is used to describe an emergency state, so the file will only be a couple of pictures or a 30 second movie. This amount of data can be transferred quickly using almost any throughput. So a throughput can be chosen such that packet loss will almost never occur. This statement is backed up by a study of UDP performance analysis of Ad Hoc Camera Networks transferring data via UDP [7], The result this claim is based on is pictured on Figure 11. This result also justifies the choice to use smaller UDP packets to send the actual data.

3.5 Results

To ensure that our protocol works, we should test our protocol in an environment where congestion or a heavy load on the wireless network can be simulated. The best that could be done to create such an environment was using a phone, 2 laptops and a commercial AP. One phone and a laptop are streaming multimedia applications as Netflix and Youtube and 1 laptop is spamming ping to the router. To conduct the test a desktop with wireless Wi-Fi adapter and the Openwifi board are used. The wireless adapter is connected to the Openwifi AP. This setup is shown in figure 12. All the devices are using the 802.11a protocol in channel 44 using a center frequency of 5.22GHz.

The simulation environment is far from ideal. The data sent by the devices to generate a load on the network is not enough to have a clearly visible impact on the performance of the test results. The results of the tests are highly vulnerable to changes in the setup or environment. If the board or the cable to control the board are touched, the performance of the transmitting speed or connection is affected in a negative way. By example, using the build in ping function in Linux, the average rtt over 50 packets can change from less than 2ms to 100ms to 2ms over a time span of a few seconds.

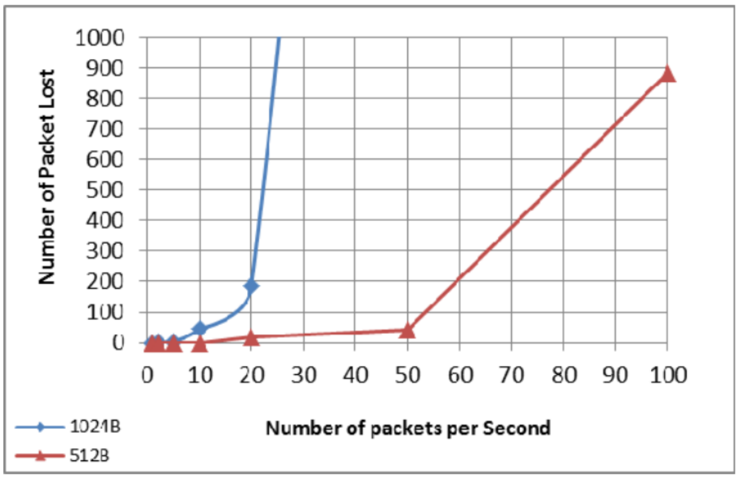


Figure 11: relation between packet loss and throughput, 500 seconds test duration

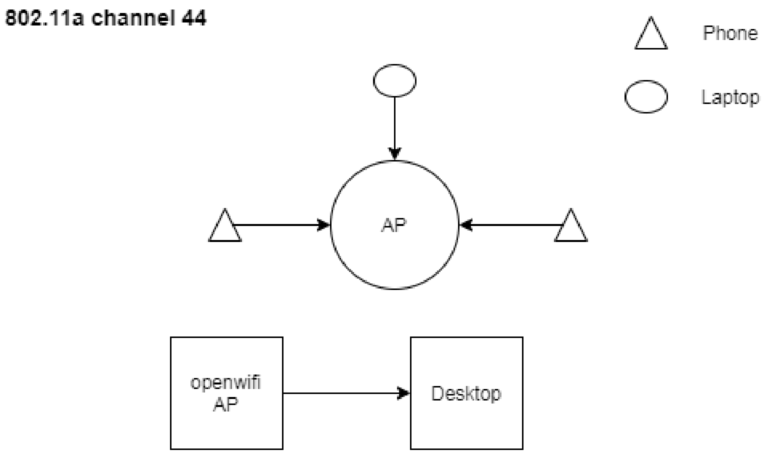
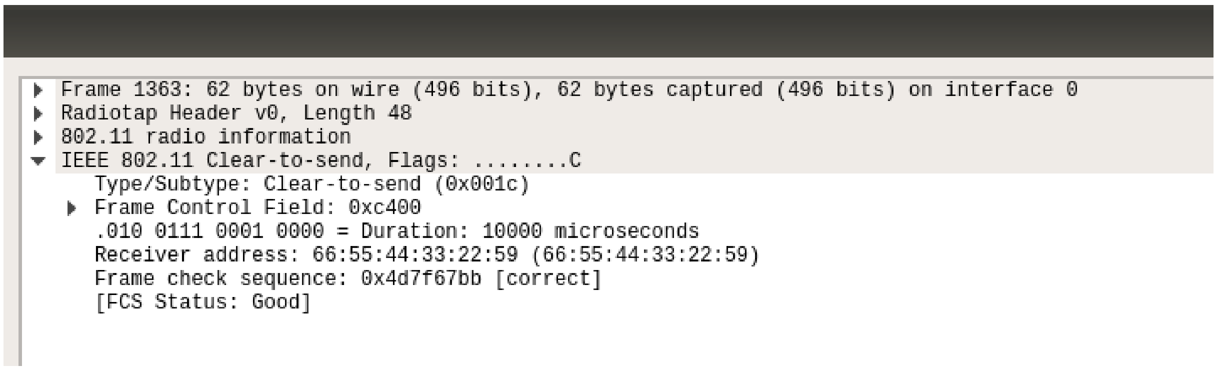


Figure 12: Testing-setup



```
▸ Frame 1363: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0
▸ Radiotap Header v0, Length 48
▸ 802.11 radio information
▾ IEEE 802.11 Clear-to-send, Flags: ........C
    Type/Subtype: Clear-to-send (0x001c)
  ▸ Frame Control Field: 0xc400
    .010 0111 0001 0000 = Duration: 10000 microseconds
    Receiver address: 66:55:44:33:22:59 (66:55:44:33:22:59)
    Frame check sequence: 0x4d7f67bb [correct]
    [FCS Status: Good]
```

Figure 13: Captured CTS-to-self packet in Wireshark

3.5.1 CSMA/CA

To test the effect of CSMA/CA, 100 UDP packets are sent using a python socket. The time between when 2 packets are send in Python is then calculated. This approach however, was too naive. Since there was no measurable difference between the two packets for CSMA turned off and CSMA turned on. This is because we measure the time it takes to send 2 successive packets to the buffer and thus we don't know when a packet left the device. The same holds for the the time between 2 successive received packets when they are read from the buffer. A less naive approach is by using Wireshark which could give a more precise timing measurement. The disadvantage of the use Wireshark is the difficulty of processing the data. By using Wireshark the time between 2 packets should be calculated manually. To get more accurate result the average time between transmitted packets should be calculated over a large amount of packets. This is an unworkable method in Wireshark. An ideal approach would be to capture the packets right before they are send over the antenna. This could possibly be done by capturing the signals of the FPGA. Such an approach should be used to get accurate data from the performance of our protocol. However, we did not have the necessary tools available.

3.5.2 CTS-to-self

The effect of CTS-to-self is tested by turning on the monitor mode on the Wi-Fi interface of the desktop. By doing this, we are able to capture the CTS-to-self packets send by the Openwifi AP. The duration of the CTS-to-self was set to 100 ms, 1000 ms and 10000 ms respectively for 3 test cases. Devices were connected to the Openwifi AP. The effect of the CTS-to-self messages is visible on the ongoing traffic in Wireshark. No notable difference in performance was noted on the devices using real time voice over ip applications. A screenshot of a captured CTS-to-self packet is showed in figure 13.

3.5.3 Network probe

As the network probe relies on latency measurements, a comparison was done between the linux ping and our implementation of latency measurement. In the circumstances that we were in, it was nog possible to control the kind of traffic that was on the network. So the most reliable results that can be obtained in a home network setting will be when nobody is using the network. Around 3PM a group member got up and disabled all network devices except for the router and the computers used to test. We expected that the latency would worsen if the packet size went up and that in this setting the uplink and downlink

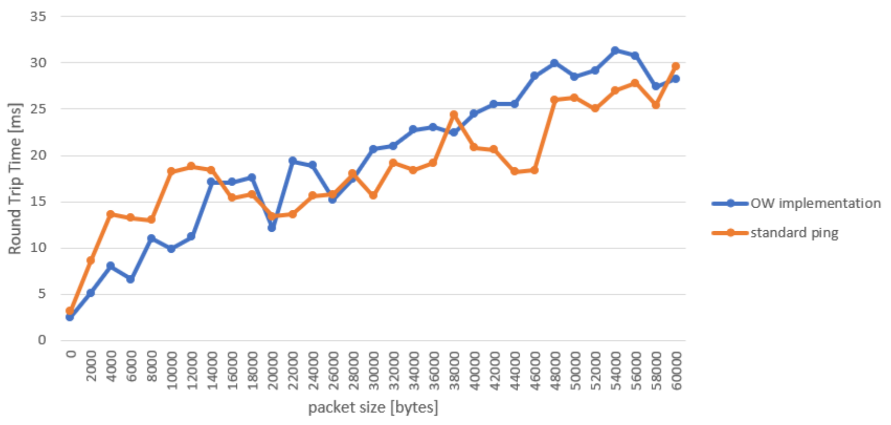


Figure 14: relation between hop count and udp throughput

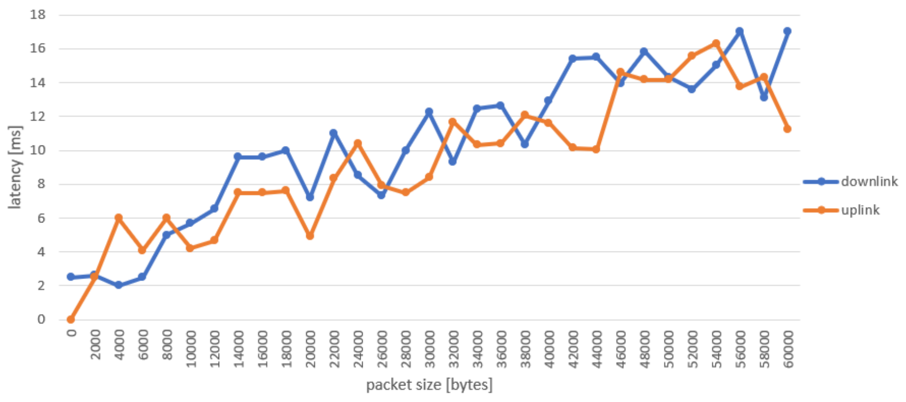


Figure 15: relation between latency and packet size

latency should be aproximatly equal and follow the same trend. The results can be seen on Figure 14 and Figure 15.

Just as expected, the uplink and downlink latency in this setting are approximately equal and we saw that the average deviation of the latency measured by the standard ping program and our implementation is 1.04ms with the standard ping reporting lower latency.

3.5.4 Conclusion

The results achieved by testing does not give confirmation that our protocol works. Although this does not mean that our protocol does not work. With the use of more correct test methods the validity of our protocol could still be proven. One of these methods is generating a more heavy load on the network. A second method is amplifying the output of the Openwifi device so it can reach the other users on the network. At last using a more precise timing measurement will help obtaining more accurate results. Improving all three of these conditions could support our claim.

3.6 Connotations

Before introducing a new solution, it is important to compare the performance and functionality with existing solutions, which are on the market. A product that has a comparable functionality as ours is the Industrial WLAN from Siemens. This product has emergency stop functions implemented for critical conditions. Examples that Siemens gives for this are the control of amusement parks and machinery in the automotive industry, where a reliable emergency stop is of high importance. This is similar to the protocol we try to implement in combination with the Openwifi chip. Our implementation only requires a device with an Openwifi chip and software which runs on the AP. The solution of Siemens differs in the way that it requires specific hardware on both the client side and the infrastructure. This hardware is also only for very specific solutions. Our solutions can work with any kind off access point which is compatible to run the specific software. The absence of the need for very specific hardware can also reduce the cost of the solution.
For future development of the project, there are a number of things which can be improved upon:

- Implement the reservation of the Wi-Fi channel if the network is unreliable
- Implement a modulation and coding scheme if the network is deemed reliable at first. To try and further combat packet loss
- Amplify the output signal of the Openwifi board and test it in a real world setting
- Implement parts of the software in Linux kernel modules for more efficient execution
- Find a way to synchronise timing across the devices for more accurate time measurements for probing
- Use a flag register in the FPGA to signal that the board is in emergency mode and implement parts of the protocol on driver level
- Use network simulators to get results on the theoretical working of the protocol

The reservation of the Wi-Fi channel is the software which is responsible for taking the correct action when an emergency message is received. An application layer framework is implemented to send this emergency message. The software to take the correct action is device specific. For this reason, we did not implement this. For our project, we planned to implement a modified modulating and coding scheme. When the network is not congested, a modulation with a lower data and coding rate to enhance the chance of a successful transmission of the emergency messages. Lowering those rates, decreases the chance for a packet to be dropped and the chance of a bit error. We did not find the time to implement this. Most of the implementation of our project is programmed in Python. For a better performance of the protocol, all of this could be implemented in modules for Linux and in the driver. By modifying the driver and adding a register, higher layers would able to turn CTS-to-self on or off. Now a new driver must be build to make a change to CTS-to-self.

4 Project Management

4.1 Schedule

Our schedule is represented with the help of a Gantt-chart. It can be found at the end of the report. Of course, a main influence on the project progress and our working method was the international outbreak of the Corona-virus. The university switched to a full online education method and eventually the government obligated people to stay quarantined. The country went into a lock-down. Because of the great impact of this event, we will discuss or working methodology in the periods before and after the lock-down separately. Next to this, our team found some struggle by the lack of participation of our team member, Leander. From the beginning, communication and participation of this member was poor. We discussed this problem several times and the struggle of Leander was that he found it very unpractical that he was not assigned specific tasks to do. This however, was true for every team member and is inherent to the learning process of a project. It is far less defined than an educational course and thus asks for a different approach of studying. Nevertheless, Leander eventually dropped the course during the Easter break and thus left the group.

In the first weeks, we planned two dedicated, physical appointments every week. Next to this we saw each other almost daily, during the lectures of other courses. We also had an online communication environment on Slack and Discord where we could also contact our advisors. In each of these dedicated appointments we had a short checkup with our advisor Xianjun Jiao. After these first weeks of research, we chose our research question. It is at this point in time that the university switched to an online education method and physical gatherings were prohibited for the rest of the semester. We decided to maintain the two, weekly appointments but instead of physical meetings, we held online meetings on the communication platform Discord.

With the research question determined, each group member received a subject for research which he would then explain to the group a week later. After this, a basic flow chart of the invented protocol and the main framework was designed. We then split this framework into smaller parts that each group member had to implement individually. During the time following this task division, Leander did not show up anymore on the 'online' meetings. This caused that the team no longer had sight on the progress of his individual work. After several attempts to contact him, he communicated that he had dropped the course. In the week following his final decision, the further planning of the project was made. We had allocated one week to design test cases, one week to perform the test cases and in the last week we planned to make the report. Although this planning was made, we did not managed to pursue it. During the period were normally everything should have been tested, we were still trying to 'paste' the individual implementations together. Eventually we decided to create test cases for the different parts of the protocol separately. In the end we did not had time left to handle the bugs that were exposed by the testing. In the original planning there would have been time left to solve these bugs.

4.2 Evaluation

In a meeting in the last week, we reflected on the progress of the project and discussed the deviation from the planning. Several approaches became clear that would have benefited the progress. First of all, our team lacked a sort of team structure. We did not have a clear team leader that took the initiative when the schedule came in danger. A solution would have been that each team member would have been team leader for a certain period of time. The role of the team leader would have been to plan the structure and lead the meetings. Which aspects of the projects should be discussed at the meeting et cetera. Especially during a quarantine where everybody is at home by themselves, an assured initiative by a team leader would have been very beneficial. A second malfunction was that we never really set any marked deadlines for ourselves. This caused us to deviate from the planning far too easy. A solution to this would have been to create the Gantt-chart earlier on in the project. The team leader should have then used this as a guide. A last pitfall, was the fact that we divided everything in to smaller, personal tasks. Although this assured an efficient work progress, it caused us to lose insights into each others implementations. This delayed the cooperation to implement the individual work into the main framework. The unstructured meetings did also not cover up this division. Again a solution would be a team member that had been appointed the task to lead the meetings.

5 Acknowledgement

We would like to thank our advisors prof. dr. Xianjun Xiao and dr. Wei Liu and supervisor prof. dr. ir. Ingrid Moerman for being patient and teaching us about Openwifi while giving us the freedom to chose our own research project. The very in depth knowledge of prof. dr. Xianjun Xiao about wireless communication protocols and his experience in the field, were essential to this bachelor thesis. We thank you for the opportunity to work with this project.

Bibliography

[1] prof. dr. ir.Ingrid Moerman. Moban: chapter 5 wireless lan. Personal contact, 2018-2019.

[2] N. Shenoy, J. Hamilton, A. Kwasinski, and K. Xiong. An improved ieee 802.11 csma/ca medium access mechanism through the introduction of random short delays. In *2015 13th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 331–338, 2015.

[3] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach, Seventh Edition.* Pearson Education, 2017.

[4] What's the difference between 2.4ghz and 5ghz wifi. https://www.centurylink.com/home/help/internet/wireless/which frequency-should-you-use.html.

[5] Kari J. Nurmela and Patric R. J. Ostergard. covering a square with up to 30 equal circles. *Teknillisen korkeakoulun tieojenka sittelyteorian laboratorion tutkimusraportti.(finish)[research rapport of the computer science department of the technical university of Helsinki]*, 62, 1905.

[6] S. Bansal, R. Shorey, and A. A. Kherani. Performance of tcp and udp protocols in multi-hop multi-rate wireless networks. In *2004 IEEE Wireless Communications and Networking Conference (IEEE Cat. No.04TH8733)*, volume 1, pages 231–236 Vol.1, 2004.

[7] L. Hassnawi, R.Badlishah Ahmad, Abid Yahya, Mohamed Elshaikh, Ali Alrawi, and Zaid Ali. Performance analysis of motorway surveillance system based on wireless ad hoc camera network (wahcn). *Journal of Communications and Information Sciences*, 2:59, 04 2012.

[8] Alessandro Rubini Jonathan Corbet and Greg Kroah-Hartman. *Linux Device Drivers.* O'REILLY, 2005.

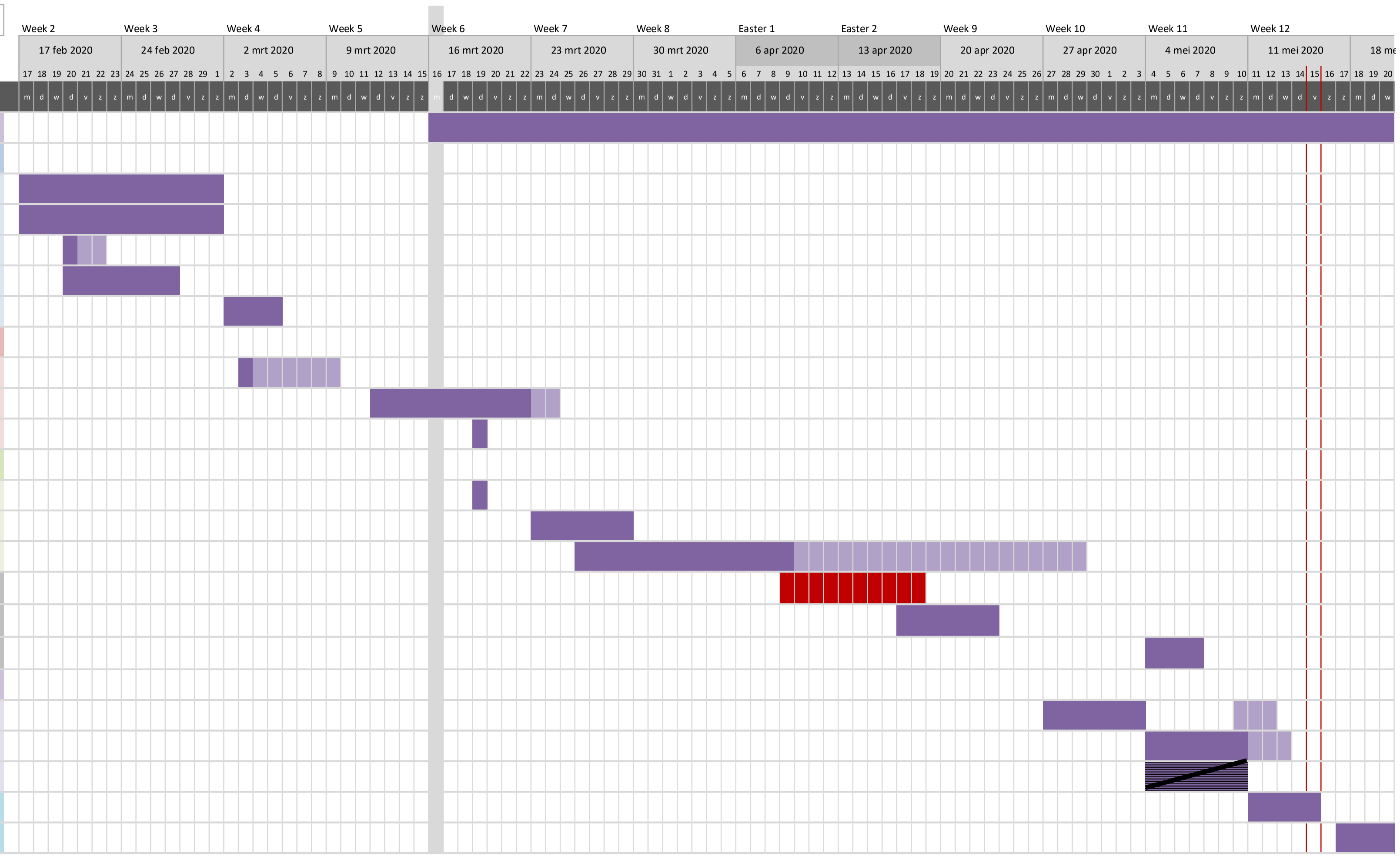
[9] dr. ir. Michael Mehari prof. dr. ir Ingrid Moerman prof. dr. Xianjun Jiao, dr. Wei Liu. open-sdr/openwifi. https://github.com/open-sdr/openwifi, 2020.

Gantt-chart: CCP OpenWifi

ma, 17/2/2020
1

Legend:
1: planned 2: delayed 3: unforseen

TAAK	START	EINDE
Quarantaine Corona	16/03/20	20/05/20
Orientation in OpenWifi project		
IEEE802.11 protocol	17/02/20	1/03/20
Github-repository, general code analysis	17/02/20	1/03/20
Instalment software environment	20/02/20	20/02/20
Hands-on-exercise: driver adjustments	20/02/20	27/02/20
Presentation 1	2/03/20	5/03/20
Research question: Emergency communication		
Determination problem statement	3/03/20	3/03/20
Individual research on different subjects	12/03/20	22/03/20
First draft protocol flow chart	19/03/20	19/03/20
Implementation protocol high level (Python)		
Main framework design	19/03/20	19/03/20
Main framework implementation	23/03/20	29/03/20
Individual code implementation	26/03/20	9/04/20
Difficult communication with Leander and his eventual decision to quit the project		
Presentation 2	17/04/20	23/04/20
Presentation online job fair	4/05/20	7/05/20
Hardware implementation		
Creating test-cases and environment	27/04/20	3/05/20
Gathering data form simulation	4/05/20	10/05/20
Implementation of MCS (modulation & coding s	4/05/20	10/05/20
Report	11/05/20	15/05/20
Online Final presentation	17/05/20	20/05/20


Week 2
17 feb 2020
Week 3
24 feb 2020
Week 4
2 mrt 2020
Week 5
9 mrt 2020
Week 6
16 mrt 2020
Week 7
23 mrt 2020
Week 8
30 mrt 2020
Easter 1
6 apr 2020
Easter 2
13 apr 2020
Week 9
20 apr 2020
Week 10
27 apr 2020
Week 11
4 mei 2020
Week 12
11 mei 2020
18 me