TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Elektrotechnik und Informationstechnik Institut für Nachrichtentechnik

Deutsche Telekom Lehrstuhl für Kommunikationsnetze

Diploma Thesis

REAL-TIME WIRELESS GUIDANCE SYSTEM FOR CYCLISTS

Jonas Bechtel
Born on: Xth YYY ZZZZ in W-City
Matriculation year: 2011

to achieve the academic degree

Diplomingenieur (Dipl.-Ing.)

Supervisor
M.Sc. Máté Tömösközi

Supervisor
Dipl.-Ing. Ievgenii Tsokalo

Supervising professor
Prof. Dr.-Ing. Dr. h.c. Frank H. P. Fitzek

Submitted on: 5th June 2018

Jonas Bechtel
Real-Time Wireless Guidance System for Cyclists

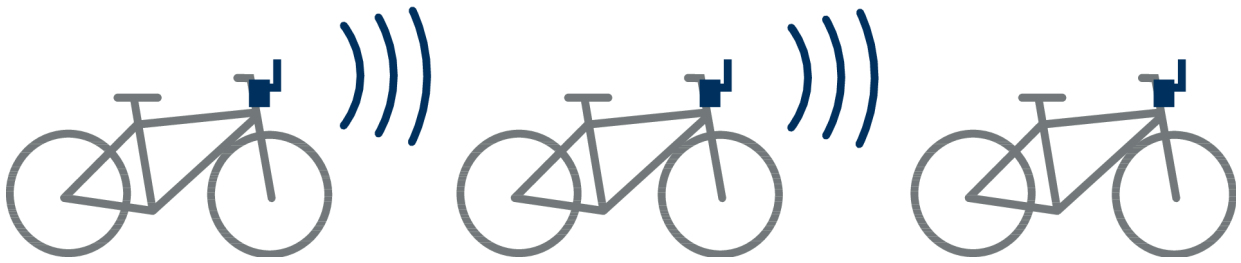
Diploma Thesis, Technische Universität Dresden

Fakultät Elektrotechnik und Informationstechnik

LibreOffice Writer found following parallel thesis project version:

```
0.501--License--Juni_2018
```

This thesis is destined to be uploaded to http://jbechtel.de/site/Tools/R-ConvSpeak/



TECHNISCHE UNIVERSITÄT DRESDEN
FAKULTÄT ELEKTROTECHNIK UND INFORMATIONSTECHNIK

Task for Diplom thesis

For Mr Jonas Bechtel IT/2010

Real-time Wireless Guidance System for Cyclists

Task description

The groups of cyclists often need a communication platform for exchange of the guidance information. The example of such groups are the participants of "Critical Mass" events. With this work, Mr. Bechtel should design the communication platform. First, he makes the feasibility analysis based on the required scenario profiles using available communication solutions on the market. The scenario profile can be constructed with regard to the network architecture, mobility, traffic demands, etc.

The developed platform should be able to handle multiple nodes at various distances and adapt to changes of the network topology in real-time. The scenario profiles will consider both rural open areas and dense urban environments. The impact of various weather conditions has to be included as well.

The student can propose one or several solutions of the communication platform. The key goal is the development of a set of measures that can quantify the performance of the examined solutions. For this purpose, different criteria like minimum data rate, resilience, etc., can be considered. The performance analysis can be based both on the simulation and test field deployment.

Supervisors: Máté Tömösközi and Dipl-Ing. Ievgenii Tsokalo

Reviewer: Prof. Dr.-Ing. Dr. h.c. Frank H. P. Fitzek

Second Reviewer: Dr.-Ing. Roland Schingnitz

Started at: 12.12.2017

To be submitted by: 22.05.2018

The diplom thesis is written in English.

Prof. Dr.-Ing. Steffen Bernet
Vorsitzender des Prüfungsausschusses

Prof. Dr.-Ing. Dr. h.c. Fitzek
Verantwortlicher Hochschullehrer

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts habe ich Unterstützungsleistungen von folgenden Personen erhalten:

Weitere Personen waren an der geistigen Herstellung der vorliegenden Arbeit nicht beteiligt. Mir ist bekannt, dass die Nichteinhaltung dieser Erklärung zum nachträglichen Entzug des Diplomabschlusses führen kann.

Dresden, den

Unterschrift

Zusammenfassung

Zurzeit werden Fahrräder sehr beliebt und viele sehr unterschiedliche Ferien-/Abend-Touren, die man auch “Critical Mass” nenne kann, entstehen. Daran können mehrere hundert Personen teilnehmen. Falls die Tour offiziell organisiert ist, muss über den ganzen Verband kommuniziert werden.

In dieser Diplomarbeit wird eine Kommunikationsplattform entworfen und in einem neu geschriebenen Simulator implementiert. Der Simulator bildet als Hardware-Modell das RFM69-Transceiver-Modul auf SPI-unterstützenden Systemen nach, genau gesagt BeagleBone Green und Arduino Uno. Arduino Uno erweist sich als fehleranfällig und einschränkend, wogegen BeagleBone Green möglicherweise die entworfene Plattform mit geringfügigen Adaptionen ausführen könnte.

Der Simulator nutzt eine Landkarte mit städtischen und ländlichen Bereichen, auf der eine Route markiert wird. Die Tour bewegt sich entlang dieser Route und der Simulator berechnet den Funkpfad zwischen fünfen der Fahrräder anhand eines flachen Modells. Währenddessen führt der Simulator die implementierte Software der Kommunikationsplattform aus, welche vorgeladene Sprachdaten im Adaptive-Multi-Rate codec AMR) sendet, und zeichnet die Empfangslogs der fünf Stationen auf.

Die Aufzeichnungen werden hinsichtlich zweier Routing-Varianten ausgewertet: Ad-hoc-On-Demand-Distance-Vector-Routing (AODV) und Broadcasting mit einem neuen Broadcast-Schleifen-Verhinderungs-Mechanismus. Die Auswertung wird hinsichtlich eines möglichen Benchmarks ausgewertet.

Es stellt sich heraus, dass die Leistung von AODV hinsichtlich von Paketverlusten ähnlich wie die Leistung des Broadcastings ist, aber die Sendeintensität reduziert. Die Sendeintensität wird von einer neuen Methode namens „Registered Radio Reception“ erfasst, welche ein Resultat in der neuen Einheit „ecrm(s)“ ausgibt. Weiterhin wird herausgefunden, dass der Paketverlust und die Registered-Radio-Reception gute Maßstäbe zur Bildung eines Benchmarks sind.

Abstract

As the bicycle is getting very popular many diverse holiday mass cycle tours arise which can also be called “Critical Mass” events. They can count several hundreds of cyclists. In case the tour is an officially organized one the need arises to communicate all over the trail.

In this thesis communication platform is designed and implemented in an simulator which is written from-scratch. As hardware model the simulator aims to emulate RFM69 transceiver module on two systems which support serial peripheral interface (SPI) interface natively, namely BeagleBone Green and Arduino Uno. Arduino Uno is found to be error-prone and limitating whereas BeagleBone Green could potentially run the designed platform with few adaptions.

Simulator uses a map with distinct urban and countryside areas on which a route is marked. The cyclists go along this route and simulator calculates the radio path between five of them based on a flat model while executing the implemented software of the communication platform which sends artificial speech data in Adaptive Multi-Rate codec (AMR) and captures the receiving logs of the five stations.

The captured logs are evaluated with regards to two variants of routing: Ad hoc On-Demand Distance Vector Routing (AODV) [AODV] and simple broadcasting with a new broadcast loop prevention mechanism. The evaluation is evaluated with regard to a possible benchmark.

It is found that AODVs’ performance in terms of packet loss is similar to broadcasting while reducing sending intensity. Sending intensity is captured by a new procedure called “registered radio reception” which outputs a result in new unit “ecrm(s)”. It is further found that packet loss and registered radio reception are good measures to form a benchmark.

Contents

1 Introduction

This thesis contains lots of information on integrating several components to a speech communication system as well as information about how they can be simulated/emulated. The contents that you can discovered here are only a part of the actual work paths that have been taken to fulfil the task given in the task description. So let's start with the task:

Part I is to design a communication platform which allows (some of) the cyclists which you got to know in abstract to talk so that others (specially all other stations in field) can here the spoken word.

Part II defines some requirements on the platform.

Part III is the "benchmark" which covers the design of a set of measures.

Originally the measurements which should feed the benchmark were intended to be made on real hardware. Unfortunately real hardware has been delivered late and didn't mean to work (SD card initialization error on Arduino) which required an intensive search for alternatives which ended in creating an own simulator whose ecosystem can be discovered in Figure 1.1. (First parts of simulator and hardware abstraction layer already began since task description was signed but were by far not usable.) So lots of programming work for Arduino has become obsolete. Not only did this allow an extension of the submission date of this thesis by two week, it also causes the focus of this thesis to lay on implementation topics rather than evaluation topics.

But at least two good measures were found with regards to evaluation so this thesis with it's software provides a complete integrated rush from single sine waves up to packet switching networks and speech data.

I hereby explicitly want to express my thank for the possibility to work on this interesting topic which tangents to one of my interests – the cycling!

Jonas Bechtel

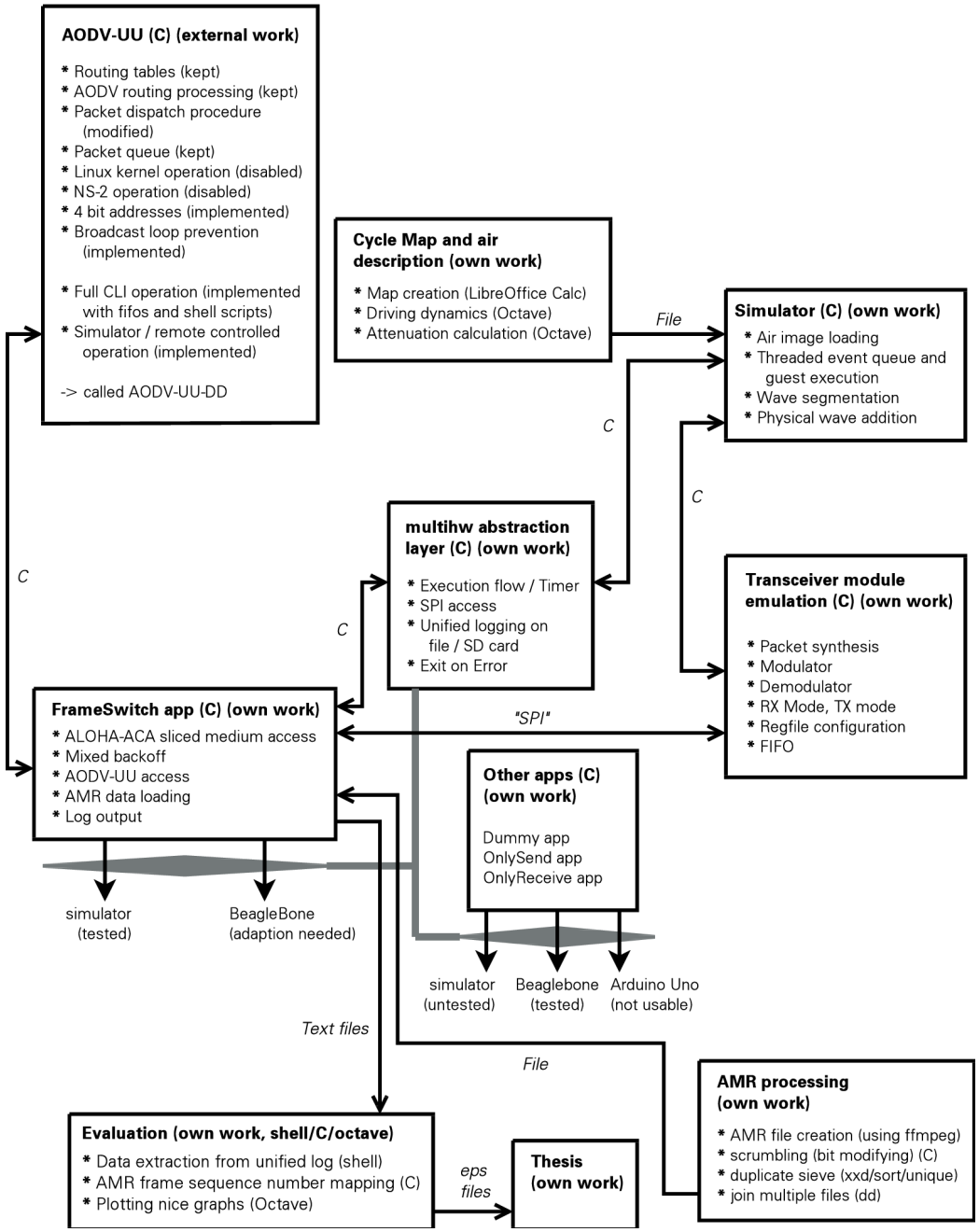


Figure 1.1: Interaction and contribution of all components. AODV-UU implementation is [AODV-UU]

2 Physical Environment

As physical environment we discuss electromagnetic waves as well as the urban/countryside setting with the travel-route of the cyclists.

2.1 Physics of the Air

For this thesis' data transmission schemes the most interesting considerations from the big field of physics are cut out: the electromagnetic waves. Driven by the considerations of Maxwell you can deduce their wave equations (which is $\Delta E = \frac{\epsilon \mu}{c^2} \cdot \frac{\partial^2 E}{\partial t^2}$ for electric field) and from which the field strength formulas can be deduced. For example a homogeneous plane wave which propagates in x direction can be expressed as with an electric field strength of $E_x = 0,\ E_y = A \cdot e^{i\omega \cdot (t - a \cdot x)},\ E_z = 0$ (2.a)

and an magnetic field strength of $H_x = 0,\ H_y = 0,\ H_z = \sqrt{\frac{\epsilon}{\mu}} \cdot A \cdot e^{i\omega \cdot (t - \frac{\sqrt{\epsilon\mu}}{c} \cdot x)} = \sqrt{\frac{\epsilon}{\mu}} \cdot E_y$ (2.b)

[PHYSIK chapter VII]. Interpreting this equations you can see that magnetic vector and electric vector both are always orthogonal to each other and to direction of propagation.

Radio waves which are used in data transmission applications are not plain waves but more complex. It is difficult to find concise closed descriptions on them so much considerations are based on empiric descriptions derived from technical behaviour of those waves. The considerations are described in the subsequent subsections.

Transmission coefficients in different situations

[RPRO, chapter 8.2] and [IRFP] mostly accordingly present some empirical path attenuation models which are widely used and are based on extensive measurements: the Okumura-Hata model, the COST-231/Hata model, the Young model and the Lee model as well as ITU Indoor model and it refers to further models.

When covering the Lee model [RPRO] presents the found parameters for four different cities which are heavily variing. It concludes:

> "The variability in the above empirical parameters for seemingly similar cities emphasizes the importance of empirical determination of L_0 and γ whenever possible."

Unfortunately neither of the existing models can be used for this thesis:

- Okumura-Hata model was introduced for mobile cellular networks. It assumes one of the stations to be located 30 m above ground which is impossible to realize on a bicycle.
- COST-231/Hata model is an extension to frequencies above 1.5 GHz. Which cann't be used for thesis' 433 MHz transmission.
- Young model fails for distance below 1 km [RPM].
- Lee model is applicable for distances greater than 1.6 km while the cyclist (see below) will have at most a distance of 900 m.
- ITU Indoor model cann't be used outdoor
- Referred COST231-Walfish-Ikegami model is applicable for frequencies above 800 MHz.

Same observation is made by [RPM] which developed an own model in light of a sensor network for smart metering applications. Developed model has a data base of 12 markers basically along one street so it has a very small data base compared to the formerly described models.

I found the proposed formula dœsn't work with the tabulated values in the paper so I implemented the formula with exchanged parameters α/β/γ so the result fitted best to the table presented. In effect I used attenuation formula

$$L_{urban} = 35.224\,\mathrm{dB} + 31.635\,\mathrm{dB}\cdot\log_{10}\left(\frac{f}{\mathrm{MHz}}\right) + 46.614\,\mathrm{dB}\cdot\log_{10}\left(\frac{d}{\mathrm{km}}\right). \qquad (2.c)$$

Upon notice the papers' authors about the seemingly non-fitting results in the table of [RPM] sent me a corresponding Excel sheet whose calculations included a summand which in effect makes the measured value smaller. So if you subtract 14.3 from every of the values of the column "Measured" (so it gets even more negative) then results of the proposed formula fit best to the measured values. Also the results of the compared models formulas do fit then.

Unfortunately I didn't find the time to adapt simulation to the new insights (so-to-speak simulator dœsn't use the exact formula from paper) but as parameters are strongly varying and require in-place empiric measurement (see above) this dœsn't impact simulation quality.

For countryside calculations "Flat Earth Direct plus Reflected Model" from [RPRO] was used which is rather theoretical. It has following formula:

$$L_{\mathrm{p}}^{\mathrm{flat}} = 120\,\mathrm{dB} + 40\,\mathrm{dB}\cdot\log_{10}\left(\frac{d}{\mathrm{km}}\right) - 20\log_{10}\left(\frac{h_1}{\mathrm{m}}\right) - 20\log_{10}\left(\frac{h_2}{\mathrm{m}}\right) \qquad (2.d)$$

[LANDUSE] covers wave propagation much more detailed (calculates impulse responses in forest, next to forest, etc. whilst regarding the reflections beneath the leafs) but requires lots of assumptions which didn't meet this thesis' main focus so such an detailed approach was dropped.

When setting antenna heights to 1 m and frequency to 433.92 MHz which is in the center of the used ISM band (433,05 MHz to 434,79 MHz) following formulas were derived with attenuation only dependent on the distance of the stations:

$$L_{urban}\left(\frac{d}{\text{km}}\right) = 118.66\ \text{dB} + 46.614\ \text{dB}\cdot\log_{10}\left(\frac{d}{\text{km}}\right)$$
$$L_{country}\left(\frac{d}{\text{km}}\right) = 120.00\ \text{dB} + 40.000\ \text{dB}\cdot\log_{10}\left(\frac{d}{\text{km}}\right) \quad (2.e)$$

As parts of the bicycle ride have mixed urban – countryside paths (see map below) these formulas have to be combined. In order to keep the log() functions properly working they'll get provided the full distance even if only partially used. So following mixture approach is chosen:

$$L_{mixed}\left(\frac{d_u}{\text{km}}, \frac{d_c}{\text{km}}\right) = \frac{d_u}{d_u+d_c}\cdot L_{urban}\left(\frac{d_u+d_c}{\text{km}}\right) + \frac{d_c}{d_u+d_c}\cdot L_{country}\left(\frac{d_u+d_c}{\text{km}}\right) \quad (2.f)$$

with $d = d_u + d_c$ with d_u = urban fraction of air line and d_c = countryside fraction of air line.

This transforms to the actual in the simulation used path attenuation formula:

$$L_{mixed}\left(\frac{d_u}{\text{km}}, \frac{d_c}{\text{km}}\right) = 120\ \text{dB} + \frac{46.614\cdot d_u + 40\cdot d_c}{d_u+d_c}\text{dB}\cdot\log_{10}\left(\frac{d_u+d_c}{\text{km}}\right) - \frac{d_u}{d_u+d_c}\cdot 1.34\ \text{dB} \quad (2.g)$$

Rain attenuation can be neglected in this thesis as the used frequency is 434 MHz which is below 5 GHz. At frequencies below 5 GHz attenuation by rain is around or less than 0.01 dB/km [RPRO chapter 5.4].

Wave addition – search for a closed solution

We assume the demodulator to handle signals like

$$R = a\cdot\cos(f\cdot t + p) \quad (2.h)$$

with a being the amplitude and f being the frequency. Possibly demodulator "is interested" in phase p (so-to-speak demodulators' electronics change their behaviour when phase changes).

An addition of waves with different amplitudes which results in a formula 2.h cannot be found in literature. Literature usually regards frequency and phase differences of the input signals but assumes equal amplitudes, like the nice animated graphs and explanation in [SUPERWAVES].

So simulator has to follow a custom wave addition procedure. Consider the two signals to be added:

$$S_1 = a_1\cdot\cos(f_{s1}\cdot t + p_1) \text{ and } S_2 = a_2\cdot\cos(f_{s2}\cdot t + p_2) \quad (2.i)$$

With cosine addition [MERZIGER]

$$\cos x + \cos y = 2\cdot\cos\frac{x+y}{2}\cdot\cos\frac{x-y}{2} \quad (2.j)$$

the added signal results in following formula:

$$
\begin{aligned}
R &= a_1 \cdot \cos(f_{s1} \cdot t + p_1) + a_2 \cdot \cos(f_{s2} \cdot t + p_2) && \mid\ p_3 = p_1 - p_2 \\
&= a_1 \cdot \cos(f_{s1} \cdot t + p_3) + a_2 \cdot \cos(f_{s2} \cdot t) && \mid\ a_3 = a_2 - a_1 \\
&= a_3 \cdot \cos(f_{s2} \cdot t) + 2\,a_1 \cdot \cos\left(\frac{f_{s1} \cdot t + p_3 + f_{s2} \cdot t}{2}\right) \cdot \cos\left(\frac{f_{s1} \cdot t + p_3 - f_{s2} \cdot t}{2}\right) && \mid\ p_4 = p_3/2 \\
&= \overbrace{a_3 \cdot \cos(f_{s2} \cdot t)}^{\text{remaining wave}} + \overbrace{2\,a_1 \cdot \underbrace{\cos\left(\frac{f_{s1} + f_{s2}}{2} \cdot t + p_4\right)}_{\text{joint amplitude}} \cdot \underbrace{\cos\left(\frac{f_{s1} - f_{s2}}{2} \cdot t + p_4\right)}_{\text{fading}}}^{\text{joint waves with fading}}
\end{aligned}
\qquad (2.k)
$$

You see two superimposed sine waves in the formula: one belongs to the individual wave with original frequency and one joint wave which is fading. The joint waves' frequency is centered between the two input frequencies (see "joint amplitude" in formula). Joint wave is a fading wave with new amplitude $2*a_1$ while the remaining wave has an amplitude of $a_3 = a_2 - a_1$. W.l.o.g we assume that $a_2 >= a_1$; Otherwise you have to exchange the two input signals before using the formula.

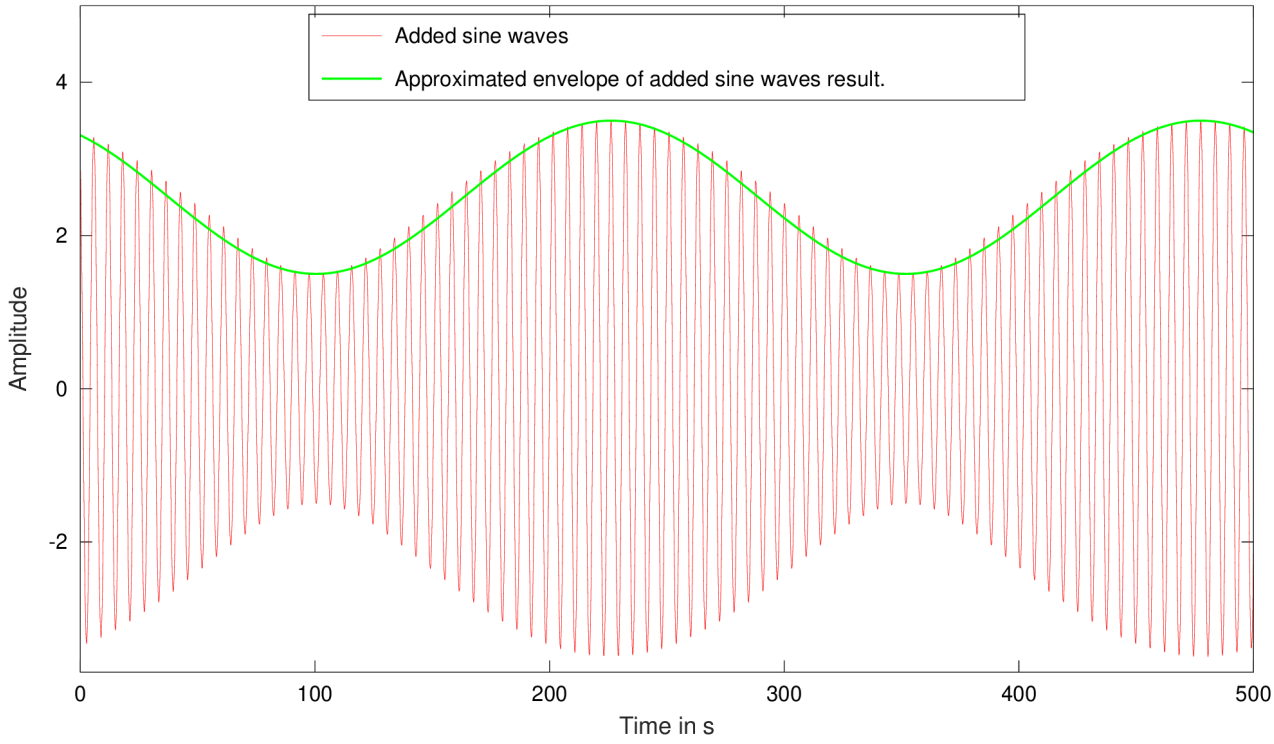


Figure 2.1: Result of the addition of $S_1(t) = 2.5 \cdot \cos(1.025\,\text{Hz} * t + 0.2 \cdot \pi)$ *and* $S_2(t) = 1 \cdot \cos(1.000\,\text{Hz} * t + 0)$ *including the approximated envelope* $E(t) = 2.5 + 1 \cdot \cos(0.025\,\text{Hz} * t + 0.2 \cdot \pi)$.

In Figure 2.1 you see the precise addition result of S_1 and S_2. As you can see the constructed envelope function (a cosine function) dœs not fit the actual envelope off the added waves. At this point you can see that it is not possible to describe the added waves with a function

$$R = \left[a + c \cdot \cos(f_m \cdot t + p_m)\right] \cdot \cos(f \cdot t + p) \qquad (2.l)$$

so it is not possible to construct a solution like 2.h which is a special case of latter formula setting $c = 0$.

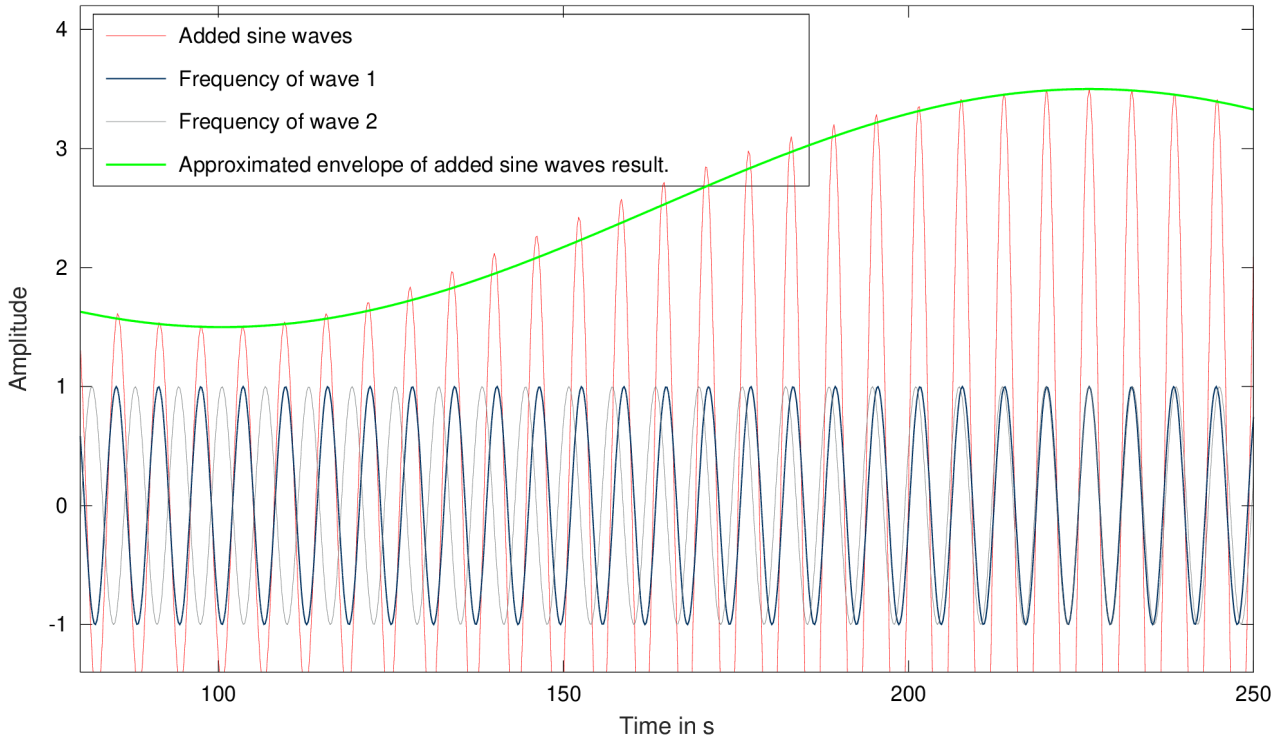


Figure 2.2: Detail of Figure 2.1 with added frequency indicating functions. The frequency of wave 1 is shown by plotting wave 1 with amplitude 1. The frequency of wave 2 is shown by plotting wave 2 with amplitude 1. (So wave 2 is actually plotted as-is while wave 1 is plotted in frequency indicating variant in order to make it comparable to wave 1.)

Another interesting point shows up when you count waves in Figure 2.2: The resulting wave (red function) has exactly the same number of maximums/minimums as the dominant wave 1 (indicated by blue function though with different amplitude). But the extrama and nulls of the red wave are affected by wave 2 so they don't show up in constant intervals. This is another reason why it is impossible to describe the resulting wave in a concise cosine term.

When adding same-amplitude waves with different frequencies the resulting frequency of the joint sine part cannot be determined because in the envelope minimum (which happens at total destructive interference time and is zero then) the resulting wave extrema can't be counted.

Frequency allocation

Nowadays the allocation of frequencies / radio licenses has become crucial to radio applications for several reasons:

- Scarcity of radio spectrum (= no free frequencies) [PLNC]. Many frequency bands have already multiple registered applications [BNETZA].
- Different properties of different frequencies. E. g. in 60 GHz it is technically not more than short range radio possible as one of oxygens' resonance is at 60 GHz and leads to high attenuation rates of this frequency [RPRO ch. 5.3] [METRICS].
- In contrast, low frequency bands are demanded as the waves are said to go through bodies unobstructed [SOS-2] are spare.
- International applications have to work in many countries yet many regions have own regulation [SOS].
- Frequency allocations change over time requiring adapted user settings, different firmware or new hardware. Event managers and microphone producers[1] don't like this [SOS].

Awareness on electromagnetic impact

Another aspect is the raising awareness on possibly negative impacts of electromagnetic radiation on human bodies. While legislators don't change the permitted transmitting power of devices or base stations [WHITE] suggests that (partially) radiation free zones/buildings/vehicles could be installed to protect affected citizen while facilitating their social participation.

Radio applications will be more widely accepted a) by frequency regulators when bands are allocated / relocated and b) by the people who claim to be affected by electromagnetic hypersensitivity when precise information (emission profiles) are available and prospective use is weighed up against the emission profiles.

Open Task: Finding Metrics

The need for metrics, induced by the need of frequencies leads to statements like [METRICS]:

> The Radio Act of 1912 dictated perhaps the first spectrum efficiency requirement when it said that "In all circumstances, except in case of signals or radiograms relating to vessels in distress, all stations shall use the minimum amount of energy necessary to carry out any communication desired. [...] there is still no widely-accepted method for applying generalized spectrum efficiency metrics to specific radio (i.e., wireless) systems and services. Without metrics, it is impossible to demonstrate improvements in spectrum efficiency.

1 You could object that microphone producers' sales raise when frequency allocations get shuffled but seemingly they take sides with their customers. Maybe because these kind of customers cost things out and deny frequent hardware change. Or because they want to concentrate on their core business of "making sound".

> NTIA is committed to ensuring that the government's use of this valuable resource is as efficient and effective as possible. [...]
>
> The literature review found broad consensus on the general form of fundamental spectrum efficiency metrics spectrum efficiency metrics for terrestrial broadcasting, mobile, and fixed services. But the application of generalized spectrum efficiency metrics to specific radio systems and services, each with messy individual idiosyncrasies, is more problematic, and complexity increases when we attempt to apply these metrics to spectrum sharing scenarios and rapid network densification.

So any new and old radio application must stand the review whether it uses bands in efficient manner.

2.2 Devised Map

As thesis has facilities to support the countryside – city dualism. In order to allow simulations which include the {countryside → city} and {city → countryside} transitions a sample map is created. As data representation same approach as in [LANDUSE] is taken and used on a assumed flat (2D) map: the map consists of squares with one land usage assignment per square. Size of squares is chosen 50 m like in the paper (where one square is called one "pixel").

Map has been created on a usual spreadsheet program (LibreOffice Calc) with 54 lines and 510 columns on which some urban and some countryside areas (freely chosen) were placed (each cell has been given a number which represents a usage, so "50" stands for countryside and "52" stands for urban areas). Then route was put on the map. For the sake of simplicity route is also given in square granularity so for defining "cyclists will cross this cell" you have to enter "82" instead of "52" on urban squares and "80" instead of "82" on countryside squares. Actual rout than was inside a window of 35 x 423 squares with an air distance of $\sqrt{(16 \cdot 50\,\mathrm{m})^2 + (422 * 50\,\mathrm{m})^2} = 422.3 \cdot 50\,\mathrm{m} = 21{,}1\,\mathrm{km}$ between starting square and destination square. Route does not go straight forward so actual route distance is 25.4 km. Final map and route is shown in Figures 2.4 and 2.5.

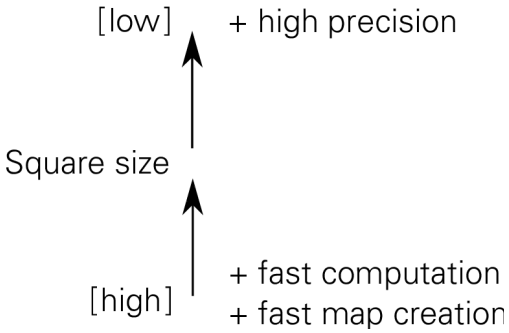


Figure 2.3: Trade-off graph for choosing the map granularity.

Numbers 35 and 423 seem to be small but it takes some time to fill in $35 \cdot 423 = 14{,}805$ cells manually. So the idea you can get of the trade-off graph in Fig. 2.8 that a fine granularity (low sized square) map allows high precision calculations and so leads to better scientific results may fail in cases when you create less maps because personal "creative" time runs out. This situation could be improved by attaching to existing geographic information system (GIS) or creating a full custom tool for map generation.

Figure 2.4: Screenshot of LibreOffice Calc which has loaded map data. Countryside is coloured green, urban area is coloured red and the fine white meandering line is the actual bicycle route. The sharp-cut white "artefacts" show the assumed reflection line of some high building.

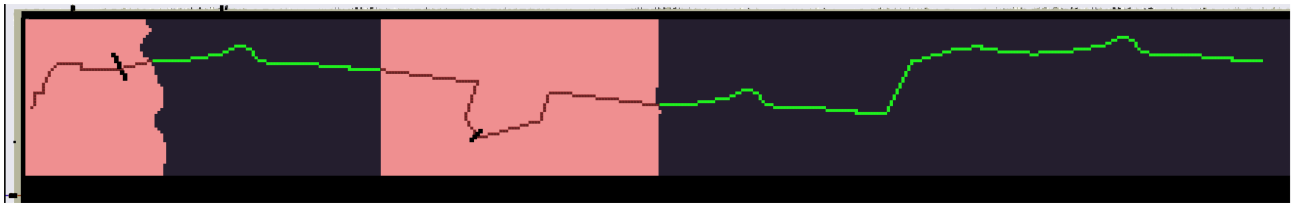


Figure 2.5: Same as Fig. 2.4 but with some photo processing so that the tour (green and brown line) is clearly visible.

2.3 Convoy of Bicycles

This thesis is about a convoy of bicycles of which some are equipped with the actual radio hardware. These equipped bicycles swim somewhere in the convoy and it is important to find a model where they are in order to simulate the radio links between them.

Such a model is not commonly discussed in literature. Main task of research is to determine the capacity and flow of moving cyclists on some fixed infrastructure [DYNAMICS, BURGERS] or to compare vehicular dynamic models to bicycle dynamic models [IDM]. There's no relationship between the cyclists assumed and no packing state with luggage. The models do not cover "fixed" cyclist on a "moving" infrastructure. Specially tracking individual cyclists is not done which is crucial for this thesis' task as the radio hardware equips particular cyclists.

Dynamics Model

So an own model has to be used.

[IDM] refers to an experiment which shows a typical speed of 4.2 m/s. As the situation on the convoy is more exhaustive we use a overall speed of the tour of

$$v_m = 3.5\,\frac{\mathrm{m}}{\mathrm{s}} = 12.6\,\frac{\mathrm{km}}{\mathrm{h}} \qquad (2.\mathrm{m})$$

as the overall speed of the tour which is besides also found as maximum speed of single-lane cycle ways in [BURGERS].

It is not taken into account that the speed changes. This would also require a variable W-O distance as density increases with lower speed which is shown by [IDM]. Also not in calculations are breaks for traffic lights and drinking/toilet. Also not in calculations are accidents which potentially lead to gaps in convoy.

We assume a 900 m queue of cyclists of which five are equipped with a station of designed radio system: "W", "B", "C", "D" and "O". For an overview which station is where at what time see Figure 2.6

Cyclist with station "W" is the first which always makes 3.5 m/s (see 2.m). Station W is point of reference for all station, so point of reference moves every second.

Station "O" is always the last cyclist instinctively keeping an accurate distance of 900 m to station W.

Station "B" is around 300 m behind W. It begins with 275 m and falls back to 325 m in 150 seconds. Then it gœs to 275 m in 100 seconds and the loop begins again.

Station "C" is "corking" ([2]) an then going to front (= 30 m behind W) again. This cycle begins with going 150 meters with the trail, then a modulated crossing is there where cyclist of station C starts corking (then his/her velocity v = 0). When the end of the trail has come (~ 45 m in front of O) C, which is sportive an make 2 m/s more than the others, proceeds to the front again in 412.5 seconds (see 2.n).

2 While the number of cyclist who have joint the tour is high they will occupy a crossing for several minutes which sometimes annoys people who want to get/drive to the other side of the street. For safety reasons it is strongly recommendable to explain to those people that they can't do so. This is task of the people who do "cork".

$$t_{back\_to\_front} = \frac{900\,\mathrm{m} - 30\,\mathrm{m} - 45\,\mathrm{m}}{2\frac{\mathrm{m}}{\mathrm{s}}} = \frac{825\,\mathrm{m}}{2\frac{\mathrm{m}}{\mathrm{s}}} = 412.5\,\mathrm{s} \qquad (2.n)$$

On countryside C is always 40 m behind W, but after having proceeded 2000 m there will be once another crossing where C will begin to cork.

For D, there are two policies implemented. In simple policy D always keeps between 600 m and 650 m behind W in the same way B dœs. The actual used policy is slightly more sophisticated: In urban region, D corks 38 seconds after C corks (so-to speak next crossing is about 110 m after Cs' crossing). After reaching the end, D proceeds to the front (= 30 m) in 580 seconds (velocity increased by 1.5 m/s). If C is not present when D arrives in front, D will cork after 19 seconds. On countryside D always remains between 600 m and 650 m, cycling in the same way B dœs (see Figure 2.7).

Remark: despite of the fact that in real world the positions continuously change and that the ride format can carry sub-second accuracy timings the main assumption for the dynamics is that every cyclists stands still for one second and then jumps to next seconds' position in an instant.

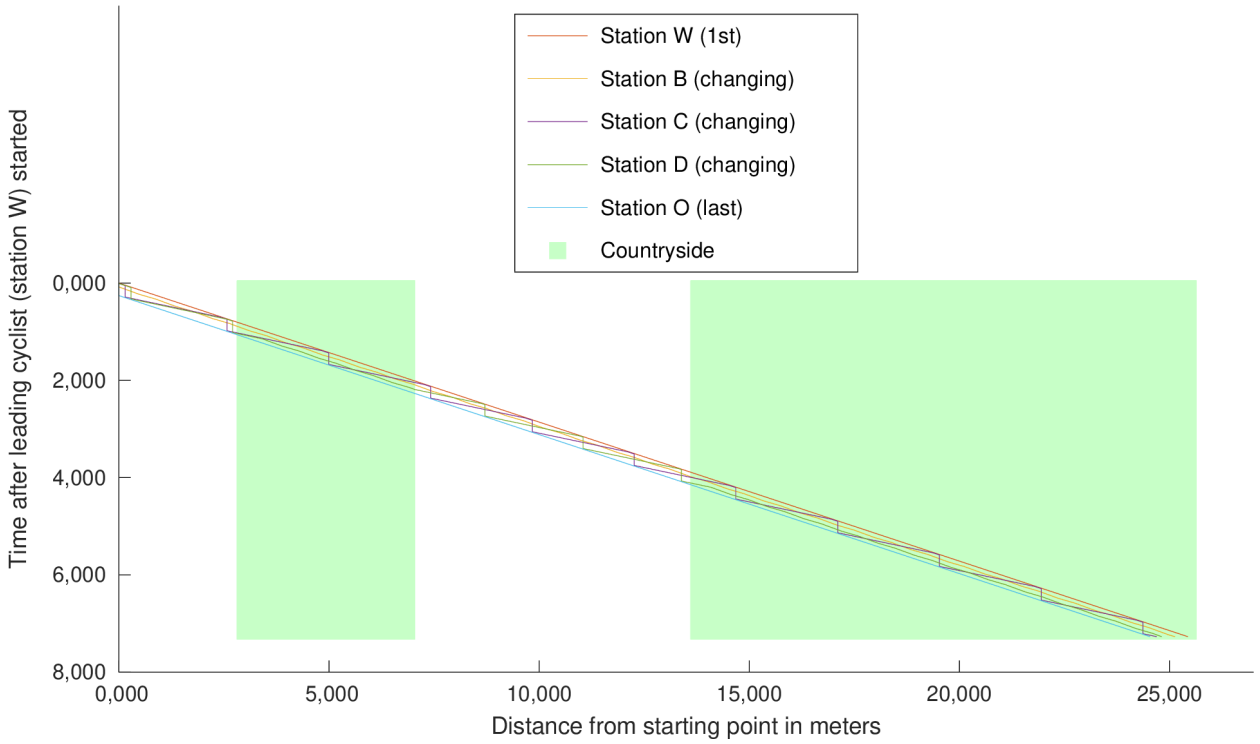


Figure 2.6: Overview of the radio hardware equipped cyclists on all 25 km of tour. Station W is the first so at any given time (y axis) it is on the most right position (x axis). Time is counted from top to bottom of y axis like the train timetables are internally handled at Deutsche Bahn.

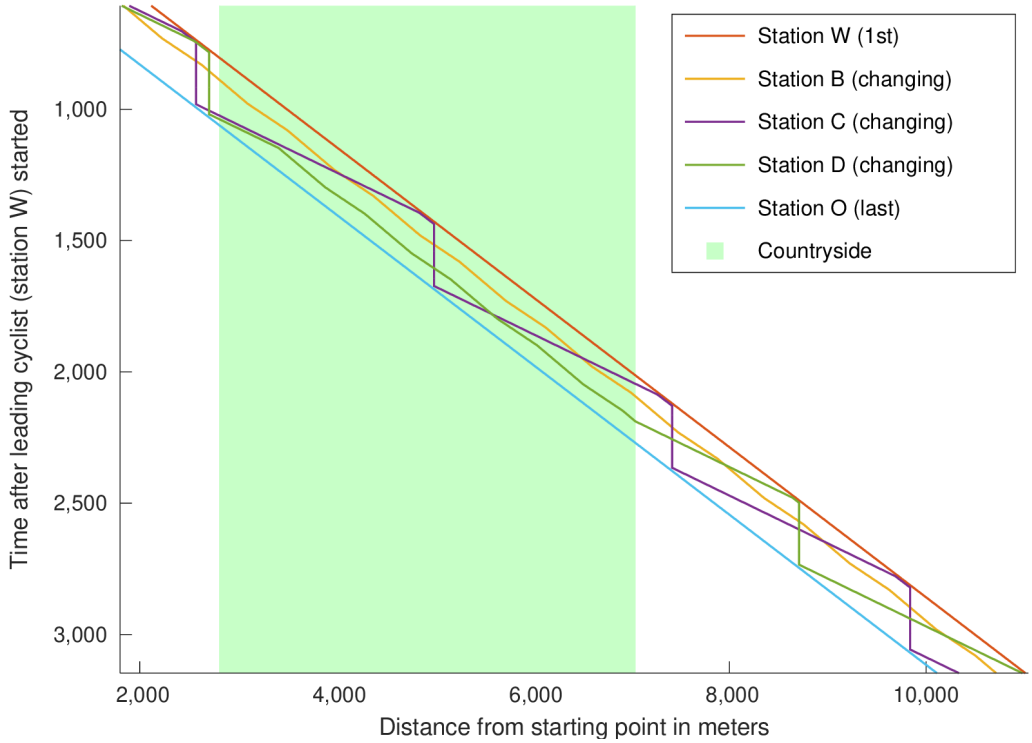


Figure 2.7: Detail of Figure 2.6.

The interaction between the cyclists can also shown with reference station W as shown in Figure 2.8. However it is not possible to mark the urban / countryside areas there other than by line width.

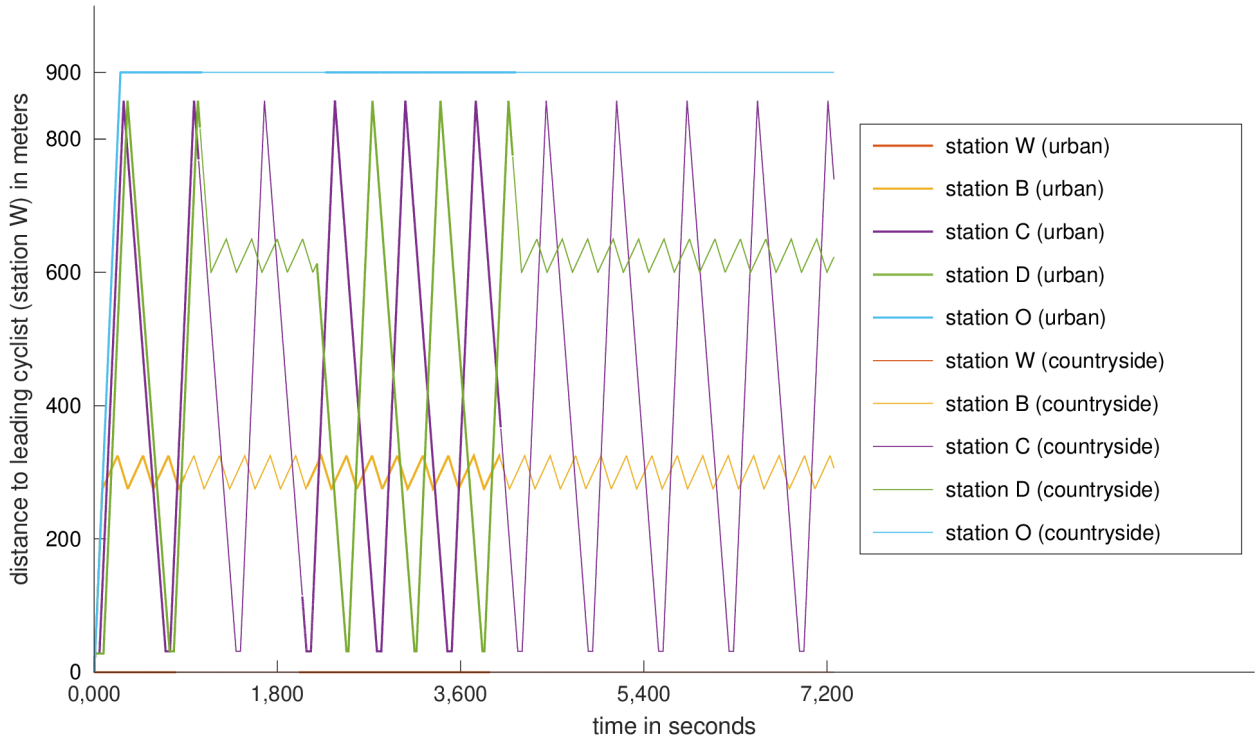


Figure 2.8: Positions of stations with reference to station W.

3 Networking Technologies

This chapter explains how a network gets evolved. Theoretical concepts are discussed as well as practical implementations.

3.1 Steps to Build a Network

It takes several technical steps for radio stations to build up a packet switching network amongst them. A quick overview over building up a network gives Fig. 3.1 - 3.3; another perspective can be found in Fig. 3.4 - 3.6. Steps are also described here:

The first step is to ensure that a communication between two devices works from an electrical point of view. This includes using the same frequencies, modulation shemes and common bit synchronizing conventions. This first step is accomplished by hardware, concretely by radio freqeuency (RF) modules.

The second step is to establish a structure which generally allows multiple neighbours per station. The techniques for doing this are called *DMA with DMA standing for „division multiple access“ and * standing for T as time, F as frequency, C as code and S as space. Though today the second step can easily be performed on general purpose computers the actual time/frequency/code/space selection can also be managed by integrated circuits designed for exactly this purpose.

The third step combines the bilateral links (neighbour-neighbour) to a big network by establishing packet switching methods (stream switching methods in analog domain). This is done by „routing algorithms“ on general purpose computers or on dedicatedly designed integrated circuits in the big Internet Protocol (IP) nodes.

Normally the „steps“ are called layers with naming „Physical Layer“ (L1) = first step, „Data Link Layer“ (L2) and „Network Layer“ (L3) = third step. I use a different naming in order to stress the relationship between the level of hardware specialization and network construction.

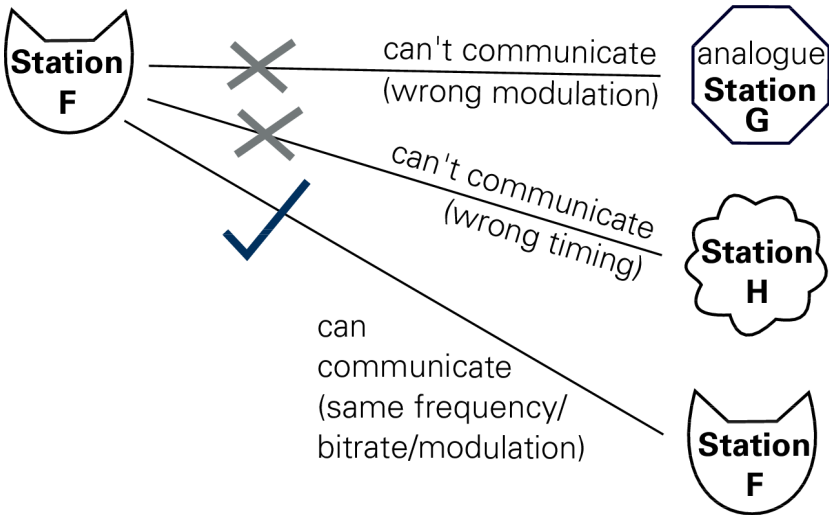


Figure 3.1: Step 1 (top view). Stations understand their neighbours (with exception of neighbours G and H which use different electric parameters)

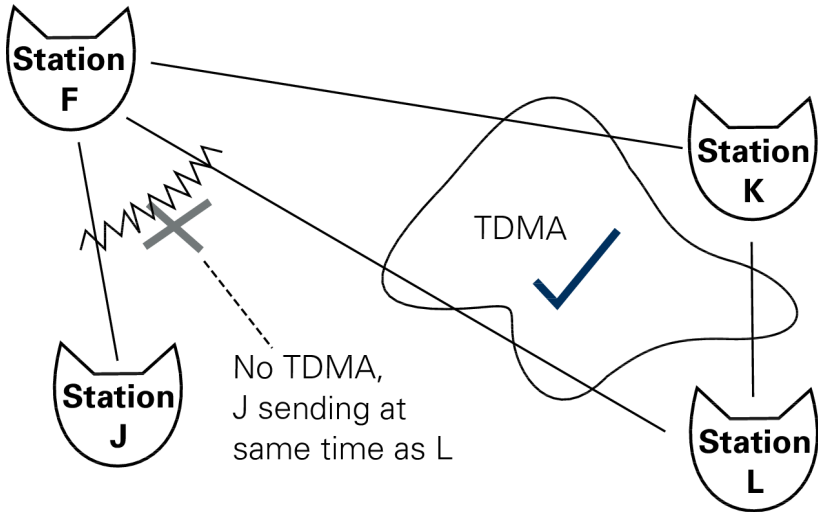


Figure 3.2: Step 2 (top view). Stations understand their neighbours and share the resources (with exception of station J which intercepts the resource sharing)

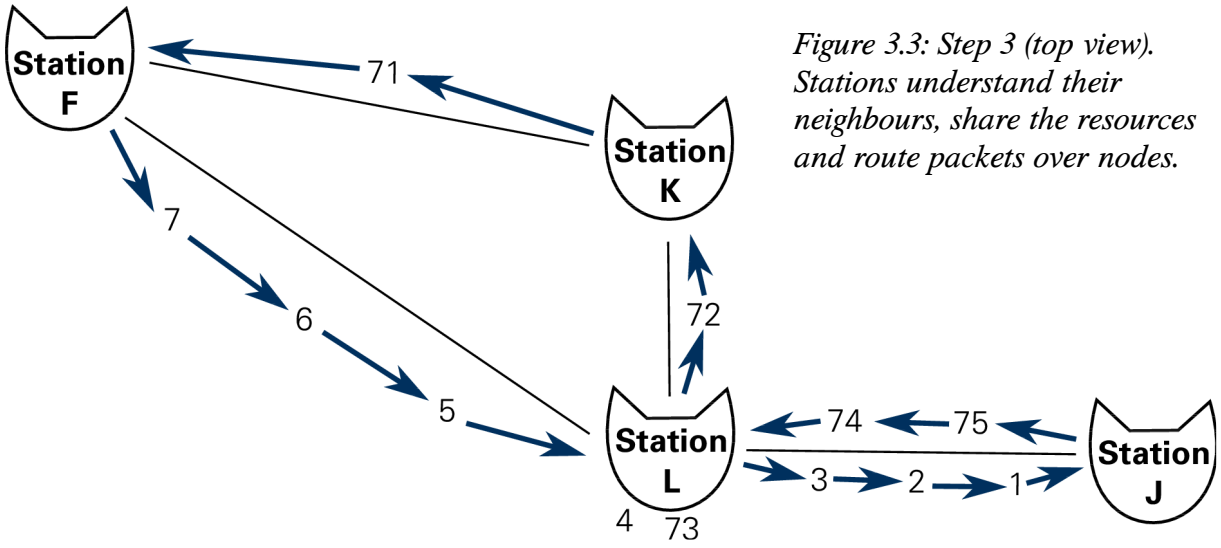


Figure 3.3: Step 3 (top view). Stations understand their neighbours, share the resources and route packets over nodes.

As we humans usually look at the technics in a manner that it works and dæs some stuff we easily forget how technics' view is. In order to take a different perspective let's change the view on the things.

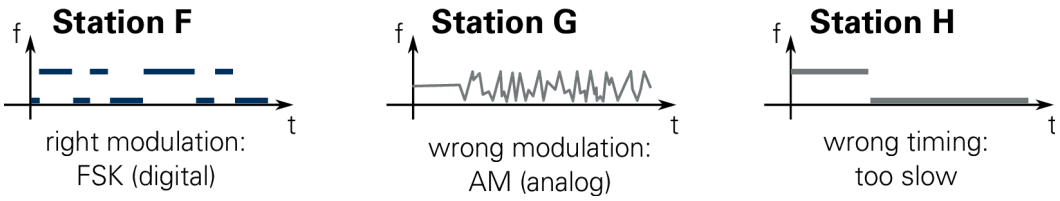


Figure 3.4: Step 1 from stations' point of view. It "sees" the raw signals coming from the other stations

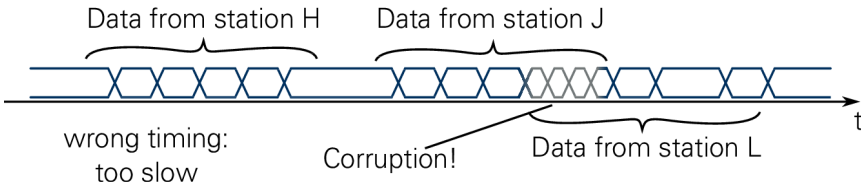


Figure 3.5: Step 2 from stations' point of view. It "sees" data coming in but the reception times from station J and station L overlap so some (usually unusable) mixture appears at demodulator.

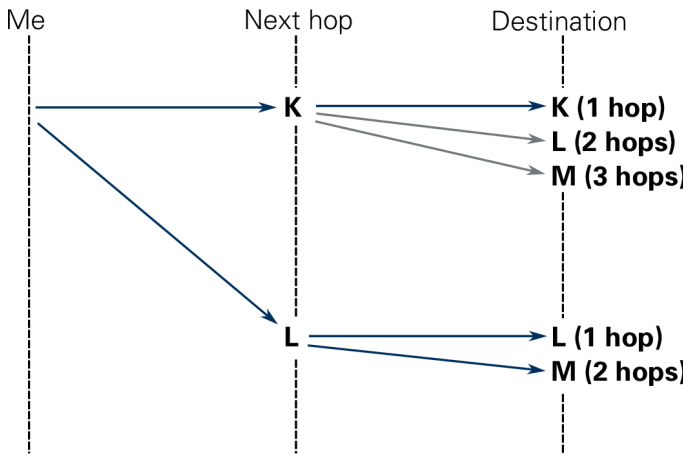


Figure 3.6: Step 1 from stations' point of view. It "sees" the raw signals coming from the other stations.

3.2 Wireless Transmissions

Radio Transceiver Modules

There are different Radio Transceiver modules with different levels of integration. Whereas low level modules only provide the first step of networking (level 1) (RFM 69) there are higher level modules which provide link connectivity information (level 2) (e. g. NeoCortec NC2400 [NC2400]).

The RFM69 Transceiver Module

Out of the different modules the RFM69 transceiver [RFM69] can be considered a low level transceiver. It has about 60 register bytes providing fine grained control over the transmission parameters, such as preamble length, sync word, bandwidth, gain correction, frequency correction, packet length, mode changes (half-automatic operation possible), checksum calculation and even some encryption.

This detailed control implies that you have to know what to do when operating the module and that it cannot be fully emulated in this thesis‘ code. But the exact description in data-sheet en-eases a precise emulation at least of the implemented parts.

The module provides On-Off-Keying (OOK) and Frequency-Shift-Keying (FSK) modulation and allows bit-rates up to 300,000 bps which is sufficient in terms of requirement analysis and which is, aside the market availability and the fine grained control, the reason for having chosen this module.

Further Information on RFM69 (HCW) module can be found in chapter 4.1 Requirement Analysis.

3.3 Multiple Access Schemes

As shortly described on page 14 the second step includes multiple access schemes which are described in this section. It is possible to combine these schemes or to use one alone.

Frequency division multiple access (FDMA) means that stations change to frequencies which are free so that they communicate while knowin/signalling that other stations can use neighboured bands of the same (probably regulated) frequency band/allocation. Traditional use cases of FDMA are analog speech transmission (“Sprechfunk” in German) and AM/FM radio. An extreme case of FDMA is the new technique of ultra wide band (UWB) where devices check a very broad frequency range for free frequencies.

Code division multiple access (CDMA) is a specialized way of sharing transmission ressources. It works even with multiple senders at the same time at the same frequency. The

mathematic key is to generate long convolution codes. The bit synchronized receiver input phrase is convoluted with the original widely known phrases. When the correlation result is high the algorithm knows the presence of the sending station as well as the exact time it was sending. This works even for several stations in parallel. Downside of CDME is the low datarate which is caused by the redundancy of the codes. CDMA is used in satellite navigation systems.

Space division multiple accesss (SDMA) works by installing parallel wires which don‘t interact with each other so each wire can provide an full channel. In radio technique different „spaces“ can be created by directional radio transmissions (German: „Richtfunk“) or by transmitting at locations so distant that one pair/network dœsn‘t get interfered by the other pairs/networks.

Time division multiple access (TDMA) has been widespread since the digital data processing has become cheap and easy. Main idea is to retain the outbound data till the own sending time has come. Then it is sent out fastly leaving time for the other stations. TDMA is used on many different media and topologies, amongst others wired telephony networks (plesiochronous/synchronous digital hierarchy: PDH/SDH), Ethernet connections, most digital wireless radio (WLAN/Wifi, mobile phone communication) and on-board commuication (SPI, I2C).

3.4 Routing from Top and Device View

Routing is all about choosing the intermediate nodes between the sender and receiver of a packet such that the overall network performance possibly is optimal (optimal in terms of maximal overall throughput or minimal average latency or minimal buffer storage to be installed in the nodes or guarantees of transmission of certain traffic (emergency signals in factory halls) or other measures).

Whilst having a top view seeing all state at once the optimal routing is a complex mathematical task, it is even more complex to find a simple algorithm for running on the actual stations which don’t have complete information of the networks’ links. Specially in radio networks link state information obsoletes fast due to station movement leading to different kinds of fading thus requiring a frequent update of the link state informations. But in radio networks the air resource is more limited then wired links in terms of access and data rate. So routing radio networks is a big challenge – imagine you cycle through a city whilst roads are vanishing and appearing in front of you, next to you and behind you.

The AODV Protocol

The AODV routing protocol [AODV] is designed to function in just such radio situations. Every station advertises itself with so called Hello packets which make it known to its direct

neighbours. When a packet to a non-direct neighbour is to be sent, a route discovery (RREQ packet) is started.

The route request is answered with an route reply (RREP packet) on the exact reverse path which the RREQ packet had taken.

If an intermediate station fails to find a path or if a link breaks station sends out a route error (RERR instead of RREP).

AODV uses sequence number to keep routing tables clean of outdated entries.

4 Communication Design

In contrast to chapter 5 Simulation Design this chapter covers all considerations needed for designing the actual system.

4.1 Requirement Analysis

This section covers the requirements for the communication platform.

General Conditions for Platform-In-Use

Following list is the general conditions the platform has to fulfil:

- Platform is usable in different weather conditions, e. g. wind, rain. Plattform will be used rather in summer.
- The system is on the move; stations are attached to particular bicycles.
- No stationary technique (base stations) is used to enable border crossing routes and independency of radio coverage outage.
- Station distances among each other do vary over time; stations also can interchange positions (see Chapter 2.3).
- At any time at least one speech channel must be transmitted and must be able to reach every station so guidance information can be given to all cyclists by loudspeakers attached to the stations. It would be nice to be able to use a duplex speech connection.
- Groups are possibly joining and dividing
- There's no master station and no time synchronization.

Data revenue

Different kind and amount of data will be produced by stations and is to be transmitted:

Address and Status Data

Address data is necessary to address the devices. Assuming there are at most 16 stations 4 bit of address data are sufficient.

The actual stations' status and respective positioning information could take further 8 bit per packet so in total address and status data will be about 20 to 30 bit.

Signalling data

The signalling data can be assumed very small. There are about 10 aspects of a signal (= “stop”, “go”, “accident”, ...). These about ten aspects of a signal can be represented with 4 bit.

Speech

Following speech codecs are taken into consideration:

- Adaptive Multi-Rate audio codec (AMR / AMR-NB) taking 4.75 to 12.2 kbit/s [3GPP]
- Adaptive Multi-Rate Wideband audio codec (AMR-WB) taking around 16 kbit/s but provides a 6.6 kbit/s mode for bad connections.
- Full Rate audio codec (GSM.FR / GSM 06.10) takes typically 50 frames of 33 byte per second.
- Half Rate (GSM 06.20) uses 5.6 kbit/s
- New codec: “Codec 2” with 700 bit/s to 3200 bit/s

Data shall be transmitted with few latency and at least with rate of data production. (Hereby a soft real-time demand was made). A partial recording with later playback is not feasible because the use of the devices was too complex then and could be a cause for traffic accidents.

To be able to use all of the named codecs except for GSM.FR a single channel datarate of 6.7 kbit/s is sufficient.

Cœfficients of safety

Following cœfficients of safety are applied:

- 1.7 for overhead (protocol, addresses)
- 1.3 for switching times (in half duplex operation an extensive transmit-receive switching is performed. This takes e. g. 15 to 20 ms with transceiver LMD-400-R and 150 µs for RFM12B)
- 3.0 for retransmissions (as one station has to send the received data to the next stations so same data is in the air for several times)

These cœfficients have to be combined so following value results:

For signalling data with addressable stations up to 34 bit are needed. With switching and retransmission times the required amount of time corresponds to $34\,\text{bit} \cdot 1.3 \cdot 3 = 133\,\text{bit}$. One aspect of signal shall take at most one second to transmit so 133 bit/s is required data rate.

For speech data up to $6.7\frac{\text{kbit}}{\text{s}} \cdot 1.7 \cdot 1.3 \cdot 3 = 44.4\frac{\text{kbit}}{\text{s}}$ is needed.

Assumed Dynamics

We assume there are five stations with one in the beginning of the convoy and on in the end. (See also Chapter 2.3) Convoy over time may face following situations:

Neat situation

In breaks everyone is close to the group. Thus the characteristics are not line-wise but shows circles e. g. so all stations "see" the other stations. Then speech transmission in duplex would be nice.

Wide situation (fast moving)

When moving "fast" (about 14 km/h) group has low density so from beginning to end there's a distance of 1 km.

Very wide situation (accident)

When an accident happens the distance of the exteriors could be even bigger (1.5 km) and the distribution of stations more scattered (maybe one station at 0 m, 300 m, 1200 m, 1400 m, 1500 m each). In this case it would be nice to have at least a half duplex connection.

Very wide situation results in requirement that over a gap of about 900 m a speech channel (44 kbit/s in total) shall be transmitted.

4.2 Hardware selection

To get a running guidance system some hardware must be chosen according to the requirements in last section. For choosing the hardware it has to be compared which is done here:

Abilities of selected transceiver modules

To build up the system a choice of transceiver modules is required. Every equipped cycle needs one transceiver module. Following is a selection of available modules, the data is taken from the respective datasheets:

LMD-400-R

LMD-400-R is a DIP/THT module with metall casing. It has a switching time of about 15 ms and modulates in gaussian frequency shift keying (GFSK) with a pulse width of 200 µs to 20 ms allowing datarates of 2,400 to 4,800 bit/s → this is not sufficient for speech transmission. LMD-400-R has no buffering at all but provides demodulator output directly at output pin / takes modulator input directly from input pin.

RFM12B

RFM12B comes as small board to be soldiered at the edges. It has a transmitter-receiver turnover time of 150 µs and modulates in frequency shift keying(FSK). It can handle data rates of up to 115.2 kbit/s in digital and 256 kbit/s in analogue mode. Minimum required signal input for receiving is -112 dBm when receiving at 1.2 kbit/s. Modulator/demodulator can be either connected directly or accessed via buffer.

RFM69HCW

RFM69HCW [RFM69] comes as small board to be soldiered at the edges (compatible to RFM12B). It's transmitter-receiver turnover time is 385 µs to 5 ms, depending on configured radio parameters. It can communicate over 800 m with 1.2 kbit/s. It requires a minimum received signal of -114 to -112 dBm for 4.8 kbit/s transmissions. RFM69HCW provides switch-able automatic gain control (AGC) and automatic frequency control (AFC) and can be addressed with serial peripheral interface (SPI).

RN2483

RN2483 is a tiny board to be soldiered at the edges. It provides up to 300 kbit/s data rate with FSK modulation but can also communicate in LoRa format. It covers up to 15 km radios in suburban terrain and 5 km in urban terrain. The module contains an small processor.

CC1125

CC1125 is a very tiny integrated circuit in quad flat no leads package (QFN). It supports several modulations, amongst them 2-FSK and on-off-keying (OOK) is a tiny board to be soldiered at the edges. It provides up to 200 kbit/s data rate, has a transmitter-receiver turnover time of 50 µs and can receive singals of down to -129 dBm at 300 bit/s.

SiPy

SiPy uses CC1125 integrated circuit. It is a tiny computer. It seems to run on propretary firmware. Datasheet and dœsn't provide information about the Wifi special capabilities.

Abilities of Computing Hardware

Many different small computing platforms are available in market, out of which to are considered here:

Arduino

- Microcontroller on which low level software can be loaded.
- The open source integrated development environment (IDE) cares about interrupts, pins etc.
- Appropriate for controling pins (digital/analog) with precise timing.

- Can access serial peripheral interface (SPI) peripherals.
- Many abilities (audio, SD card access, ethernet) need additional peripherals.
- Typically clocked with 8 to 32 MHz
- Price is about 25 € - 30 € per unit

Single board computer (SBC): BeagleBone, Raspberry Pi, Cubieboard, ...

- Computer with GNU/Linux operating system which is started from SD card or internal memory.
- Most models have high computational powers (compared to Arduino)
- Many pins with many different functions (e. g. UART, SPI, I2C)
- Typically many ports (Ethernet, USB, ...)
- Price is about 55 € per unit

According to last sub-section finally transceiver RFM69 HCW was chosen as this module is easily available at study thesis location (Germany), has enough data rate thus fulfils the requirements, works with SBCs and has a reasonable prise. Frequency chosen is 433 MHz as this is an ISM band (Industrial, Scientific and Medical band) which can be freely used by Critical Masses to when they apply the communication platform.

As Computing hardware Arduino Uno and BeagleBone Green were chosen to be tested.

Arduino Uno

It turned out that Arduino Uno was unable to access SD card reliably. Time-to-fail depended from the used SD card model but with any SD card Arduino at some point refused to access SD card.

BeagleBone Green

BeagleBone Green single board computer was chosen because of the ability to work with GNU/Linux so an easy integration with simulation computer was provided.

4.3 Speech Codec

As speech codec Adaptive Multi-Rate (AMR) codec was chosen because it was said to allow certain bits to be extinct. Throughout all computations for the thesis mode MR475 (4.8 kbit/s) was chosen which is sufficient for speech quality while being bandwidth-efficient.

In AMR a second is split into 50 AMR frames, so $f_{AMR} = 50\ \frac{frames}{s}$. (4.a)

In mode MR475 one frame consists of 95 bit which makes (padded to 96 bit) 12 byte. So the total datarate $r_{MR475} = 95\ \frac{bit}{frame} * f_{AMR} = '95 * 50\ \frac{bit}{s} = 4.75\ \frac{kbit}{s}$. (4.b)

AMR Packet Corruption

I couldn't information that some bits of AMR frames can be destroyed without hurting the quality much so I did try myself and wrote a program to scrumble bits.

AMR encoded frames are split in three classes: class A, class B and class C. Class A bits are more important than class B bits and class B bits are more important than, if present, class C bits. Chosen AMR mode MR475 only uses class A and class B bits [3GPP].

By try-and-error was found that AMR reference encoder/decoder keeps bits in standardized subframe-order while ffmpeg (libopencore-amrn) software readily sorts them with class A bits in the beginning and class B bits in the end.

For this thesis some listening tests for different kind of sound data were made. The statement that class A bits shall not destroyed can be confirmed. For full listening table refer to Appendix B Full AMR corruption result chart.

4.4 Data Dispensation and Buffering

- Microphone could potentially discard old data

- Microphone dœs not return future data even if requested

Buffering of packets in aodv-uu implementation. (Limit: 512 packets)

Buffering of packets in FrameSwitch App (Medium Access) (Limit: 512 B? 1024 B?)

→ Precedence of AODV packets over AMR packets

→ This provides an optimal (= minimal) average wait-to-be-sended time (under assumption: packets fitting perfectly in one superpacket, so there's no space loss). [HAMANN] states that queueing to-be-processed elements with "Shortest Processing Time" policy (synonymous to Shortes Job Next) leads to a minimal average wait-to-be-sent time.

→ Problem may be “packet starvation” which means that many small packets come into the sending queue thus never allowing a big packet to be sent. But as route messages typically appear in high intervals this is not a problem.

4.5 The ALOHA-ACA Protocol

Standard ALOHA is a medium access protocol which mainly consists of sending when you have data and trying to repair connection when no acknowledgement arrived.

Reservation ALOHA (R-ALOHA) and slottet ALOHA need a central station which manages the synchronization.

While standard ALHOA leads to a capacity usage of 30%, R-ALOHA performs with usage levels of about 80% [W-ALOHA] which is in general sufficient for the thesis’ throughput requirements. But as there’s no guarantee that a master station will be present all time over the whole cyclists trail R-ALOHA can’t be used in this thesis.

So we use ALOHA-ACA (ACA for acknowledged collision avoidance – although not solely avoidance is performed) which provides an implicit reservation mechanism: a station sends its packets at some arbitrary time and this way it dœs automatically state that it reserves the slot belonging to the sent time.

Time ist split in frames of three seconds which every frame split in 30 slices. (So 10 slices per second, 100 ms each). Stations’ “main slice” is one of slices 0-9. Every main slice is automatically accompagnied by two side slices which are called “main side slices”. All other 27 slices are called side slices. Main side slices are exactly one and two seconds later than the main slice. (So if the main slice is at slice 5, main side slices are 15 and 25.) In effect there’s one main-like slice per second.

Frames have no synchronized borders so when one station thinks it was sending at slices

2	main slice
4	side slice
12	main side slice
22	main side slice
27	side slice

another station whose frame border is 6 slices = 600 ms later could see it as

6	main (side) slice
16	main (side) slice
21	side slice
26	main (side) slice
28	side slice.

The three main and 27 side slices are distinguished by “main” bit in ALOHA packet.

The only thing which has to be synchronized is the slice border. Though this synchronizing is only half-implemented and deactivated in FrameSwitch application[3] it is fully specified in the ALOHA packet structure: The end of every ALOHA packet consists of a simulated preamble and simulated syncword, a simulated length byte (“1”) and a selfconception byte (the content of the simulated “1” byte long packet). In selfconception byte the senders address is repeated and the sender tells what it supposes to be the time in frame (in 16 steps with value 0 for “in the beginning” and value 15 for “next to end”) when sending last byte.

Every station should adapt to the others slice synchronization. I suppose that this approach works well in general but may have problems if on start-up one half of the stations has a good fitting start-up synchronization (happened by “random”, by the time of switching on) and the other half of the stations is delayed by 50 ms (½ slice) to this. When two internally synchronized groups with a 50 ms (½ slice) bias join problem is the same. Maybe this problem then could be resolved by manually switching off an on one of the stations.

4.6 The Broadcast Loop Prevention Mechanism as Routing Method

The broadcast loop prevention mechanism was implemented when programming for this thesis as one requirement of speech transmission was that all stations play the soundfile that a station sends out.

Although included in AODV-UU implementation for this thesis it is an independent routing algorithm.

Broadcast Loop Problem

When broadcasting a packet it will most probably come back because the air is reciproke. And even if it wasn‘t the air there would be most probably triangle positioned stations which would let run the packet in circles. To prevent a high data traffic different measures can be taken which are shown in Table 4.1:

Table 4.1 Methods to prevent an indefinite circle of packet.

Measure	AODV-uu status	AODV-standalone status
TTL	implicitely assumed	implemented
BC-TTL-REC	not implemented	implemented :-)

3 Preliminary name for communication platform

Time-To-Live (TTL) counters belong to the packet data and are set to a high value (higher than the count of hops to the farest thinkable destination station) and decreased by every station when it processes/forwards the packet. When TTL reaches zero, station dœsn‘t forward the packet so it can‘t take air resources any more.

Broadcast TTL recalls (BC-TTL-REC) act when the TTL is still valid (greater zero). They compare the TTL of the incoming broadcast packet with the TTL of the last same-origin-broadcast packet. If the TTL is lower, it is assumable that the packet has made some circles so it gets dropped. As the topology of radio systems may be changing, the stored TTL register expire after some time. To find a reasonable value for the expiry timer three considerations can be taken into account:

1. Maximum-Loop-Time (MLT): This is the time a packet needs to get broadcast to very far stations and comes back on another path describing a circle. When the MLT of a packet is about x, the timer should be set to values higher than x*c ($t > x*c$; c is some arbitrary correction factor for the insecurities in RTT estimation) because otherwise the timer will be reset before the packet has come back an then is inactive, so the BC-TTL-REC mechanism is without effect.
2. Frequency of broadcast packets: Low frequencies don‘t matter (period $p >> x$) because the loop will be prevented by the timer ($t > x*c$). High frequencies (period $p << x$) allow to reduce the timer value as the timer effect is replaced by the new packets. So if it can be assumed that typically every y there‘s a new packet broadcast (e. g. period y = 20 ms for AMR speech data) it is sufficient to set the timer to a slightly higher value than the packet period. ($t > 2*y*c_2$ with c_2 = another correction factor and 2 = preparation for one lost packet). The newly broadcast packets then can keep the mechanism active while the complete operation.
3. Speed of topology changes: If the topology is assumed to change rapidly a degrading link (means more hops away) which incorporates sinking TTL values can be assumed: If the timer waits very long, many valid broadcast packets will be discarded because of their lower TTL value.

So in total, 1.) recommends a high timer value, 3.) recommends a low timer value and 2.) enables us under some circumstances to down-parametrize the timer value without hurting 1.).

Remark: AODV packets have a TTL of 1, so they are meant to reach the direct neighbouring stations but not to be forwarded by them.
So Broadcast Loop Prevention is effective for the non-AODV packets, in other words: about the AMR speech frame payload packets.

Remark: BC-TTL-REC even works if not all stations use this mechanism. It is transparent to the AODV standard.

4.7 Backoff

When the same deterministic algorithm is run on two identical stations and is at the same state in both stations (may happen if both stations are switched on at the same time) it will fail in collision avoidance strategies because both stations will always send at the same time and therefore will not receive the others' message so they'll never get an acknowledgement and consequently never reserve a valid slice. (Reception will fail on [unbeteiligte dritte] stations as well because they're unable to decode the superimposed waves).

Imagine you are walking a gangway which is interrupted by a glass door. You want to proceed and go through the door, but on the other side of the door there's another person which also wants to get through the door. Assume you are in medium hurry and you wait for two seconds until you go. Unfortunately person also tries two seconds after you met. You both, noticing that it won't work out, change your decision and wait further five seconds. Assuming that you and the other have the same program running your decisions will be always the same and you enter a loop which will be infinite.

To avoid this problem you could add some external decision guidance which hopefully is different from the other's guidance. For example, the procedure is following: I count how many building entrances I have walked by today. This is the number of seconds till I try again. If this dœsn't work you use the number of trees which stood at your way today. Let's replay the situation with the new algorithm: You both try it after two seconds, which fails. Then you try it, let's say, after 9 seconds while person tries after 29 seconds. The conflict now is resolved by you walking through the door after 9 seconds.

This example showed how a so-called backoff procedure works. The individual amount of waiting time can chosen from some history or from true random data, if available. In old days' Ethernet and in recent WiFi networks a so-called exponential backoff is used.

Implemented Backoff

The thesis's ALOHA-ACA implementation uses a mixed approach for the binary backoff. When not receiving any acknowledgment for main slice for some defined time (2 frames ≈ 5.9 seconds for new main slices (reservation tries) and 3 frames ≈ 8.9 seconds for existing main slices) or after receiving a conflict notification for main slice a call to hop(), the pseudo random output helper, returns a boolean value which controls the decision whether to change the main slice at all. Next usage of hop() is the aimed distance (decision between 4, 5, 6 and 7) to other slices, probably some other stations' main slice, when walking through the slice table. The fold-and-find algorithm can be fine tuned with an input array.

4.8 Random in the Implementation

When simulating the tranceivers' behavoiur it is very useful to achieve a deterministic run which can be repeated in exactly same manner. That means that a newly started simulation shows exactly the same result (or crashes at exactly the same location) as a formerly run same-parametrized simulation. Such kind of precision is condition for debugging deferred program crashes (see Appendices) but also to look at the same run with different debug message contexts enabled (e. g. watch only packet flow or watch demodulation decisions or combine main simulator event queue info with the AODV-UU messages).

So as random seed the sations' address which is considered unique is used. The seed is fed to a tiny random algorithm () as Code 4.1:

```
uint32_t random_number(MHI_ARG_single) {
    gl rand_state = gl rand_state * 1103515245 + 12345;
    return ((unsigned)(gl rand_state/65536) % 32768);
}
```

Code 4.1 Random generation function as conveyed by "rand" man page [MAN] from POSIX.1-2001

(rand_state act similar as a global/static variable and MHI_ARG_single is some helper which dœsn't matter here.)

Ever when a decision has to be taken, a new bit of the recent random data is read out by hop() function. When all bits of the random number are used, the next random number is created with random_number().

5 Simulation Design

In contrast to high-level network simulator NS-2 the approach taken in this thesis is to keep as close as possible to hardware and to physics – knowing that the CPU time effort will be much higher than when simply “transmitting” or blocking packets by well-defined random.

So in this chapter you find some information on the simulator design.

5.1 Event Loop

After startup the simulator thread enters event loop. Event loop handles “execution claims” and “events” while properly advancing the synchronized global clock.

Claims are a simple time value. They are stated by application and ensure that application won’t notice anything which happens before the stated time. Claims are automatically created by multihw.c (chapter 5.3) which keeps track of the applications’ execution state. Theoretically it is possible that an application (or multihw.c) changes it’s claim time without prior/subsequent notice to simulator but in practice this isn’t needed.

Events are implemented as time value and type information and can carry additional data. Event types are following as in table 6.1:

Table 5.1: Im Simulator verwendete Event-Typen

Type	Initiator	Description
'1'	simulator	Simulator initialized, now the actual simulation begins. [Happens once at the very beginning of the event loop.]
'$'	simulator	Tell transceiver that the submitted data is sent out.
'R'	transceiver (“Give my wave data to the other transceivers”)	Inform transceiver that it has received some waves.
'R'	simulator (Giving further wave data to the transceivers after the original call was interrupted by claim/event)	Inform transceiver that it has received some waves.
‘0’	application	Please quit all simulation. [Prioritized event – any other thing will happen behind that.]
‘E’	application	Please quit all simulation because here an severe error encountered. [Prioritized event – any other thing will happen behind that.]
[‘C’]	[simulator]	Telling application that it has a claim now. [This is not relevant for event loop.]

5.2 Electromagnetic Wave Handling

When designing a radio communication system the electromagnetic waves are modelled as sine waves with a frequency and an amplitude. With regards to antenna design the polarization is also taken into consideration but in this thesis antenna design plays no role and polarization isn’t taken into account. Usually one transmitter emits one wave but MIMO systems which are not discussed in this thesis emit multiple waves from several antennas aiming to use spatially separate channels. [RPRO, chapter 8.3]

This chapter explains how the simulator handles and sum up electromagnetic waves.

Data format

Computational unit is a so-called “waveblip” which depicts an electromagnetic wave with a constant frequency, constant amplitude, constant phasing and a duration. In this thesis polarization dœsn’t matter as all antennas are assumed to be dipole antennas and to be installed vertical. Also noise isn’t processed at any stage. Consecutive waveblips are stored in “wave_sequence” struct where enough memory is allocated. As interchange unit the struct

"wave_seq_segment" is used which points to one wave_sequence and contains information what portion of the wave sequence is already read out.

Wave composition

We assume that every station emits one pure sine wave around carrier frequency when sending. For every bit modulator creates a "waveblip" with constant frequency, constant amplitude and given duration. The transmission of packets thus generates a bunch of waveblips, see Figures 5.1 and 5.2.

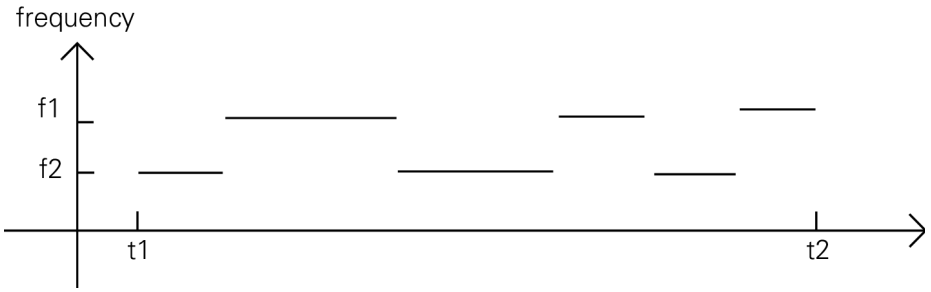


Figure 5.1: Station a emitting bit phrase 01100101. This is represented in 6 waveblips: one bit length f2, two bit lengths f1, two bit lengths f2, one bit length f1, one bit length f2, one bit length f1 with "bit length" = duration of one bit.

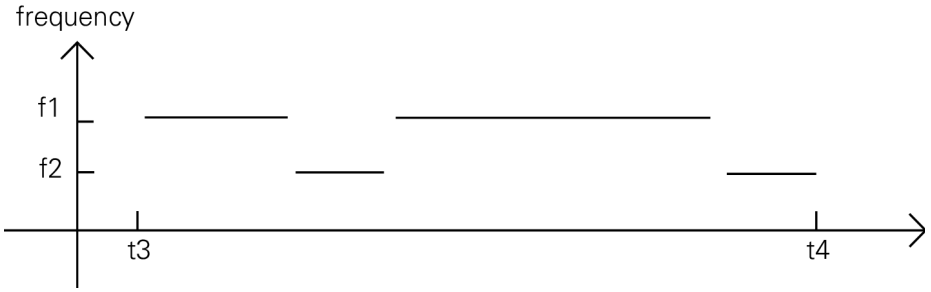


Figure 5.2: Station b sending bit phrase 11011110. This is represented in 4 waveblips: two bit lengths f1, one bit length f2, four bit lengths f1 and one bit length f2 with "bit length" = duration of one bit.

Wave addition (split in time)

If only one transmitter is sending there's no need to add waves. If two transmitters are sending at the same time the addition routine splits the waveblips in time before doing the mathematical addition, see Fig. 5.3.

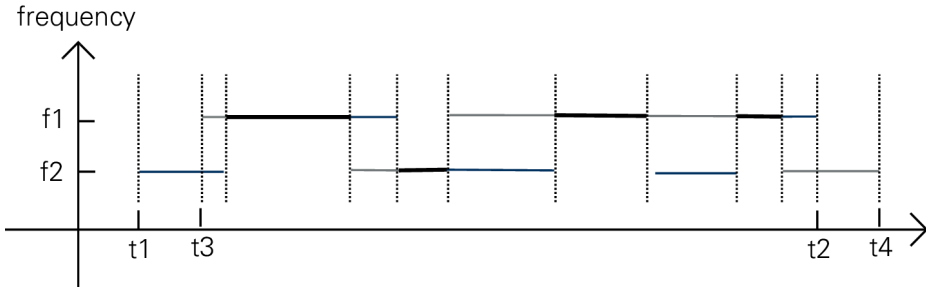


Figure 5.3: Illustration of overlayed waveblips from stations a and b when bit phrase of b (gray) is slightly delayed to bit phrase of a (blue). The dotted lines indicate where the waveblips will be split.

Wave addition – simple case

In Figure 5.4 you see the added wave sequences of Figures 5.1 and 5.2 for the simple case that – in the chosen FSK modulation – the frequency is $\text{f1} = f_c + \frac{1}{2} f_{dev}$ or $\text{f2} = f_c - \frac{1}{2} f_{dev}$ (5.a)

and both waves arrive with same phase and amplitude.

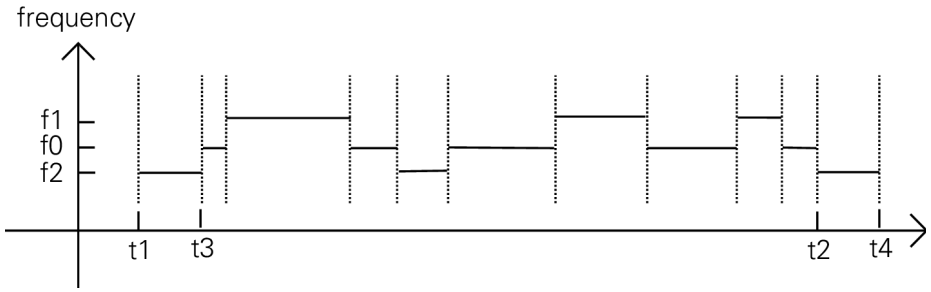


Figure 5.4: The waves of the simplified case (Figures 5.1 and 5.2) finally added. The dotted lines indicate the borders of the waveblips. The fading of the signal at frequency f0 is not shown.

In reality, inaccuracies of carrier frequency and deviation are very probable so the simulator must perform a more generic approach for amplitude, frequency and phase addition which is discussed in following section:

Wave addition – approximated solution

As shown in chapter 2.1 (Physics of the Air) there's no closed solution in terms of a single sine wave which can be easily processed by the emulated demodulators. So the simulator (especially `air.c`) has to approach a good (fast to calculate while accurate and easy to handle) solution when adding up $S_1 = a_1 \cdot \cos(f_{s1} \cdot t + p_1)$ and $S_2 = a_2 \cdot \cos(f_{s2} \cdot t + p_2)$ (see 2.i). We assume $a_1 \geq a_2$ w.l.o.g..

First of all if $a_1 \gg a_2$ we don't have to consider the weaker wave. In air.c this is done by probing factor $b = 10$: if $a_1 > b \cdot a_2 = 10 \cdot a_2$ then S_2 is ignored.

Then the envelope change is checked.

If envelope is changing rather slowly over waveblip duration (that is, either if the waveblip is very short or the frequency difference is very small, for actual formula see Appendix C) the procedure dœs test for constructive/destructive inteference and generates a resulting waveblip accordingly.

If the envelope is changing very fast over waveblip duration (that is, more than 60 flips (= periods) in waveblip duration) it is assumed that the demodulator electronics don't recognize the frequent changes but "sees" some wave with some lower amplitude instead. From Fig. 2.2 on page 7 it is derrived that the frequency of the higher amplitude wave is more present than the frequency of the lower amplitude wave. The higher the amplitude difference factor the higher is the dominance so the adding procedure mixes both frequencies with the ratio of their respective amplitudes.

If the envelope is changing not slow and not fast the addition procedure would actually have to calculate all interference states for the not-too-many periods. This would mean to create 4*[number of interference periods] resulting waveblips. But for the sake of CPU time only the very first ark is calculated and the rest of time the wave is set to zero.

After having processed all waveblips till next event loop break the resulting wave sequence is given to the transceivers.

Remark: the simulator checks the inteference flips (= envelope periods) with respect to waveblip duration. This is independend from any transceiver settings (especially bit rate) which supports a clean seperation of simulator and simulated hardware configuration. But as downside it depends on the actual waveblip fragmentation which makes precision harder to control.

Further remark: In case of more than two stations sending at one time for simplicity of calculation two waves are added and the further waves are added to the result sequentially. For numerical reasons the waves with highest amplitude are considered first. This also makes calculation more efficient as the lowest amplitude waves are (correctly) likely to be ignored.

5.3 multihw Hardware Abstraction Layer

multihw.c is the key code which mediates between the actual application and – by choice – the simulator or real hardware. As it integrates the C project parts (simulator, application, AODV-UU) it is written in C.

Timing / Execution Flow

Startup is organized following way: The main thread of the simulator binary is the startup code and the event queue. Startup code initializes the five stations and for each of those it

invokes the applications' multihw instance which initializes a new thread. The new thread calls the startup code of the application (called "FrameSwitch") which pauses after startup giving the execution flow back to simulator (or remaining application startups). So all stations get started one-by-one. Inside the application startup the stations' transceiver will be initialized.

Then event loop gets started and operation is straightforward. (Only at second view straightforward is the mechanism to hand over wave data because different threads will alternately invoke the wave addition function.)

In AODV-UU the main loop has a polling mechanism which works with `select()`. When no data is coming, select() will be ineffective and the service tasks (`timer_age_queue()`) are invoked from time to time then processing the to-be-deleted route entries and packets. In CN (stands for "Communication Networks") variant which is in effect the multihw interface the AODV-UU service tasks are called at every received packet and every payload which is to be sent.

RFM/SPI Access

multihw provides an integrated interface for accessing the RFM regfile. As RFM regfile gets accessed by SPI, multihw dœs also support SPI.

Unified Logging

multihw introduces functions to generate pretty log messages which automatically incorporate stations' name and system time. These functions are implemented in a way they do work with Arduino as well as on BeagleBon or in Simulator.

Exit on Error

multihw provides a exit on error facility which blinks some LED on the target system if this target system is not Linux or breaks the process execution on GNU/Linux system.

"Global" Variables

[GVAB] tells some reasons why global variables are not recommendable in most situations:

- Non-Locality: When the access is scattered all over a big program it is less comprehend-able and more difficult to debug.
- No Access Control or Constraint Checking: This is a problem of stability and security.
- Implicit Coupling: It is not clear which of several global variables belong together.
- Concurrency Issues: Affects multithreaded programs.
- Namespace pollution: you can accidentally use a global variable used by other program parts without recognizing the double use.
- Memory allocation issues: different per programming language, in C it may be unclear which compilation unit shall carry the actual variable definition/space.

- Testing and confinement: In a testbench a program which is called multiple times consecutive it cannot be ensured that all variables are reset to the initial state

Despite the fact that global variables usually are a design flaw of modern programs (even if sometimes needed) for our own purposes stations may not use global variables at all, so here's one added reason:

- Parallel instantiation: When running several instances of one program, especially in simulator, this program accesses the same variable thus either interchanging data (whilst it should be seperate for the simulation purpose) or showing strange behaviour or simply crashing per perplexity. If the program dœs not crashes, dœsn't behaves strange and dœsn't interchanges data in parallel instantiations, the global variable is unnecessary and can be safely removed.

If a station would record the number of open route requests in a global variable "`rrnum`" for example another station which has recorded another number will make wrong decisions when reading the new wrong value.

The AODV-UU [AODV-UU] code has some global variables scattered over the source files. It had been refactored for NS-2 interfacing: A C++ class was defined and all variables were declared as belonging into this class. But these variables are copied (= "code duplication") so that every removal or creation of global variables of standard AODV-UU has to be reflected in its NS-2 interface which makes it difficult to maintain. Technically this is realized with preprocessor macros which are widely known for the #ifndef FILENAME_H line at the beginning of common header files.

The thesis' approach that was taken to connect AODV-UU to simulator is a plain C approach: All global variables have to be listed on one place (the main struct) and cann't be declared on any other place. So this struct is declared once global if running as "normal" AODV instance but is handled locally when simulating several instances. Like the NS-2 interface this refactoring is realized with preprocessor macros but in a more extensive way.

6 Evaluation

Some measures are developed for this thesis which are concentrated in following Table 6.1. There you can refer to the subsection of the particular measure:

Table 6.1: Measures and their feasability for evaluation/benchmarking. A measure is called "automatic" if the numbers can be achieved by measurement code while or after simulating. A measure is called "informative" if some generic knowledge about the algorithms can be easily deduced from the numbers. This column "Informative?" is copied from chapter 6.6.

Measure	**Automatic?**	**Informative?**
Simulator CPU time (6.1)	yes	no
Sending Performance (6.2)	yes	yes (for debugging)
Packet Loss (6.3)	yes	yes
Registered Radio Reception (6.4)	yes	yes
Time to Design (6.5)	no	yes

6.1 Simulation CPU Time Consumption

On the used notebook with 2-core CPU (Intel® Core™ i5-7200U) [CPU] the simulation consumed a constant CPU load of 120 % (1.2 cores) (monitored with "`top`" utility). As there's only one active thread at every time (execution flow gets passed between stations and simulator, see chapter 5.3) it an be assumed that threading costs 20 % overhead.

Times were measured using the "`time`" utility of the hosting GNU/Linux system with following command output (standard setting):

```
$ time ./simulator >/dev/null 2>&1

real 2m28,786s
user 1m55,782s
sys  1m0,835s
```

This matches the "`top`" observation as the actual consumed cpu time ("user" + "sys" = 177 s) is 119 % of the command execution time ("real" = 149 s = 177 s / 1.19). cific measurement of the individual stations.

The resulting graph of the CPU consumption of all parameter sets is Fig. 6.1. There you see a constant system CPU and a changing userland CPU consumption for different parameter sets. The only system interactions of simulator are file writing and thread switching. All other elements (SPI access, radio transmission, calculations) are userland operations.

The file writing can be excluded as source of varying times as can easily discovered with dd command:

```
$ dd if=/dev/zero of=test.file bs=5 count=16000000
16000000+0 Datensätze ein
16000000+0 Datensätze aus
80000000 Bytes (80 MB, 76 MiB) kopiert, 22,8705 s, 3,5 MB/s
$ dd if=/dev/zero of=test.file bs=50 count=1600000
1600000+0 Datensätze ein
1600000+0 Datensätze aus
80000000 Bytes (80 MB, 76 MiB) kopiert, 2,39135 s, 33,5 MB/s
```

Our quick test with blocksize 50 B (guess on average line size) showed that a program which copies 80 MB (like the largest logfile simulator produced) with blocksize 50 B needs at most 2.3 seconds of (system) CPU time. In standard case, file writing will be buffered with buffer sizes >> 50 B [SO] so file writing time is neglectible.

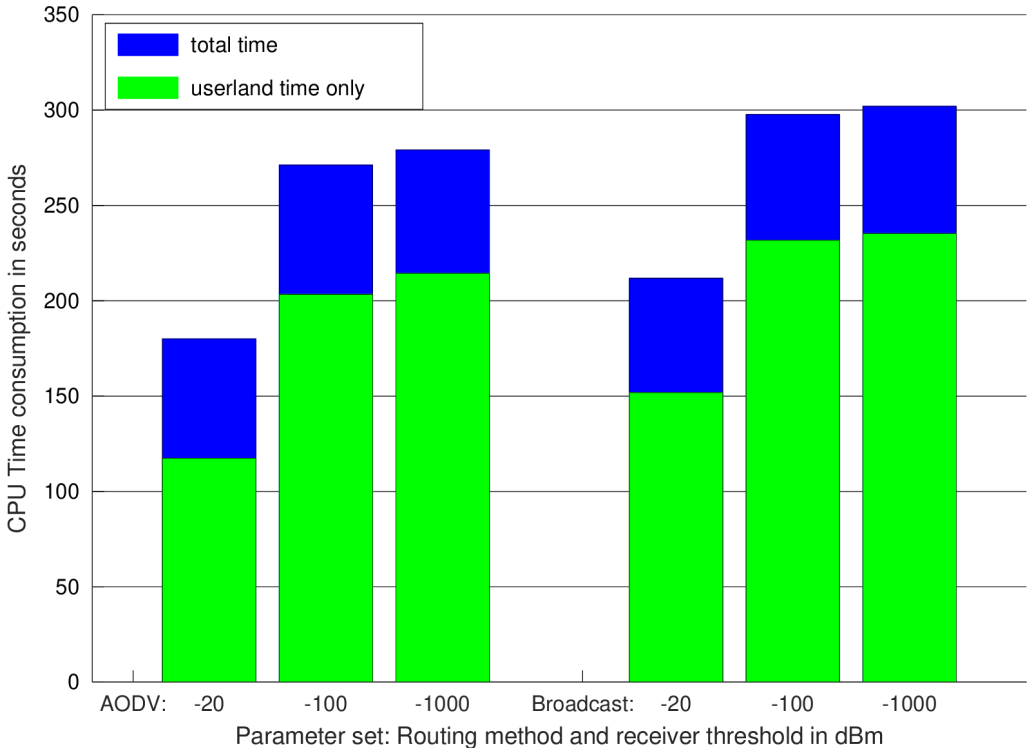


Figure 6.1: CPU time consumption for the different parameter sets

As all other considerations are excluded it can be stated that the thread switching overhead keep same independent on routing method or receiver threshold. As threads are only switched on the multihw delay request (`multitimer_delay_coarse()`) the time consumption shows that the amount of delay requests is independent on routing / data

reception. This is right in regard to (FrameSwitch) application main and reception loops which have similar timings independent on the actual application state.

System CPU time is explained; what about user cpu time? I assume the wave addition/transmission as well as demodulation being the main CPU time intensive part of the simulation. This explains the low CPU usage of the -20 dBm threshold parameter sets in contrast to the -100 dBm / -1000 dBm parameter sets.

-100 dBm and -1000 dBm parameter sets are similar (amongst same routing method) because in most cases the intermediate stations (B/C/D) do receive the data from O and perform relaying transmits. This is very clear for broadcasting variant. AODV is slightly more cautious as it sends only when a route is known. At -100 dBm a route is mostly known but not always so there's a gap between user CPU time of -100 dBm and -1000 dBm with AODV routing.

-20 dBm with AODV is lowest value in test because nearly never a valid route to station W is known so all the stations don't send data packets; they only send hello packets.

6.2 Sending Performance / AMR Frame Count

Apparently the number of sent AMR speech frames is not equal amongst the different simulation runs. This was not expected as `amrload` mechanism had enough frames to send in every run: One run takes $t_{tour} = 7267\,s$ till the pretended cyclists on the map come to their goal. The application waits for 10 seconds in the beginning till it starts loading frames. So speech data is needed for $t_{AMR} = t_{tour} - 10\,s = 7257\,s$ which gœs along with

$$t_{AMR} * f_{AMR} = 7257\,s * 50\,\frac{frames}{s} = 362{,}850\ frames\,. \qquad (6.a)$$

This is less than the available number of frames (486,779).

The FrameSwitch application in AODV mode did fetch all of the 362,850 frames (in detail: 362,877 @-20 dBm; 362,550 @-100 dBm; 362,852 @-1000 dBm) regardless of the receiver threshold. In broadcast mode the FrameSwitch application did fetch only 240,000 frames (in detail: 254k @-20 dBm; 226k @-100 dBm; 233k @-1000 dBm).

This is explainable with the buffer mechanism in combination with the loops and slice allocation: The FrameSwitch application on station O dœs load in broadcast variant the AMR speech frames via buffer and then sends them. The AODV-UU buffer plays no role as all packets with broadcast address are directly routed to the sending hardware. When sending hardwares' buffer (also emulated by FrameSwitch) capacity is reached, the speech frame loading will stop. The hardware buffer is byte oriented and the packets will never be deleted due to their age. As ALOHA-ACA fails to provide enough sending slices for station O not all of the speech data is fetched.

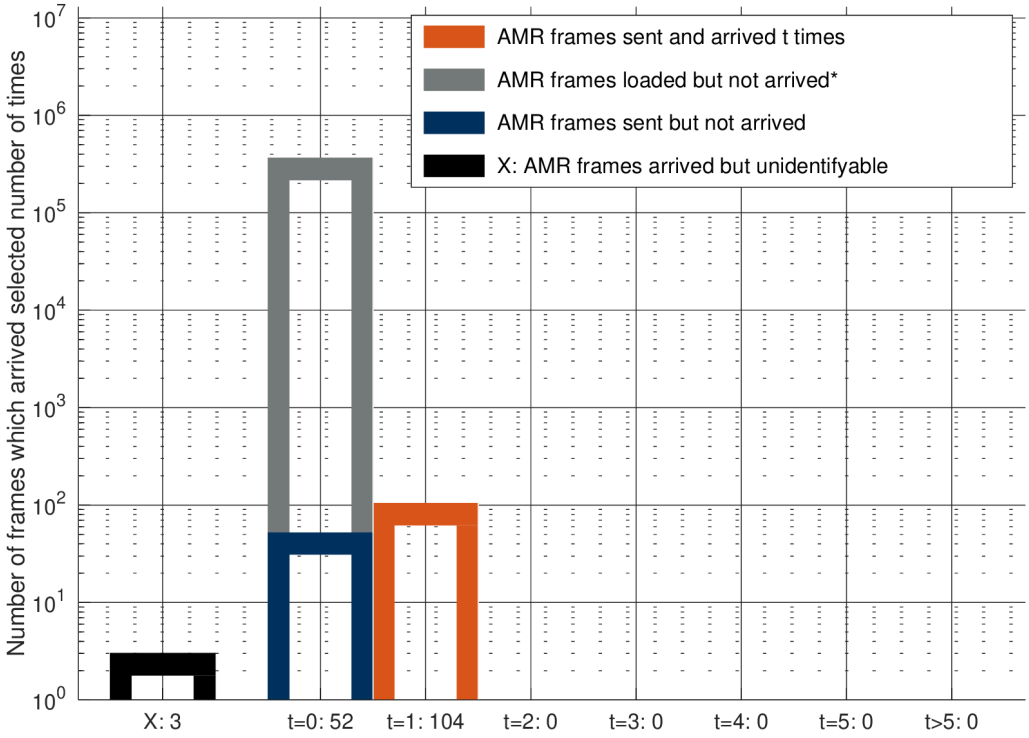


Figure 6.2: Arrival count of sent AMR speech frames at station W using AODV routing with receiver threshold of -20 dBm. Logarithmic plot.

** 362,722 frames*

In AODV variant the FrameSwitch application has different situations. When a route to destination is available the AMR speech frames are loaded, routed through AODV and directly given to hardware. Same procedure as with broadcast variant applies: when hardware buffer is full, speech loading will stall. But when no route is available, FrameSwitch application will anyway load speech frames (besides – this is a bug). But as there's no route the AODV-UU packet buffer is activated and keeps all the speech frames thus leaving the hardware buffer empty. So FrameSwitch will load as many speech frames as possible (making AODV-UU buffer very big) and get's limited by amrload which dœsn't allow to load future frames. This happens till the end. [Besides: There is too few time for send all theses frames (ALOHA-ACA has not enough sending slices and/or AODV dœs not provide a route) so they will be wiped out by AODV-UU packet buffer timeout which is a bad sign, but this isn't important for the effect of AMR source frame loading which is explained here.]

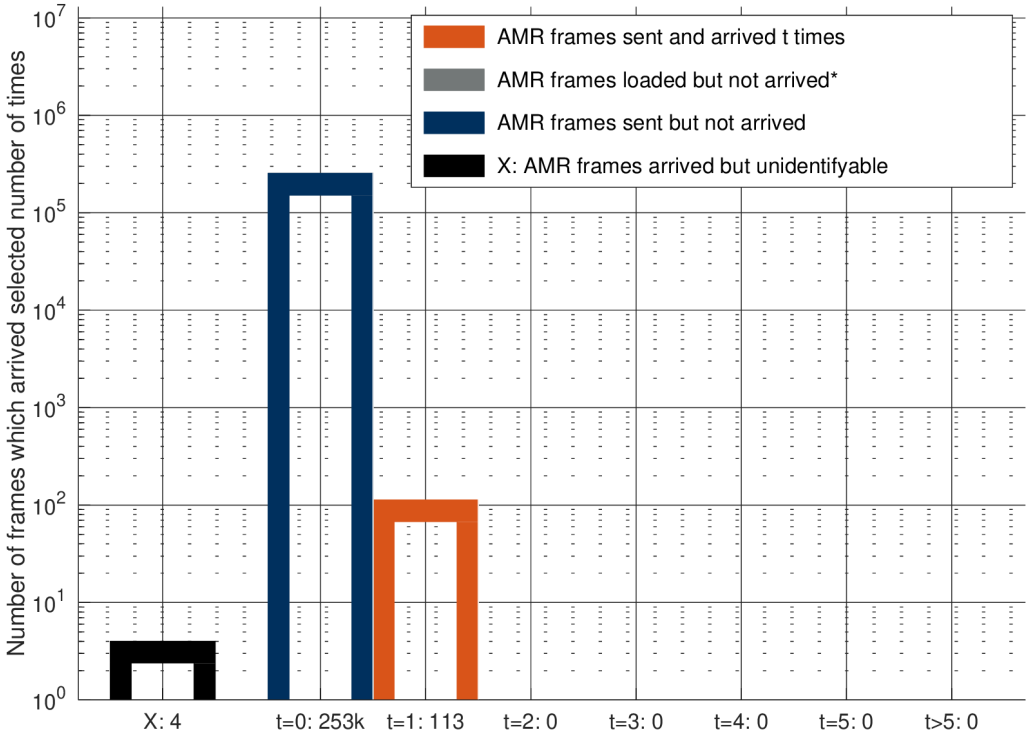


Figure 6.3: Arrival count of sent AMR speech frames at station W using broadcasting with receiver threshold of -20 dBm. Logarithmic plot.

** like sent frames → "loaded" bar hidden by "sent" bar*

An extreme example can be seen with AODV/-20dBm (see Figure 6.2): All 360,000 frames are loaded, but only 156 are sent. In contrast, broadcast/-20dBm loads lots of frames and sends them all, but only 113 do arrive (see Figure 6.3) (which is similar to AODV where 104 frames arrive).

Note that further Figures can be found in Appendix A.

6.3 Packet Loss / AMR Unique Frames

The packet loss was counted with an auxiliary frame comparison program. It loaded all 490,000 frames from the source file and sorted them for faster access. Then it increased for every received frame (printed hexadecimal in unified log) the receive count of the individual frame and registered the received sequence numbers of this frame. The number of frames for which this was not possible (no bit-exact representation in source data found) is counted in a special variable.

Receive count dœs matter because in broadcast mode it is easily possible that station W receives the packets from the nearest station but also from the farer stations. In AODV mode it

is also possible that the frames arrive multiple times as the routing tables sometimes are outdated with regards to actual link situation.

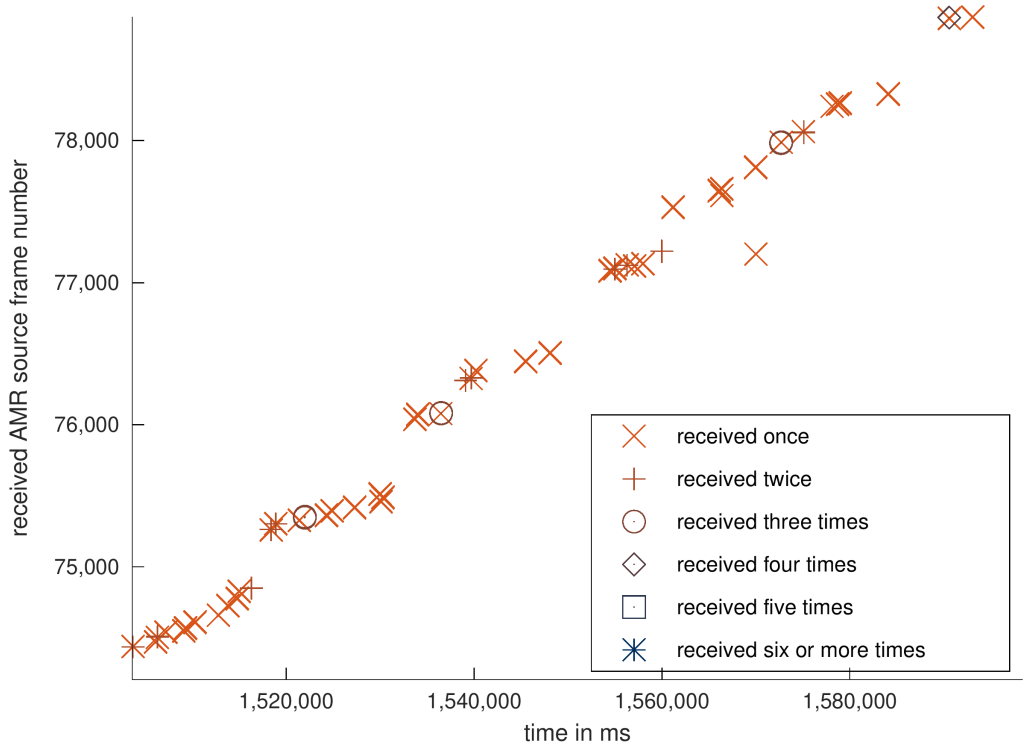


Figure 6.4: Reception time and count of selected frame numbers for station W. Many of the symbols in the graph appear more bold than in the legend. This is deducible to the ALOHA-ACA packets which have capacity for two AMR frames. (ALOHA-ACA packet capacity is constrained by transceiver FIFO size.) So here you see many pairs of consecutive AMR frames.

Parameter set for this figure was receive threshold -100 dBm with AODV routing.

In Figure 6.4 you see many small gaps and few big gaps of about 400 – 500 frames every 1000 frames. Figure 6.5 shows the overall counts for whole simulation time which indicate that only one quarter of the sent frames (sent by station O) was received at station W.

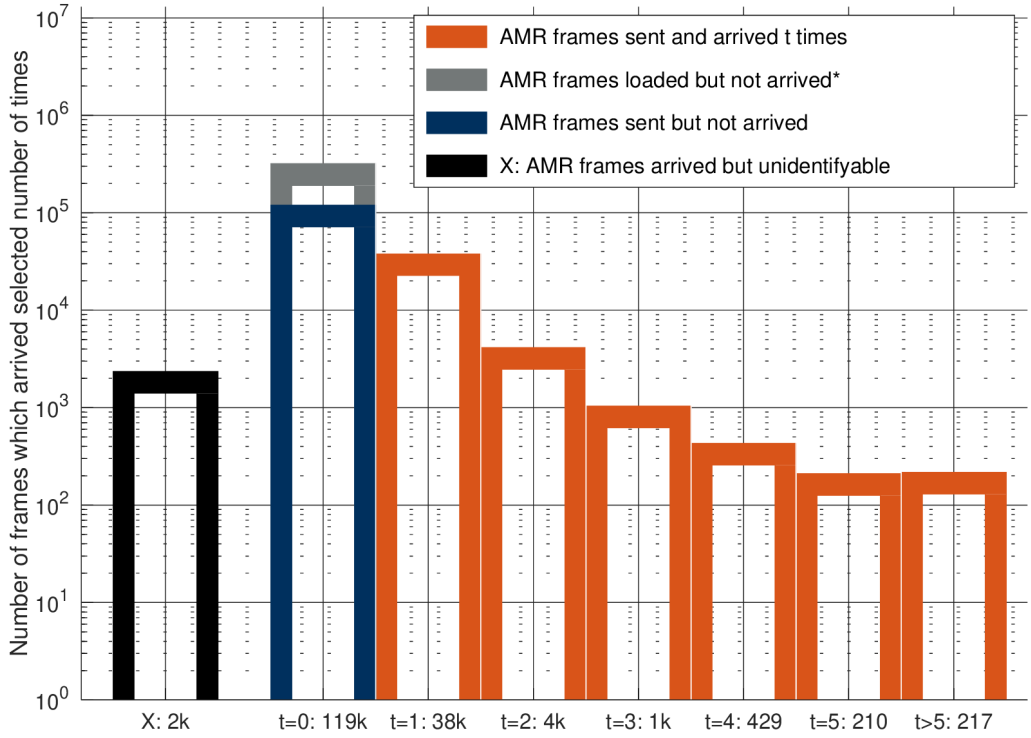


Figure 6.5: (Non-)arrival counts for station W for parameter set AODV/-100dBm. Only one quarter of the sent AMR frames could be received.

**: 319-thousand*

Situation is different for the broadcast mode. More frames are actually sent and The losses are fewer and more equal distributed, see Figure 6.6. Anyway Figure 6.7 shows a similar reception rate. Situation is better when getting closer to sending station, so you see in Figure 6.8 that station D receives much more frames: 168-thousand out of 238-thousand sent frames.

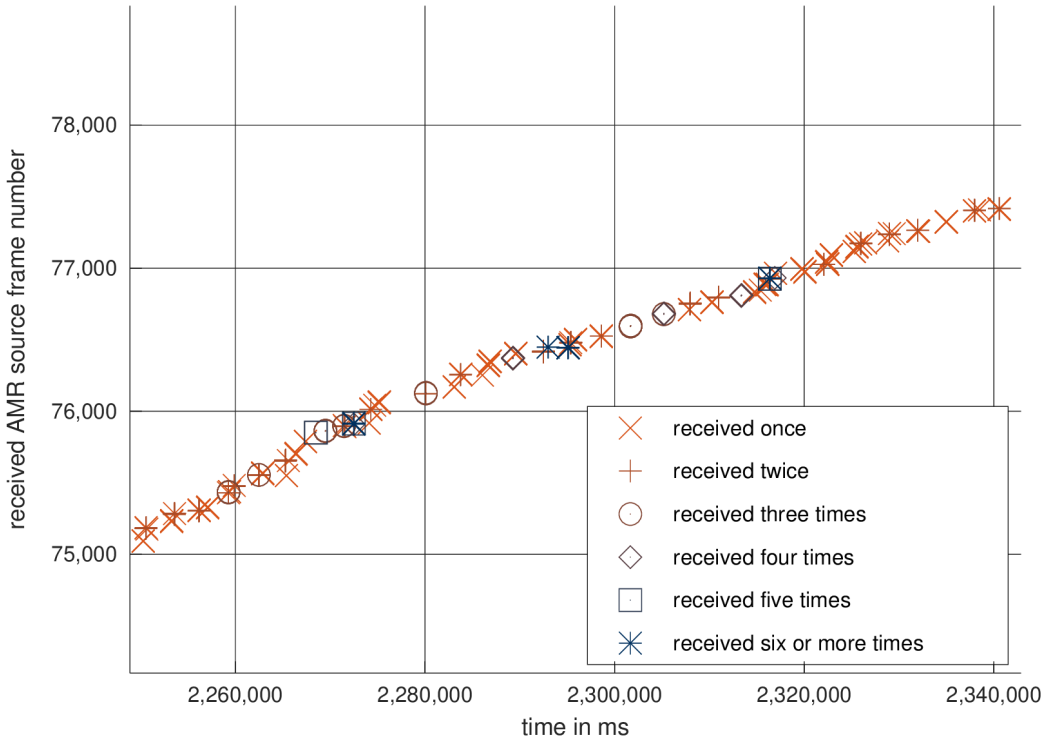


*Figure 6.6: Received AMR frames at station **W** in **broadcast** mode at **-100 dBm**. (Selected interval)*

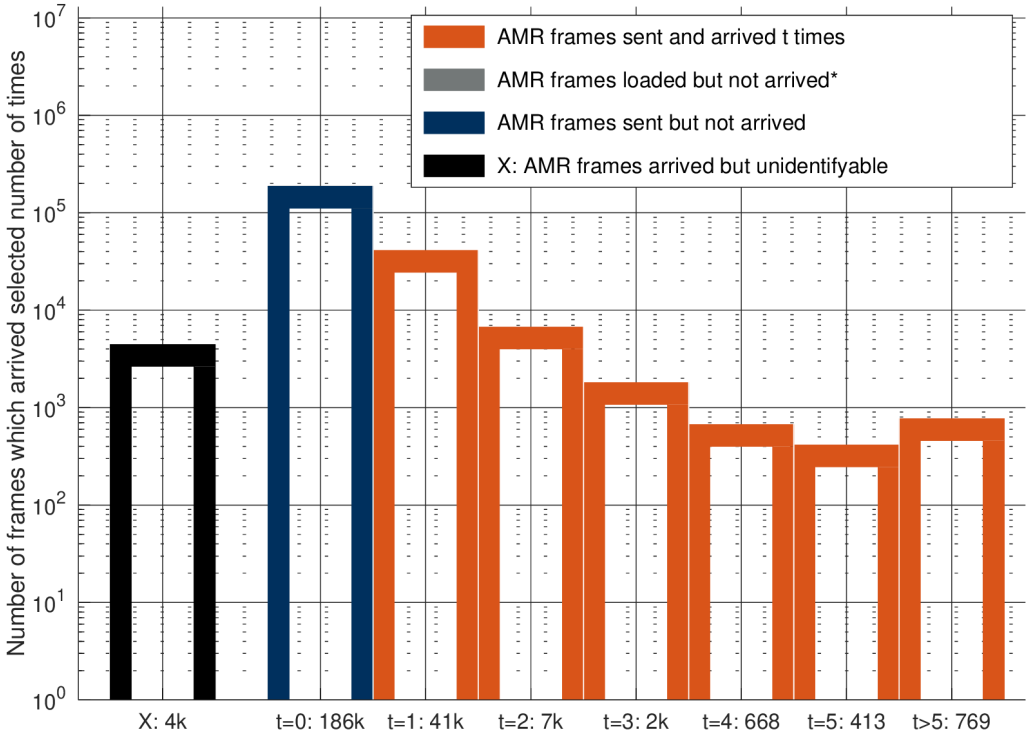


*Figure 6.7: (Non-)arrival counts for station **W** for parameter set **broadcast/-100dBm**. Only one quarter of the sent AMR frames could be received.*

** like sent frames → "loaded" bar hidden by "sent" bar*

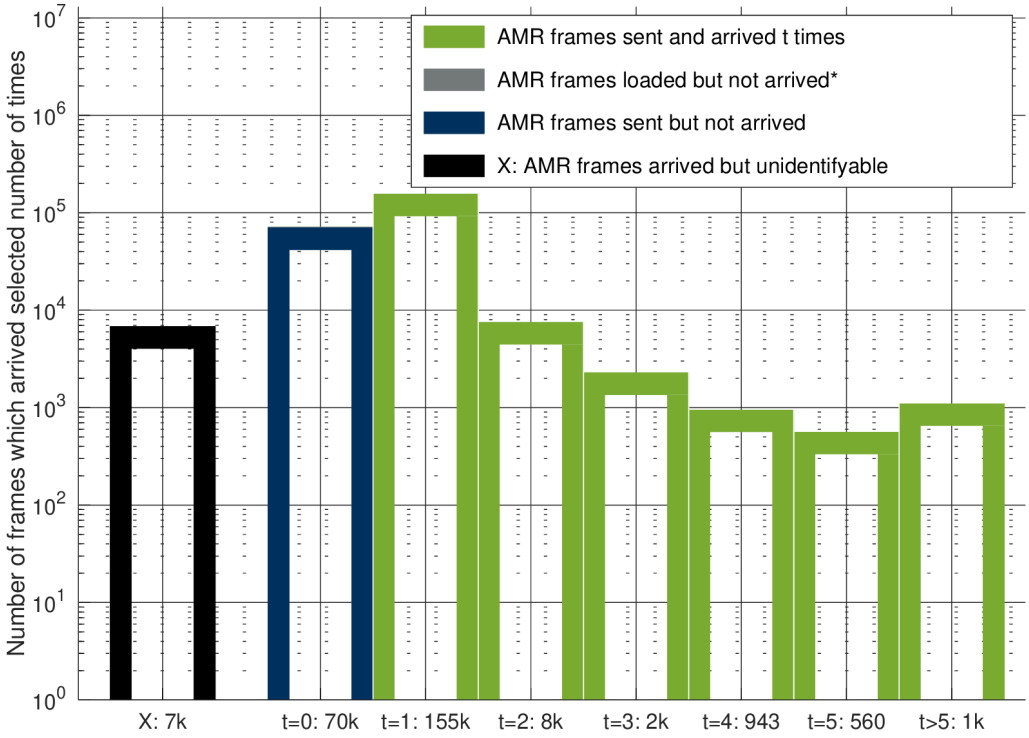


*Figure 6.8: (Non-)arrival counts for station **D** for parameter set **broadcast/-100dBm**. Only one quarter of the sent AMR frames could be received.*

** like sent frames → "loaded" bar hidden by "sent" bar*

More graphs can be found in Appendix A.

6.4 Registered Radio Reception

As found in chapter 2.1 it is recommendable to check spectrum usage with regards to efficiency. This thesis dœsn't provide a mapping of spectrum use against application benefit, nor dœs it consider spectrum sharing with different kinds of application. But one approach for metering the emissions of the guidance system developed in this thesis is to meter the imissions of equal devices, called "Radio Reception Registering" which is done in the unit "ecrm".

Cumulating Immission

Idea is to create a standard which is comparable over a wide range of applications and frequencies. Because the reception strengths easily vary among many orders of magnitude but the induced side effects / disturbance are not so heavily variing over frequency range a logarithmic approach could work well. The logarithmic unit dB / dBm is widely known in electrical engineering. But it shows negative numbers which is unusable for a cumulating metric. This problem can be wrapped with an exponential funtion with a small basis (1.5). So

the signal strength translation function is $Z(a_1) = b^{\log_{10}(\frac{a_1}{\mathrm{mW}})} = 1.5^{\log_{10}(\frac{a_1}{\mathrm{mW}})}$ $with\, b = 1.5$. (6.b)

The result is given the unit "ecrm" (Exponential Cumulated Reception in mW). It cannot become negative because of the exponential function. Table 6.2 shows selected values of reception signal strength and their translated values:

Table 6.2: Selected translation between absolute received signal strength and cumulation value.

Receive signal strength		**Translated value**
absolute	**in decibel**	**rounded, in ecrm**
10 mW	10 dBm	1.50 ecrm
1 mW	0 dBm	1.00 ecrm
100 µW	-10 dBm	0.67 ecrm
10 µW	-20 dBm	0.44 ecrm

So far this metric dœsn't support differentiating between different types of modulated data, such as analogue frequency modulation / amplitude modulation or digital frequency shift keying / phase shift keying or other schemes.

For every fraction of time this translated signal strength is summed up. So the cumulated function $C = \sum_{i=1}^{w} t_{w,duration} * Z(a_w)$ (6.c)

which automatically results in unit $ecrm \cdot s$, which is abbreviated "ecrms".

Consider following sample case: Station W and O stand next to each other for 20 minutes in the beginning of the ride which will go straightforward. Shortly after the ride started, station O reaches it's "end position" 900 m behind station W, then ride gœs on for 120 minutes. We assume that in the beginning station W receives 100 µW from station O (-10 dBm. Besides – this is the maximum receive power of variants of IEEE 802.11 wireless networks) and in whilst riding station W receives 0.1 pW (-100 dBm. Minimum receive power of variants of IEEE 802.11 [W-DBM]). Then the total received power metric results in 925 ecrms (table 6.3):

Table 6.3: Final sample calculation of ecrm scaled radio emissions

Received signal strength	**Translated value**	**Duration**	**Product**
100 µW	0.67 ecrm	x 1200 s	= 800 ecrms
100 fW	0.017 ecrm	x 7200 s	= 125 ecrms
		Total sum:	**925 ecrms**

Results of Simulation

In the software implementation of this thesis the transceiver part is responsible for calculating the registered radio reception metric. Even if the incoming wave is too weak to exploit it's power it is added to the calculation. Calculation is made per-waveblip, so it is far more accurate than the example calculation above. Downside of the approach with the transceiver part is not fully accurate as the transceiver rejects any wave segements when in TX mode (which is the case when sending). So for future implementations calculation should be implemented in simulator.

Results for all stations and parameter sets are shown in Figure 6.9:

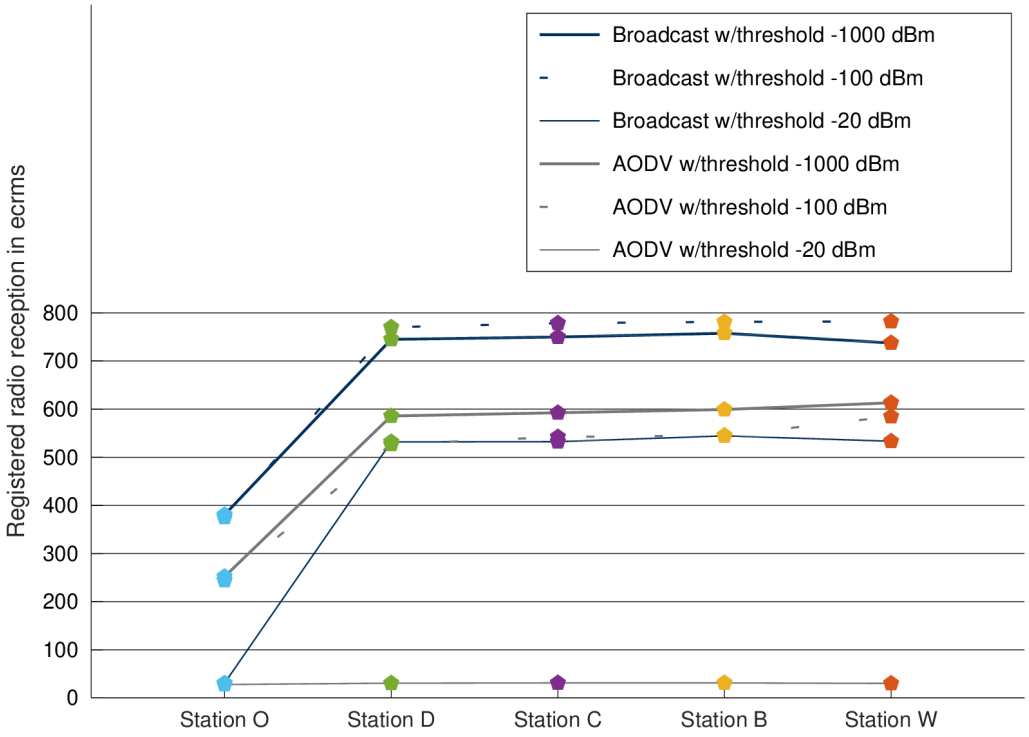


Figure 6.9: Registered Radio Reception. Values are measured and shown in "ecrms" unit which tries to indicate the usage of the air spectrum resource. Lowest values are around 30 ecrms, highest value is 781 ecrms (broadcast variant receiving all waves: -1000 dBm).

As you can see station O is obviously sending more than the other stations. While sending it can't receive data so the incoming waves aren't registered for radio reception.

From -20 dBm curves it can be deduced that AODV sends very few data when knowing that it won't arrive at destination (no route available) thus being very economical in terms of spectrum efficiency in contract to broadcast method.

The broadcast reception at -1000 dBm is lower than at -100 dBm because a) the number of originally sent frames from O is same but b) the first stations (esp. W) receive more frames and do re-broadcast them so they are busy sending and while sending the ecrm register is inactive. (see above).

The AODV reception at -1000 dBm is higher than at -100 dBm because a) the number of originally sent frames from O is higher because a route (direct route) is known and b) the intermediate stations don't relay so they have more time to register the ecrm values which outweighs c) that the intermediate stations don't relay so they don't contribute to radio level.

Please note that the reception threshold has no influence on the ecrm measurement itself because this is taken before input amplitude constraint checking.

6.5 Time to Design

An important factor in nowadays technics is the time you need to design a device → see buzzword “skills shortage”. The less time to implement the more probable is the technology to get widespread.

Of course the simulator dœsn’t measure the time needed for development I have written down the time I needed, and it is also difficult measurable by other methods. But you can try to record the time you needed for certain tasks. Looking at my list I can tell that the (understanding and) adaption of AODV-UU code has taken about tenth the time which I needed to implement the Broadcast Loop Prevention mechanism.

(Much time has gone to all those C crashes which incorporated the need to grep through the debugging output.)

6.6 Comparison of Routing Methods

In the preceding sub-sections some measures were evaluated. This shall be concentrated in Table 6.4.

Table 6.4: Measures and their feasability for evaluation/benchmarking.

Measure	Outcome	Informative?
Simulator CPU time (6.1)	AODV system uses less CPU than broadcasting.	The numeric results don't tell us much because it underlies many influences (compiler optimization level, wait intervals/frequency, crosstalk which requires wave addition, ...)
Sending Performance (6.2)	AODV loads all data but dœsn't send all → some data is lost in buffers. The broadcast method loads as much data as it sends.	Yes, this gives a hint that AODV buffering should be improved.
Packet Loss (6.3)	The total number of received packets is similar amongst AODV and broadcast routing methods. Broadcast performs slightly better in absolute numbers. But as more packets are send in broadcast mode, relative packet loss rate is clearly worse.	Yes, this is a good value to make decisions.
Registered Radio Reception (6.4)	AODV uses less of the airtime resource.	Yes, this is a good value to make decisions.
Time to Design (6.5)	It has cost some time to adapt AODV code for the test app. The broadcast loop prevention mechanism was set up much faster.	Yes, this gives you a hint whether you are constructing a rather exlusive or inclusive system.

7 Conclusion

So a communication platform is designed and even implemented in an simulator with an high grade of detail. [GVAB] which discusses the use and the errors on global variables states "A toy example is unlikely to demonstrate any problem with globals." So it can be concluded that programs which suffer from global variables are no toys – so are all network simulators which run code for the individual nodes, so is this thesis' simulator. ;-)

We found that AODV is a good routing algorithm which performs similar as simple broadcasting with respect to Packet Reception but uses less portions of the spectrum. As downside it is more difficult to implement.

All in all, the ALOHA-ACA medium access has to be improved in order to allow higher data rates to be transmitted. Maybe it is sufficient to simply shorten the waiting times in communication platform application.

For measuring sending intensity a new procedure called "Registered Radio Reception" is proposed which sums up the received radiation. It is further found that packet loss and Registered Radio Reception are good measures to form a benchmark.

7.1 Outlook

There's lot to do in different grades of complexity:

The data dispensation mechanism could be better, which would solve the problem with the difficult comparability of AODV and broadcast packet loss numbers. A packet loss detection and packet retransmission mechanism could be implemented. Noise (random waveblip changing) should be implemented. Bit errors could be evaluated, but for this you need better AMR frame mappings.

Of course when having implemented the communication platform on real hardware it then should be tested in real Critical Mass events.

An interesting task is to make it all run on real hardware and then check many cities for their attenuation formula cœfficients which in turn could be used in simulator so that it dœsn't depend on 12 measured points in one street any more.

Most complex task is to live peaceful together.

Literature

This section contains the references with regard to contents in alphabetical order.

[3GPP] 3rd Generation Partnership Project: Adaptive Multi-Rate (AMR) speech codec frame structure (Release 14). Technical Specification Group Services and System Aspects, 2017

[AODV] C. Perkins, E. Belding-Royer, S. Das: RFC 3561 - Ad hoc On-Demand Distance Vector (AODV) Routing. Experimental Draft. July 2003

[AODV-UU] E. Nordström, B. Wiberg, H. Lundgren: AODV implementation developed at Uppsala University, Sweden. https://github.com/erimatnor/aodv-uu (last commit on 13th April 2011)

[BNETZA] Bundesnetzagentur: Frequenzplan, Stand April 2016. https://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/Telekommunikation/Unternehmen_Institutionen/Frequenzen/Frequenzplan.pdf?__blob=publicationFile&v=9 (page checked in July 2017)

[BURGERS] S. Xue, B. Jia, R. Jiang, X. Li, J. Shan: An improved Burgers cellular automaton model for bicycle flow. Physica A journal, 2017

[CPU] Essential Specifications of Intel® Core™ i5-7200U Prozessor. https://ark.intel.com/de/products/95443/Intel-Core-i5-7200U-Processor-3M-Cache-up-to-3_10-GHz (page checked 1st June 2018)

[DSUE] Heinrich Nuszkowski: Digitale Signalübertragung. Vogt 4 ed. 2015

[DYNAMICS] R. Jiang, M.-B. Hu, Q.-S. Wu, W.-G. Song: Traffic Dynamics of Bicycle Flow: Experiment and Modeling. Transportation Science 51(3):998-1008, 2017

[GVAB] Global Variables Are Bad. http://wiki.c2.com/?GlobalVariablesAreBad (page checked 1th June 2018)

[HAMANN] C.-J. Hamann: Grundlagen der Schedulingtheorie. Lecture in summer term 2014 at TU Dresden, Faculty for Informatics

[IDM] V. Kurtc and M. Treiber: Simulating bicycle traffic by the Intelligent-Driver Model – reproducing the traffic-wave characteristics observed in a bicycle-following experiment. arXiv:1805.09592 [physics.soc-ph] Mai 2018

[IRFP] J. S. Seybold: Introduction to RF Propagation. Wiley 10 ed. 2009

[LANDUSE] T. Kürner, D. J. Cichon, W. Wiesbeck: The Influence of Land Usage on UHF Wave Propagation in the Receiver Near Range. IEEE Transactions on Vehicular Technology, August 1997

[MAN] Man pages project, version 4.16: man 3 rand, 13th June 2017

[METRICS] Spectrum Efficiency Metrics: A New Report. Statement on landing page https://its.bldrdoc.gov/, seen 3rd June 2018

[MERZIGER] G. Merziger, G. Mühlbach, D. Wille, and T. Wirth: Formeln und Hilfen zur Höheren Mathematik. Binomi Verlag, 2010 6 ed. ISBN 978-3-923 923-36-6.

[NC2400] neo.cortec: Integration Manual for NCxxxx series Modules. neocortec.com. Version 2.4 Sept. 2017

[OXYGEN] H. J. Liebe, G. A. Hufford, R. O. DeBolt: The Atmospheric 60-GHz Oxygen Spectrum: Modeling and Laboratory Measurements. March 1991

[PHYSIK] G. Joos: Lehrbuch der theoretischen Physik. Akademische Verlags-Gesellschaft Leipzig, 1943

[PLNC] K.D. Irianto: Physical Layer Network Coding. Presentation in winter term 2017/2018 at TU Dresden, Faculty of Electrical Engineering

[RFM69] Hoperf electronic: RFM69HCW ISM Transceiver Module. Datasheet (c) 2006

[RPM] Á. Milánkovich, K. Lendvai, S. Imre, S. Szabó: Radio Propagation Modeling on 433 MHz. 18th European Conference on Information and Communications Technologies (EUNICE), Aug 2012

[RPRO] C. A. Levis, J. T. Johnson, F. L. Teixeira: Radiowave Propagation. Wiley 10 ed. 2009

[SO] Answer by “Rudd Zwolinski” to question “Why dœs printf not flush after the call unless a newline is in the format string?” https://stackoverflow.com/a/1716621 (page checked 1st June 2018)

[SOS] Statements. https://www.sos-save-our-spectrum.org/statements/ (page checked 3rd June 2018)

[SOS-2] Alles in Allem finde ich es ziemlich unübersichtlich [...] https://www.sos-save-our-spectrum.org/alles-in-allem-finde-ich-es-ziemlich-unuebersichtlich-welche-frequenzen-ich-in-welchem-land-benutzen-darf/ (page checked 3rd Juni 2018)

[SUPERWAVES] Superposition of Waves. http://www.acs.psu.edu/drussell/demos/superposition/superposition.html (last checked 2th June 2018)

[W-ALOHA] Reservation ALOHA. https://en.wikipedia.org/wiki/Reservation_ALOHA (page checked 5th June 2018)

[W-DBM] dBm. https://en.wikipedia.org/wiki/DBm (page checked 4th June 2018)

[WHITE] I. Budzinski, W. Kühling: Mobilfunkfreie "Weiße Zonen" – irreal oder rechtlich geboten? Neue Zeitschrift für Verwaltungsrecht, October 2015

Appendices

A AMR frame reception graphs

An overview of received AMR frames is provided by following table. For broadcast usually graphs for stations W and D are available, AODV data is only available for station W:

Routing method	Receiver threshold	Arrival counts	Selected detailed frame arrival
AODV	-20 dBm	W: Fig. 6.2	
broadcast	-20 dBm	W: Fig. 6.3, D: Fig. A.4	W: Fig. A.2, D: Fig. A.3
AODV	-100 dBm	W: Fig. 6.5	W: Fig. 6.4
broadcast	-100 dBm	W: Fig. 6.7, D: Fig. 6.8	W: Fig. 6.6, D: Fig. A.1
aodv	-1000 dBm	W: Fig. A.5	
broadcast	-1000 dBm	W: Fig. A.6	

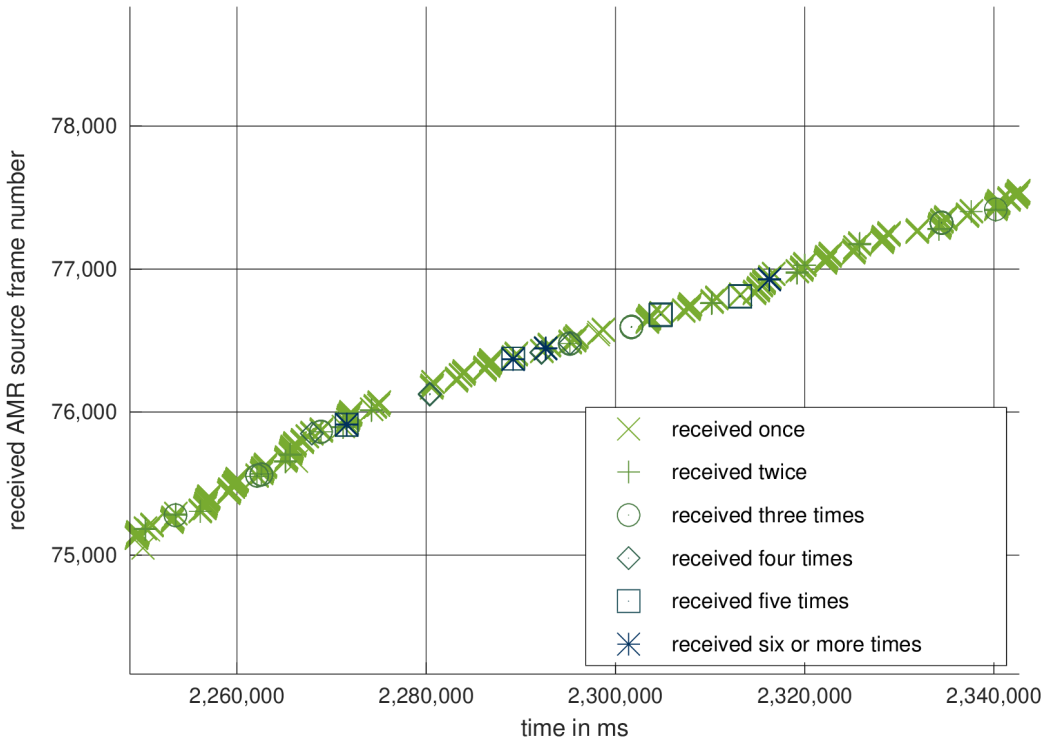


*Figure A.1: Received AMR frames at station **D** in **broadcast** mode at **-100 dBm**. (Selected interval)*

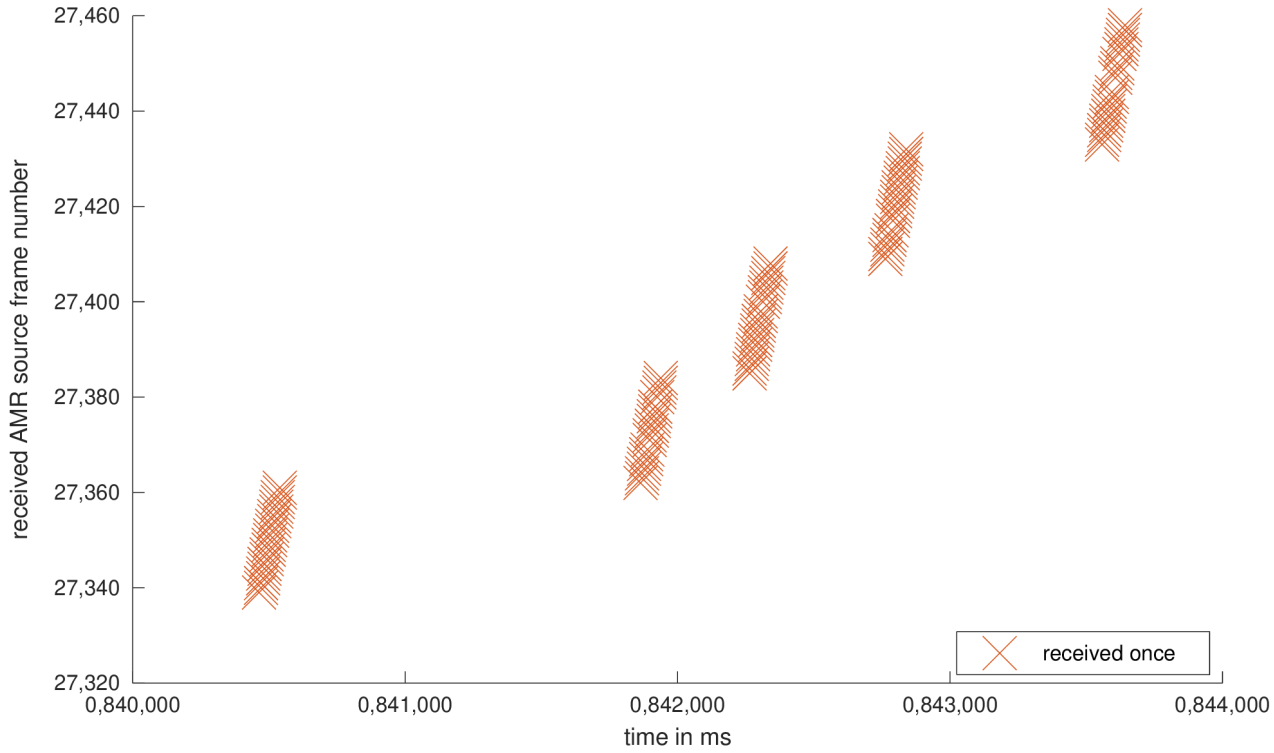


*Figure A.2: All rceived AMR frames at station **W** in **broadcast** mode at -**20 dBm** threshold.*

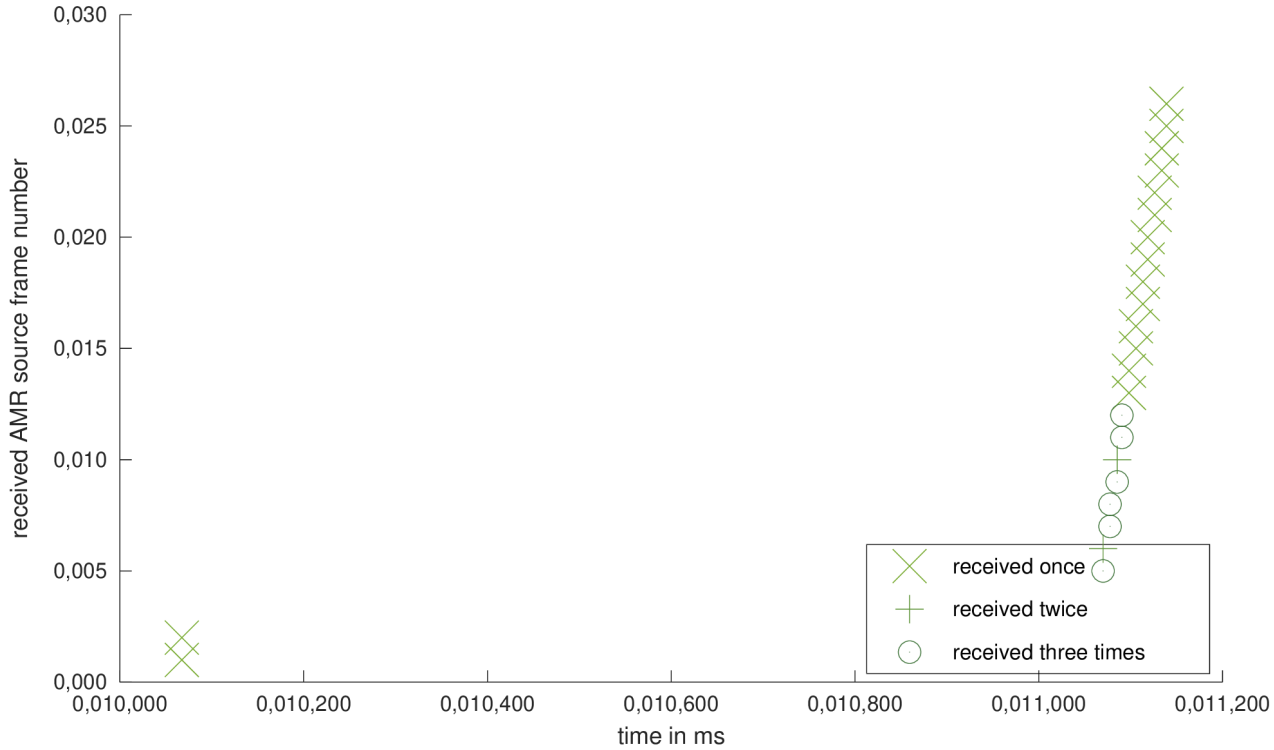


*Figure A.3: All rceived AMR frames at station **D** in **broadcast** mode at -**20 dBm** threshold.*

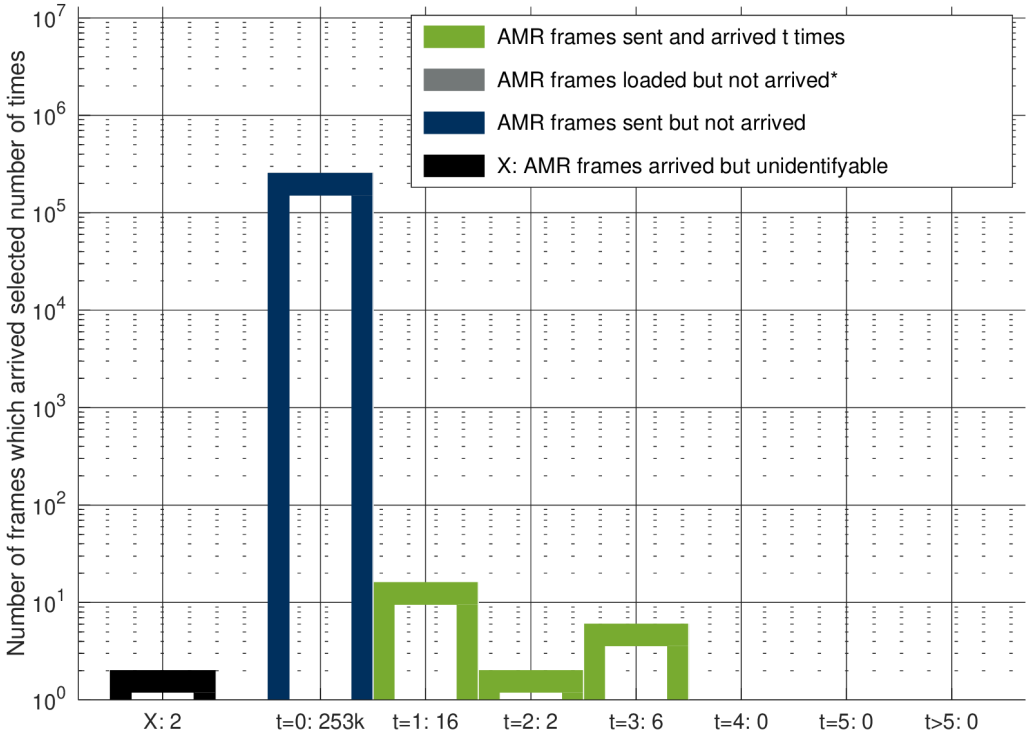


*Figure A.4: Arrival count of sent AMR speech frames at station **D** using **broadcasting** with receiver threshold of -**20 dBm**. Logarithmic plot.*

** like sent frames → "loaded" bar hidden by "sent" bar*

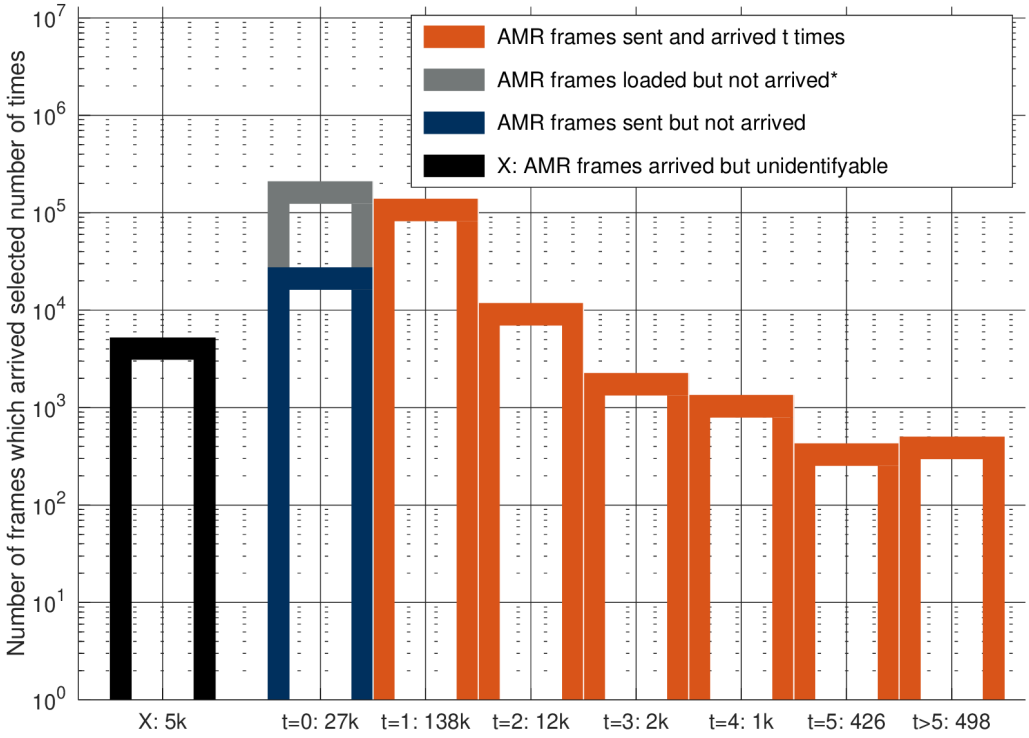


*Figure A.5: Arrival count of sent AMR speech frames at station **W** using **AODV** with receiver threshold of **-1000 dBm**. Logarithmic plot.*

** 208-thousend frames*

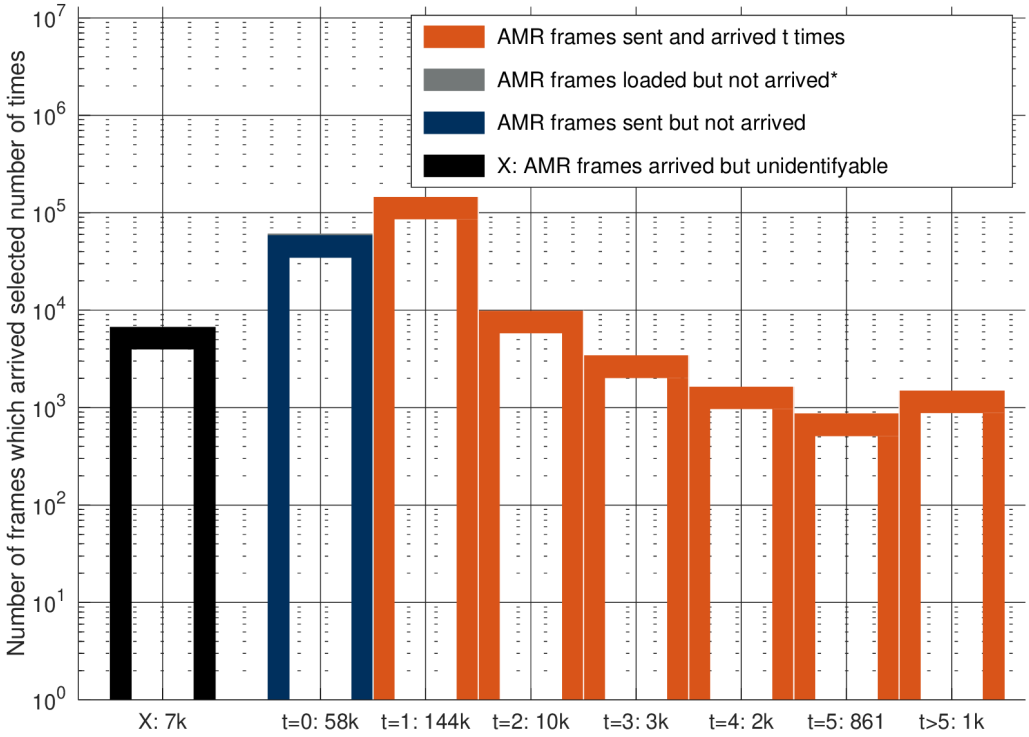


*Figure A.6: Arrival count of sent AMR speech frames at station **W** using **broadcasting** with receiver threshold of **-1000 dBm**. Logarithmic plot.*

** like sent frames → "loaded" bar hidden by "sent" bar*

B Full AMR corruption result chart

Here's the full chart of sound quality degradation on predefined bit affection explanations follow on next page.

Table C.1: sound quality impact of bit errors to AMR speech data

actual affected bits			**10 %**	**20 %**	**40 %**	**50 %**	**80 %**	**100 %**
equivalent affected bits			**20 %**	**40 %**	**80 %**	**100 %**	**160 %**	**200 %**
Mode		set	set20	set40	set80	set100		
		clear	clear20	clear40	clear80			
		flip		flip20	flip40		flip80	flip100
		rand	rand20	rand40	rand80	rand100		
Stop announcements with few train noise (regional train)	100% fine		p, (v)					
	Class A	set	p, W, n, B	0, x	0, x	ff, 0, x		
		clear	p, W, n, B	pp, 0, x	pp, 0	pp, 0		
		flip		p, W, 0-n, B	pp, 0, x		p, 0, B	p-pp, 0, x
		rand	p, W, n, B	p, 0, x	p, 0, x	p, 0, x		
	Class B	set	p, S, n-v, b	p, S, n-v, b	ff,S&f,n-v,B	ff,x → pp,0		
		clear	p, S, n-v, b	p, S, n-v, B	pp, S, n-v, B	pp, W, n, B		
		flip		p, S, n-v, b	p, S, n-v, B		p, S, n-v, B	p, W, n-v, B
		rand	p, S, n-v, b	p, S, n-v, b	p, S, n-v, B	p, W, n-v, B		
Stop announcements with train noise (suburban railway)	100% fine		v					
	Class A	set	W, n, B, x	f, 0, x	f, 0, x	ff, 0, x		
		clear	p, W, n-v, x	pp, 0, x	pp, 0	pp, 0		
		flip		p, W, 0-n, B	p, 0, x		pp, 0, x	pp, 0, x
		rand	p, W, n, B	p, 0, x	p, 0, x	p, 0, x		
	Class B	set	W, n-v, b	W, n-v, b	f, W, n-v, B	ff, W, n-v, B		
		clear	W, v, b	p, W, v, B	pp, W, v, B	pp, W, n-v, B		
		flip		W, v, b	W, v, B		p, W, n-v, B	p, W, n-v, B
		rand	W, n-v, b	W, n-v, b	p, W, n-v, B	p, W, n-v, B		
Music: Madonna eighties song "Who's That Girl"	100% fine		n-v, b					
	Class A	set	T, 0-n, B, x	0, x	f, 0, x	ff, 0, x		
		clear	0-n, B	pp,T,0-n,B,x	pp, 0	pp, 0		
		flip		n, B	p, 0, x		pp, 0, x	pp, 0, x
		rand	T, n, B	p, 0-n, B, x	pp, 0, x	pp, 0, x		
	Class B	set	n-v, B	f, t, n, B	f, t, n, B	ff, T, 0-n, B,x		
		clear	n-v, B	n-v, B	pp, t, n, B, x	pp,T,0-n,B, x		
		flip		n-v, B	T, n, B		pp, T, 0-n, B	pp, T, 0-n, B
		rand	n-v, B	t, n, B	T, n, B, x	T, 0-n, B, x		

Here's the legend for last pages' table:

pp: all in all very quiet
p: all in all quiet
f: all in all loud
ff: all in all very loud

v: content can be understood
n: you understand it if you know what's the content
0: content not recognizable

S: strange tones in gong
W: wrong tones in gong
t: time not fully recognizable (song)
T: time not at all recognizable (song)

b: bubbling and other side noice related to content
B: like b but stronger
x: independent side noice

silence: very quiet crackling

The affected bit were chosen by random at the creation of the scrambling program. The randxxx modes do chose which bits out of the preset potentially affected bits were flipped. All frames were affected in the same places.

"Equivalent affected bits" is the number of bits which must be randomly affected (so 50% get affected and 50% remain same) to achieve the actual flipped bit number.

C Envelope Change Check

This appendix shows which way the wave addition procedure checks for the resulting wave amplitude change. (Used in chapter 5.2 Electromagnetic Wave Handling)

If amplitude change is low following condition dœs hold:

$E(t-\Delta t) \geq 0.9\,E(t)$ (a) in raising case and

$E(t+\Delta t) \geq 0.9 \cdot E(t)$ (b) falling case

with approximated envelope function of the result $E(t) = a_1 + a_2 \cdot \sin(\Delta f \cdot t + p)$ (c) (see Fig. 2.1 on page 6). Raising case works analog to falling case and is ignored here.

If the condition is true it means that the situation throughout the waveblip duration is rather constant and the two waves can be mixed as if their frequencies were equal.

The approximated envelope function is a cosine/sine function which changes most in the nulls. (Meaning the nulls of the original cosine function before being shifted along y axis). It is assumed that “worst case” happens so following calculation is carried out with $t = 0$ and $p = 0$. For more accurate simulation the actual phase could be taken into account.

So finally condition is determined with: (d)

$$
\begin{array}{lcll}
E(t-\Delta t) & \geq & 0.9 \cdot E(t) & \mid\ t = 0 \ \text{ w.l.o.g.} \\
E(0-\Delta t) & \geq & 0.9 \cdot E(0) & \mid\ E(t) = a_1 + a_2 \cdot \sin(\Delta f \cdot t + p) \\
a_1 + a_2 \cdot \sin(\Delta f \cdot (0-\Delta t) + p) & \geq & 0.9\,a_1 + 0.9\,a_2 \cdot \sin(\Delta f \cdot 0 + p) & \mid\ -0.9\,a_1 \quad \mid\ p = 0 \ \text{ w.l.o.g.} \\
0.1\,a_1 + a_2 \sin(-\Delta f \cdot \Delta t) & \geq & 0 & \mid\ -a_2 \sin(-\Delta f \cdot \Delta t) \\
0.1\,a_1 & \geq & -a_2 \sin(-\Delta f \cdot \Delta t) & \mid\ : a_2 \\
0.1\,\dfrac{a_1}{a_2} & \geq & \sin(\Delta f \cdot \Delta t) & \mid\ \sin(f \cdot x) \approx f \cdot x \\
0.1\,\dfrac{a_1}{a_2} & \geq & \Delta f \cdot \Delta t & \mid
\end{array}
$$

The actual implementation dœs assume $a_1 \approx 3 \cdot a_2$ in order to use fewer variables in calculation.