Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Blaž Repas

Napredni programsko definirani sprejemnik za sporočila ADS-B

MAGISTRSKO DELO

MAGISTRSKI PROGRAM DRUGE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: izr. prof. dr. Fabio Ricciato

Ljubljana, 2017

University of Ljubljana
Faculty of Computer and Information Science

Blaž Repas

Advanced receiver for ADS-B messages in Software-Defined Radio

MASTER'S THESIS

THE 2nd CYCLE MASTER'S STUDY PROGRAMME
COMPUTER AND INFORMATION SCIENCE

Mentor: Assoc. Prof. PhD. Fabio Ricciato

Ljubljana, 2017

I would like to thank everyone that contributed or helped make this thesis a reality. Special thank you goes to my mentor, Fabio Ricciato, who has inspired my research spirit and got me interested in the subject. I would also like to thank him for his guidance, patience, attention to detail and loads of effort invested into our joint research endeavours. Thank you!

Blaž Repas, 2017

Contents

List of used acronyms

acronym	meaning
SDR	Software-Defined Radio
ADS-B	Automatic Dependent Surveillance - Broadcast
ATC	Air Traffic Control
SNR	Signal-to-Noise Ratio
ToA	Time-of-Arrival
TDoA	Time-Difference-of-Arrival
ADC	Analog-to-digital converter

Povzetek

Naslov: Napredni programsko definirani sprejemnik za sporočila ADS-B

V tem magistrskem delu sta predstavljeni zasnova in implementacija naprednega programsko definiranega sprejemnika za sporočila ADS-B/Mode-S, ki jih oddajajo plovila v letalskem prometu za namene kontrole zračnega prometa. Predstavljena in argumentirana je zasnova programske procesne verige za sprejem signalov, ki poleg osnovnega dekodiranja paketov služi tudi kot platforma za nadgradnjo in razvoj naprednih funkcij. Prav to je s pridom uporabljeno za razvoj in vrednotenje metode za določanje časa prihodov paketov, kar je ključen podatek pri pasivnem določanju položaja in sledenju letečih plovil. V tej magistrski nalogi je predstavljenih in ovrednotenih več metod za določanje časa prihodov paketov vključno z izvirnim pristopom, ki upošteva položaj vrhov v signalu ter dosega časovno natančnost tudi do $1,2$ nanosekunde. V nadaljevanju je predstavljena metoda za izboljšavo dekodiranja z razreševanjem kolizij paketov. Le-ta deluje na podlagi tehnike izničevanja signala paketa, ki je bila razvita v sklopu tega magistrskega dela. Delo predstavi tudi potencialne izboljšave, ki bi v prihodnosti omogočile razvoj kognitivnega sprejemnika.

Ključne besede

Programsko definirani radio, ADS-B, nadzor letalskega prometa, sprejemnik

Abstract

Title: Advanced receiver for ADS-B messages in Software-Defined Radio

This master thesis presents the work of designing and implementing an advanced software-defined radio receiver for ADS-B/Mode-S messages, broadcasted from aircraft for the use in air traffic control. It presents and argues the design of the software receiver chain and introduces a platform that enables augmenting the basic receiver chain with advanced features. Furthermore, this work leverages the software receiver possibilities to implement and evaluate packet timestamping techniques that produce packet time-of-arrival information necessary for passive position tracking of airplanes. Multiple timestamping techniques are implemented and evaluated including a novel peak-position based approach that achieves precision of impressive 1.2 nanoseconds. Moreover, this thesis produces a method for improving packet decoding by means of packet collision resolution. Proof of concept collision resolution is achieved by facilitating the packet signal cancellation method developed in this work. This thesis also presents improvements for the future that would allow the developed receiver to become cognitive.

Keywords

Software Defined Radio, ADS-B, Airplane Surveillance, Receiver

Razširjeni povzetek

V zračnem prometu je v uporabi veliko tehnologij, ki so namenjene zagotavljanju varnosti in preglednosti zračnega prometa. Ena izmed tehnologij je tudi sistem Mode-S za izmenjavo sporočil med kontrolo letenja in letali. V sklopu tega sistema letala periodično pošiljajo podatke o svoji geografski legi, hitrosti, višini in drugih pomembnih parametrih. To je sistem ADS-B (angl. automatic dependent surveillance - broadcast) oziroma avtomatski sistem za nadzor, ki je odvisen od delovanja oddajnika na letalu in podatke pošilja periodično ter brez zahtevka.

V sklopu tega magistrskega dela smo izdelali programsko definirani sprejemnik za sporočila ADS-B/Mode-S, ki temelji na obstoječem projektu *gr-air-modes*[1] in je izdelan v ogrodju za digitalno procesiranje signalov in programsko definirani radio - *GNU Radio*[2]. To nam omogoča fleksibilnost in uporabo vnaprej pripravljenih blokov in funkcij za obdelavo signalov, prav tako pa s tem omogočimo podporo za večino strojnih vmesnikov, ki se uporabljajo na področju programsko definiranega radia. Podrobno smo opisali arhitekturo in načrt našega sprejemnika. Programski sprejemnik je sestavljen iz več delov, ki so potrebni za delovanje. Najprej signal sprejeme vmesnik SDR (vmesnik za programsko definirani radio), ki sprejet signal iz antene pretvori v zaporedje digitalnih vzorcev. Od te točke naprej poteka vsa obdelava signalov znotraj programske opreme. Tok vzorcev najprej potuje skozi nizkoprepustni filter in fazo povišanja vzorčenja (angl. up-sampling), nato pa v blok za zaznavanje energije, ki tok oz. kos vzorcev prepušča le, če je signal dovolj močan. Nato sledi zaznavanje preambule paketa, kjer se preveri, ali

imamo v obdelavi kos vzorcev, ki vsebuje paket ADS-B/Mode-S. V primeru, da je preambula prisotna, se prične dekodiranje vsebine paketa. To poteka po pravilih modulacije s pozicijo pulza (angl. PPM = Pulse-Position Modulation). Ob uspešnem dekodiranju (pravilna kontrolna vsota) se vsebina paketa izpiše, prav tako pa je pripravljena za nadaljnjo obdelavo, skupaj s surovimi vzorci, ki pripadajo paketu. To služi kot podporna platforma za razvoj bolj naprednih funkcij, ki so opisane v nadaljevanju. V delu so predstavljene tudi potencialne izboljšave, ki bi v prihodnosti omogočile razvoj kognitivnega sprejemnika.

Poleg razvoja sprejemnika smo v delu prestavili tudi strojno okolje, ki je služilo zajem signalov in njihovo shranjevanje v datoteke. Uporabili smo posebej prilagojeno anteno za sprejem signalov na frekvenci 1090 MHz. Signal smo filtrirali in ga nato z razdelilnikom pripeljali do dveh identičnih strojnih vmesnikov za programsko definirani radio. Poleg zajema pa smo za naš sprejemnik ovrednotili tudi zmožnost dekodiranja paketov in meritve primerjali s tistimi za odprtokodni sprejemnik *dump1090*. Pokazali smo, da naš sprejemnik deluje bolje oziroma vsaj tako dobro kot *dump1090*.

Ena izmed glavnih naprednih funkcij v sprejemniku je sistem za natančno določanje časa prihoda paketa. Časovne značke paketov so zelo uporabni podatki, saj jih lahko s pomočjo triangulacijskih algoritmov uporabljamo za pasivno določanje geografske lege plovila [3]. S tem lahko lego plovila določamo zgolj s sprejemom signala in določanjem časovne značke paketa (na več sprejemnikih). To je uporabno tudi za preverjanje lege, ki jo v sporočilih pošilja plovilo. Pomembno je tudi, da iz plovila posredovana lega omejena z natančnostjo GPS sprejemnika na plovilu, pasivno določanje lege iz signala pa lahko celo doseže boljšo natančnost, kar uporabljajo tudi komercialni sistemi kontrole letenja, predvsem v bližini letališč (članek [4]). Natančnost določanja časovnih značk je torej za pasivno določanje in preverjanje lege izrednega pomena saj se direktno preslika v natančnost določanja geografske lege. Na tem mestu je potrebno poudariti, da sistema ADS-B in Mode-S[1]

[1] Sistema ADS-B in Mode-S uporabljata enak fizični format signala in modulacijsko

nista bila načrtovana za namen natančnega določanja časa prihoda paketov. Prav nasprotno - zaradi specifikacije [5], ki je nastala pred nekaj desetletji so zahteve relativno ohlapne. Na primer, posamezni pulzi v signalu lahko od nominalne pozicije odstopajo za ± 50 nanosekund, cilj našega dela pa je razvoj metode, ki bi dosegala natančnost v velikostnem redu nekaj nanosekund. Izziv predstavlja tudi dejstvo, da sisteme ADS-B in Mode-S izdeluje več različnih proizvajalcev, ki so na tržišču že desetletja. Zaradi tega imamo opravka z opazno različnimi oblikami signalnih pulzov, čeprav vsi ustrezajo specifikaciji.

Za natančno določanje časovnih značk oz. časov prihodov paketov smo razvili in implementirali več metod za določanje časovnih značk. Razvili smo več različic metode, ki temelji na korelaciji amplitudnega signala z znano predlogo (angl. template). To je kanoničen pristop k določanju časovnih značk, vendar pa v našem primeru ni deloval dovolj dobro, kar gre pripisati relativno ohlapnim tolerancam v specifikaciji. Da bi dosegli boljšo natančnost, smo uporabili izviren pristop in razvili metodo za določanje časovnih značk, ki deluje na podlagi položaja vrhov v amplitudnem signalu. Časovna značka je določena s pozicijo prvega vrha v amplitudnem signalu in s pomočjo povprečnega odstopanja dejanskih pozicij vrhov od nominalnih. S to metodo smo dosegli varianco napake v vrednosti $2,2$ nanosekunde, kar bi ustrezalo oceni geografske lege na 66 cm natančno. Z upoštevanjem jakosti signala paketa pa smo metodo še izboljšali in v posebnem primeru znižali varianco napake določanja časovnih značk na $1,2$ nanosekunde, kar ustreza oceni geografske lege na 36 cm natančno. Menimo, da to predstavlja presenetljivo dober rezultat glede na enostavnost in elegantnost pristopa, ki je tudi računsko učinkovit in primeren za implementacijo na napravah z omejeno računsko močjo. V delu so predstavljeni rezultati za vse različice razvitih metod, prav tako pa tudi primerjava z obstoječimi programsko definiranimi sprejemniki za sporočila ADS-B/Mode-S. Rezultati so poglobljeno obravnavani, prav tako pa so predstavljeni vplivi povišanja vzorčenja (angl. up-sampling) ter možna

shemo

razlaga za slabše delovanje metod, ki temeljijo na korelaciji. Delo vsebuje tudi metodo za ocenjevanje variance napak časovnih značk, ki deluje na realnih podatkih, kjer ne poznamo dejanskih absolutnih časovnih značk oziroma jih niti ni mogoče določiti.

Ena izmed glavnih prednosti programsko definiranega sprejemnika je zmožnost poganjanja algoritmov v več prehodih, prav tako pa lahko procesiranje izvajamo le na specifičnih delih signala. To je ključna prednost, ki nam je omogočila razvoj metode za izboljšavo dekodiranja z razreševanjem kolizij paketov. Kolizija nastane, ko dva različna oddajnika (dve različni letali) oddajata signal istočasno oziroma kadar se signala oddanih paketov (delno) prekrivata. Ponavadi sta v tem primeru izgubljena oba paketa, vendar pa je možno tudi, da je sprejemnik pravilno sprejel enega (ponavadi močnejšega) izmed paketov. V tem primeru lahko s pomočjo metode za izničevanje signala paketa (angl. packet signal cancellation) odstranimo signal paketa, ki smo ga uspešno sprejeli in s tem razkrijemo signal drugega paketa, ki bi sicer bil izgubljen. V tem magistrskem delu smo opisali in implementirali metodo za izničevanje signala, ki za delovanje potrebuje dobre približke fizičnih lastnosti signala, kot so amplituda, faza, frekvenčni zamik, pozicije vrhov amplitudnega signala (ki jih dobimo iz faze določanja časovnih značk) ter vsebina paketa. Predstavljeni so postopki za izluščevanje omenjenih lastnosti, ter postopek ustvarjanja sintetičnega signala paketa, ki čim bolj ustreza lastnostim izvornega signala. S tem lahko ustvarimo signal enega izmed paketov, ki ga nato odštejemo, da izničimo signal prvega paketa in izpostavimo signal drugega paketa, ki ga je nato mogoče dekodirati. Pokazali smo, da postopek izničevanja signalov deluje na realnih primerih, vendar pa skupen postopek razreševanja kolizij dobro deluje le na sintetično ustvarjenih kolizijah. Predstavili smo težave in potencialne izboljšave, ki bi omogočale dobro razreševanje kolizij tudi kadar ne poznamo vsebine niti enega izmed paketov.

V tem magistrskem delu je torej predstavljen napredni programsko definirani sprejemnik za sporočila ADS-B in Mode-S, ki vsebuje napredne funkcije kot sta natančno določanje časov prihodov paketov (pomembnih vho-

dnih podatkov v algoritme za pasivno določanje geografske lege plovil) in razreševanje kolizij s pomočjo metode izničevanja signalov. Ugotovili smo, da je razreševanje kolizij kompleksen problem, ki odpira možnosti za nadaljnje raziskave, rezultati evalvacije metod za določanje časovnih značk pa so pokazali, da lahko z razvito metodo določimo časovne značke tudi do $1,2$ nanosekunde natančno (ekvivalentno 36 cm s hitrostjo svetlobe). Predvsem pa je rezultat tega magistrskega dela implementacija sprejemnika, ki omogoča nadaljnji razvoj naprednih funkcij in služi kot raziskovalna platforma. Poleg tega pa smo predstavili tudi potencialne izboljšave in vizijo za prihodno raziskovalno delo, ki bi omogočilo razvoj kognitivnega sprejemnika za sporočila ADS-B in Mode-S.

Chapter 1

Introduction

The widespread availability of inexpensive software defined radio equipment has led the research in the field of radio communications to also become a computer science topic. The processing of radio signals can be done on general purpose computers with the help of software-defined radio interfaces. SDR interface handles physical signal conversions and hardware manipulations necessary to convert the radio signal into a digital form. SDR has been used in computer science research in many topics. There has been research in security [6, 7], distributed systems (for example, distributed spectrum monitoring described in [8]) . One of the more useful qualities of SDR is that most of the signal processing is done in software, thus allowing a quick turnaround time as a test-bed for developing and testing new protocols, such as described in [9, 10]. SDR has been also used in conjunction with artificial intelligence methods and machine learning to generate new modulation schemes and increase adaptability, shown in the article [11].

Since the beginning of air traffic, there was always a need to track and surveil each and every aircraft in order to assure safe and uninterrupted flow of air traffic. Several technologies were developed to make this possible, for example radar. There exist multiple radar technologies that are roughly divided into two categories: primary radar and secondary radar. Primary radar is the system that emits EM pulses and measures reflections - it is completely passive from the aircraft's point of view. Secondary radar, on the other hand, requires some cooperation from the aircraft. It initiates radio communication with the aircraft's on-board transponder and receives identity and flight information. Currently there are multiple secondary radar technologies in place. One that operates at the frequency of 1090 MHz is so called MODE-S [5]. MODE-S has two modes of operation: interrogation (request from ground and response from aircraft) and broadcast, where aircraft periodically broadcast unsolicited messages. This is covered by the ADS-B - Automated Dependent Surveillance - Broadcast protocol. Every aircraft equipped with an ADS-B transceiver sends out periodic updates of its location, velocity, altitude (GPS data) and other parameters of interest using radio waves. Traditionally, receivers for specific function were purpose-made, having an application specific hardware circuitry, or are implemented in semi-software fashion - with FPGAs, like for example the well known Radarcape receiver [12]. With wide availability of software defined-radio equipment, it became possible to implement a receiver for ADS-B in software together with inexpensive SDR interfaces/cards, like shown in the article [13]. Exploiting this fact, there are systems that collectively receive ADS-B signals in order to cover a large area. For example, FlightAware [14] or Flightradar24 [15] that are commercial, or a more research oriented Opensky Network [16].

Because of great availability of inexpensive SDR hardware, implementations of Mode-S/ADS-B receivers have emerged in the open-source communities as well. In fact, open-source ADS-B receiver implementations coupled together with inexpensive SDR hardware have produced an interesting outcome: crowdsourced reception, where individuals can setup ADS-B receivers to improve either the range or accuracy of the collective system (like Flightradar24 [15], or the Opensky Network [16]). That is why the work of this thesis is directly related to the most widespread open-source implementation of Mode-S/ADS-B receiver, the *dump1090* [17]. It sets the reference to which we will compare our work. Other projects include *gr-air-modes* [1] that implements the receiver in GNU Radio[2], the open-source software radio toolkit.

With any system pertaining radio reception, there is always the motivation for better reception. In case of MODE-S and ADS-B, better reception does not necessarily mean a more sensitive receiver, like it is the case with other analog or digital systems. Here we mostly have a good signal or no signal at all, since there is usually line-of-sight between the aircraft and the receiver and the aircraft transmit with very high power [5]. The problem here is that many aircraft have to share the same frequency band, which is limited and fixed (due to legacy support). When multiple transponders initiate transmission at the same time, the packets transmitted destructively merge - we say that the packets collide.

Another aspect of better reception in the case of ADS-B is precisely timing the arrival of packets. Similar to GPS and other positional systems, time information can be used to determine the position of the aircraft. By measuring the packet time-of-arrival at multiple receivers, we gain data that multilateration algorithms (MLAT) can transform into a positional estimate (shown in article [3]). The requirement of multiple timestamps for the same packet plays very nicely with the crowdsourced reception as multiple individual receivers can capture packets from the same aircraft. Given that we want very precise positional estimates, the time-of-arrival must be precise as well as the timestamping precision has direct impact on the position precision. Moreover, the paper [4] explains that multilateration is actually done in commercial air traffic control systems near airports, because the positional reports from aircraft are not accurate enough. This motivates research for timestamping methods that could achieve precision in the order of nanoseconds and thus positional precision in the order of 1 meter. Also considering the crowdsourced reception, we can use the positions determined from the timing information to verify the positional reports (GPS) that aircraft themselves are sending and detect attacks on the ATC systems [18, 19, 20]. Similarly, such opportunistic use of aircraft signals has been exploited also for indoor self-localization (show in article [21]) as a replacement for GPS.

Both aspects present certain challenges that we must face. Firstly, it is our goal that software and methods applied work well with inexpensive SDR devices, for example RTL-SDR dongles. These devices have neither instrument-grade clock precision and stability, nor the best radio front-end. Nonetheless, these kind of devices were chosen since they are low-cost and therefore suited for wide-scale deployment that is common in crowd based reception. The second challenge is a rather loose specification of the Mode-S signal format. Specifications are rather old and allow for relatively large tolerance margins, including peak temporal position and pulse shape (defined in [5]). For example, tens of nanosecond wide margins pose a problem when tying to achieve nanosecond-level resolution. Moreover, signals are generated by a very heterogeneous set of transponder models, some of which were deployed decades ago, and differences in signals generated are discernible enough that fingerprinting of transponder model (and subsequently aircraft type) is possible (article [20]) just by observing the shape of the signal. It is important to note that these appreciable differences still comply with the specifications tolerance levels. For the reasons stated above, the tasks of high-precision time-of-arrival timestamping and cancellation are more difficult to achieve comparing to more modern standards (for example, WiFi or LTE) with tightly constrained tolerance margins and where more canonical methods are applicable. There has been work on employing correlation with nominal pulse shape [4, 22] for Mode-S timestamping, however it would likely suffer when faced with different pulse shapes present in real data, yet still within the specifications. Therefore, our work focuses on developing an alternative approach that does not rely on a priori knowledge of the pulse shape, nor the exact pulse position.

1.1 Structure

Here we will briefly describe the contents, structure and goals of this thesis.

In chapter 2 we will present secondary radar technology MODE-S and

ADS-B, what are they used for and the relevance and importance of such technology. We will describe the physical properties of the radio signals used, such as operating frequency, signal bandwidth and modulation format. Moreover we will describe the basic data format and structure relevant for this work.

In chapter 3 we will present and describe the design and architecture of the implemented software receiver. We will argue about design choices that were made during the development. Moreover, we will describe each block of the receiver. We will present the structure that allows implementing other improvements that we will describe in the following chapters.

Chapter 4 describes the hardware and software setup for capturing real world data and preparing the evaluation environment. Furthermore, this chapter contains the evaluation of the decoding performance in comparison with the widespread receiver *dump1090*.

Chapter 5 will describe methods, techniques and evaluation of different methods of packet timestamping - determining the time of arrival for each packet. We will present several methods of packet timestamping and present the developed techniques of evaluation. Next we will present the results of the work on high precision timestamping.

Chapter 6 contains the work on packet collision resolution and interference cancellation. We will present packet cancellation techniques and the prerequisite estimation of signal properties.

Finally, we conclude this thesis in chapter 7 by briefly recapping the problem and concisely restate our work and present the most important results. Furthermore, we lay out possible future improvements and possible goals for future work and research.

1.2 Goals

The main goals of this thesis are to develop a working implementation of software receiver for ADS-B and Mode-S and use it as a research and development platform for developing and evaluating advanced features. Furthermore, the goals are to design, implement, and evaluate advanced features such as packet cancellation resolution and to implement high precision timestamping methods and test and confirm its performance on real data. This work is aiming to develop and implement an alternative timestamping method that will deliver nanosecond-level resolution of packet time of arrival. Furthermore, leveraging the details of temporal information, we will show a proof-of-concept implementation of Mode-S packet collision resolution technique that does not require expensive gear or multitude of radio receivers or antenna. Additionally, we will provide insight as to how current cancellation technique could be improved in the future iterations of the receiver. Another important design goal that was placed upon this work is that it should be able to work with inexpensive SDR hardware such as a RTL-SDR dongle, yet remain compatible with other SDR hardware that might become available in the future. Lastly, the principle goal of this thesis is not only to produce a working receiver, but to create an evolvable framework that can facilitate future research and allow easy implementation and evaluation of new advanced features and bring the receiver close to a cognitive one.

Chapter 2

MODE-S and ADS-B

Aviation uses different methods to track and surveil airborne aircraft in order to keep the airspace a safe and collision free environment. Traditionally and conventionally, primary surveillance radar is used to detect airborne objects. It uses a pulse of radio signals and measure the reflections to detect object's position though tracking also bearing. It is important to mention that primary radar does not need any sort of cooperation from the aircraft, in that sense it is passive.

Secondary surveillance radar, on the other hand, is an active approach. To be more specific, it requires that an aircraft is equipped with a specific hardware - a transponder. The transponder receives commands and interrogations and responds to them. In some cases the transponder broadcasts unsolicitedly - without a request from aircraft traffic control. For the scope of this thesis the two systems of interest are ADS-B and Mode-S. Normally the requests or interrogations are sent by the air traffic control by using the nominal frequency of 1030 MHz. Replies from the aircraft are sent out on 1090 MHz. This is the nominal operating frequency that is most relevant for our work.

Mode-S is a secondary surveillance radar system that is used for multiple purposes, in contrast to Mode-A or Mode-B that are used for identification. Mode-S is used to request telemetry or data, such as altitude, position, or for delivering messages. Transponders automatically respond to interrogations from ATC.

ADS-B or Automatic Dependent Surveillance - Broadcast is a specific subsystem of Mode-S transponders. Like the name suggests, it is an automatic system. That means it transmits without pilot intervention or traffic control requests. It periodically broadcasts altitude, airspeed and other telemetry.

2.1 Physical format and modulation

In order to decode Mode-S replies including ADS-B broadcast packets, we must firstly look at the physical format of the packet. Mode-S packets are modulated with pulse position modulation. Each bit takes 1 μs. In the time allocated for transmitting one bit, the pulse could occupy the earlier (left) or later (right) half of the timeslot encoding either 1 or 0. This is apparent on figure 2.1. This figure also shows the specific preamble format that distinguishes Mode-S reply packets from other systems that share the same frequency.

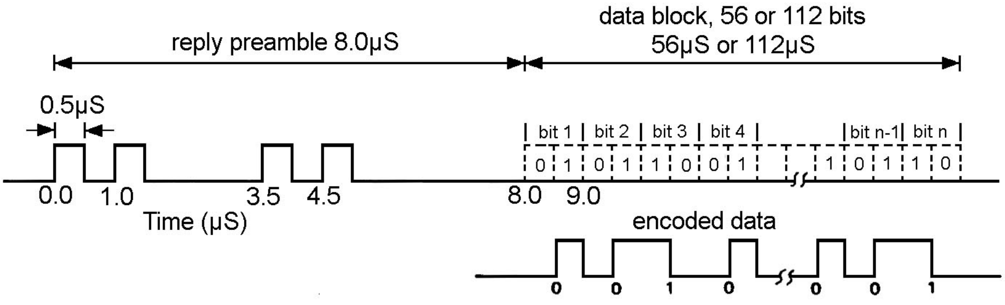


Figure 2.1: Mode-S reply packet structure

Our work encounters the modulation of Mode-S packet from two different perspectives. First and more conventional is the decoding of the packet payload data. This is the intended purpose of the format and what the specifications were written for. The other perspective is determining packet time-of-arrival (ToA), in other words timestamping. It is important to note that precise timestamping of Mode-S packets is an opportunistic application, in other words, support for high precision timestamping was not a part of the design and the specification. Furthermore, the specifications are several decades old and keeping in mind the intended purpose, allow for large tolerance margins. For example, the position of each pulse might be shifted by ±50 ns with respect to the nominal position. This poses quite a problem when nanosecond-level resolution is the aim of our work.

Chapter 3

Receiver design and implementation

This chapter will present the architecture, design and implementation of the software-defined radio receiver for Mode-S replies and ADS-B packets.

For better understanding of the following content we will briefly describe how signals are processed in software defined radio applications. There are at least two parts in every SDR system: SDR hardware and the software processing chain. SDR hardware or SDR interface card provides the interface to the physical world. SDR card receives radio-frequency signals through an antenna. It features a radio-frequency fronted that conditions the received signal and converts it into a digital representation. This is done with the analog-to-digital converter (ADC) that samples the signal with the appropriate sample rate and bit depth. One thing to note here is that rather than producing a single stream of samples, SDR cards produce two streams of samples: the ones that are in-phase and the ones that are in quadrature with the operating frequency, more commonly referred to as I/Q samples. This actually produces a stream of complex numbers (in-phase as real and quadrature as imaginary elements) that allow easier digital processing of the captured signal. Although the receiver presented in this thesis works in real-time/online with SDR hardware, most of the research was done offline: by capturing the samples into a file (also called a trace) and replying the file to the software as if it was a SDR device. The software part of SDR application is the following digital signal of the two streams of samples. This usually includes some signal conditioning (filtering, re-sampling), but is not limited only to simulating operations traditionally done in hardware. It can also do non-linear processing and is more or less limited only by program design.

We have chosen to implement the receiver using the open source radio and signal processing toolkit called GNU Radio [2]. This allowed us to use readily available components such as filters and common signal processing blocks. Due to the fact that there already exists an implementation of Mode-S/ADS-B receiver in GNU Radio, we have chosen not to implement our version from scratch but to base it on the project known *gr-air-modes* [1]. It served as a starting point to explore the working of such a receiver and to reuse components that are necessary but instrumental for our work. It would also be possible to base our work on top of the more popular *dump1090* project, however, dump1090 is only designed to work together with the inexpensive RTL-SDR dongle. Although the goal was to ultimately use the same SDR hardware, we pursued the goal of being compatible with other hardware as well. Here the modular approach of GNU Radio toolkit comes to play as it allows to just swap the hardware signal source module for one SDR card with another one. This has significant value as each year better or less expensive SDR hardware becomes available, allowing our receiver to be forward compatible.

3.1 Architecture and design

Our receiver features several blocks:

- Input (trace file or SDR device),
- DC blocker,
- Low-pass filter,

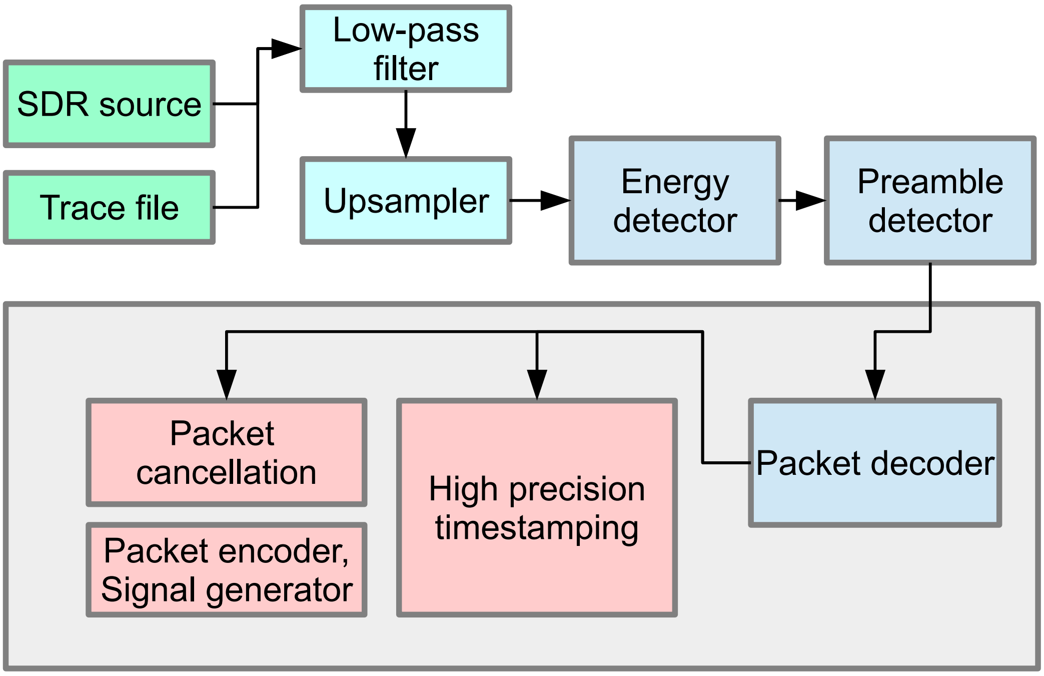
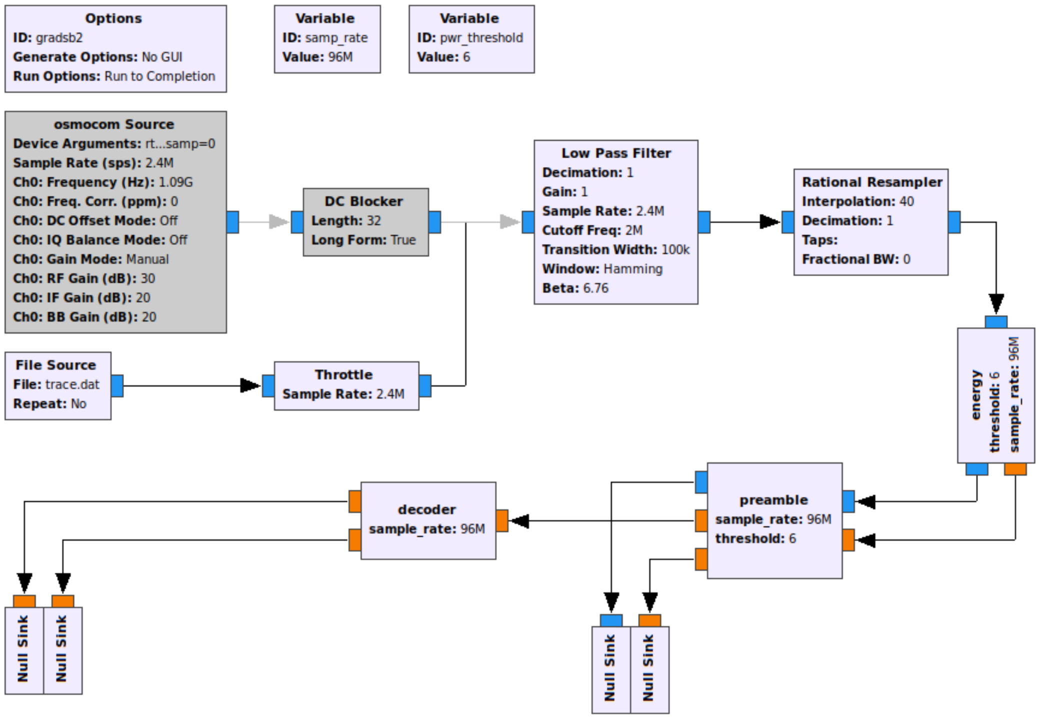


Figure 3.1: Schematic of the receiver chain

- Up-sampler,
- Energy detector,
- Preamble detector,
- Payload decoder,
- High precision timestamping module,
- Packet cancellation module.

Each of them will be described in its own section. The last two (high precision timestamping and packet cancellation) were the focus of this work and are thoroughly described, each in its own chapter.



Options
ID: gradsb2
Generate Options: No GUI
Run Options: Run to Completion

Variable
ID: samp_rate
Value: 96M

Variable
ID: pwr_threshold
Value: 6

osmocom Source
Device Arguments: rt...samp=0
Sample Rate (sps): 2.4M
Ch0: Frequency (Hz): 1.09G
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Off
Ch0: IQ Balance Mode: Off
Ch0: Gain Mode: Manual
Ch0: RF Gain (dB): 30
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20

DC Blocker
Length: 32
Long Form: True

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 2.4M
Cutoff Freq: 2M
Transition Width: 100k
Window: Hamming
Beta: 6.76

Rational Resampler
Interpolation: 40
Decimation: 1
Taps:
Fractional BW: 0

File Source
File: trace.dat
Repeat: No

Throttle
Sample Rate: 2.4M

energy
threshold: 6
sample_rate: 96M

preamble
sample_rate: 96M
threshold: 6

decoder
sample_rate: 96M

Figure 3.2: Our receiver implementation - GNU Radio chain

The signal enters the receiver chain either from a file (captured trace) or from a SDR device. Since we have implemented this receiver in GNU Radio toolkit, we have access to source blocks for multiple SDR devices, including RTL-SDR and somewhat de-facto standard - Ettus Research USRP devices.

3.1.1 Energy detector

The first block after signal conditioning (DC blocking, filtering) and upsampling is the energy detector. Since we are consuming a rather high sample rate signal and even upsampling it, it is important that we have an efficient method of deciding what possibly constitutes a signal and what we can ignore as the background noise. An effective way of limiting further processing is to to only forward samples when a certain signal-to-noise ratio has reached the threshold. This implicitly gives us a requirement: estimating the baseline noise level, so that the energy detector is able to determine the signal-to-noise ratio. The threshold is predetermined and acts as a trade-off between receiving weak signals and having lower processing power demands.

In our receiver, the baseline noise estimation is done by observing the $a - th$ percentile value of the amplitude. We have empirically found this value to be around 0.55, but can also be configurable as it depends on the type of hardware used and the noise profile of the environment.

3.1.2 Preamble detector

When the energy detector is triggered (SNR threshold surpassed), a chunk of samples is delivered down the chain and into the preamble detector. The role of preamble detector is somewhat similar to the role or energy detector. Both are intended to reduce the workload of the future stages and thus relieving some workload. Preamble detector decides whether or not the chunk of I/Q samples actually contain something resemblant of a Mode-S reply preamble.

Our receiver implements several criteria upon which it is decided if the chunk contains a Mode-S reply and should be processed further by trying to decode the payload:

- Amplitude ratio between the first peak and noise above threshold
- Amplitude ratio between off-peaks[1] and noise
- Amplitude ratio between preamble peaks and off-peaks above threshold

Most of these criteria were inherited from the original (*gr-air-modes* [1]) receiver. The most notable difference is that we have removed the temporal alignment by preamble correlation since we have found it unneeded for preamble detection and consumes more computational resources than required. The latter is especially important when the initial upsampling in the

[1] Off-peaks are areas, where the signal from the transmitter should not be present. See figure 2.1.

receiver chain is using a high upsampling factor. The alignment that was previously done by preamble correlation was pushed into the decoding phase where the signal is now aligned by the first preamble peak location and it has therefore no effect on decoding.

After a seemingly valid preamble has been detected, this module captures a chunk of samples (at least 2.5 packet lengths worth) and passes the chunk of samples to the next stage, the payload decoder. It also records the sample number (timestamp) of the start of the sample chunk, allowing further stages to use it as a reference when computing an actual packet timestamp.

3.1.3 Payload decoder

Payload decoder receives a timestamped chunk of samples from the preamble detector. The first operation is finding the peak of the first preamble pulse. The mechanism for finding the peak is the same as described later in chapter 5. The main difference is that a wider search interval is used to compensate for coarser alignment in the preamble detector.

Next, the aligned signal is sampled every 0.5μs to obtain chips for decoding. This produces a sting of chips - amplitude samples. Chips belonging to the preamble are ignored and the remaining payload chips are converted to bits by the rules of (B)PPM modulation described earlier. This decoder uses differential technique: it compares two chips that correspond to two timeslots that a pulse could occupy and compares the two. If the amplitude of the first chip is higher than the other, the bit value assigned is 1. Similarly, if the second chip is larger in amplitude, the decoder assigns a value of 0 to the pair of chips. This produces bit values that can be interpreted in terms of ADS-B or Mode-S message specifications. Furthermore, the CRC value is checked to verify if the packet was decoded successfully.

3.2 GNU Radio and implementation details

GNU Radio is an open-source toolkit for software radio. It provides infrastructure and facilities for software signal processing. First of all, it provides interface modules for access to SDR hardware (for example, RTL-SDR library and osmocom). Though these modules we gain access to the stream of I/Q samples generated by the SDR hardware device (radio front-end) and allows real-time processing of the said stream. Furthermore, GNU Radio provides pre-built blocks for common operations in signal processing. This includes filters, re-samplers and other signal conditioning operations normally found either in analog hardware circuits of digital signal processing chips. We have utilized these blocks for low-pass filtering (when source trace or SDR hardware is used with a bandwidth higher than 2.4MHz) and signal up-sampling.

Other modules in our receiver implementation, namely energy detector, preamble detector and packet decoder have been developed in C++ from scratch, while reusing parts from the existing *gr-air-modes* receiver. GNU Radio also provide support for integrating C++ code into the processing chain. This is done by leveraging the GNU Radio facility for out-of-tree modules. This allows users to write blocks in either python or C++ that can be used as blocks withing GNU Radio processing chain.

High-precision timestamping and packet cancellation modules have been developed as a C++ library. The reason behind this decision is that this advanced features can easily be used inside the packet decoder module or individually within a different implementation or a test-bed environment.

Last but not least, great care has been taken to ensure correct and accessible flow of raw I/Q samples that belong to a packet, such that the samples are readily available for timestamping and cancellation modules and that packet sample time is preserved correctly though the processing chain. In order to achieve this, we have utilized tagged streams (provided by GNU Radio) that allow augmenting the stream of samples with additional information and thus preserving the necessary data about the stream of samples corresponding to a packet.

3.3 Evolvability and future work

We have put considerable effort into making the receiver not only perform successful packet decoding of non-colliding packets, but also keep the design modular and ready for adding and implementing additional advanced features. To demonstrate this we have implemented two advanced features that augment the basic receiver:

- High precision packet time-of-arrival estimation,
- Packet collision resolution.

In order to successfully implement these features, we designed the receiver to be able to pass as much information from preamble detection and payload decoding phases onto successive block without loosing or distorting important timing information along the way. Furthermore, we implemented a transport system for delivering raw I/Q samples from decoding phases to subsequent consumer blocks, for example cancellation and high precision timestamping modules. This provides a framework for developing advanced features that rely on the decoded payload and/or raw I/Q samples of the signal.

It is our vision for the future to develop a cognitive receiver for Mode-S/ADS-B. Although not included in the receiver version developed for this thesis, we are conducting research and implementing features characteristic of a cognitive receiver. Specifically, we are developing a pair of modules that would together work as a cognitive transmitter profiling and recognition system. It would consist of two parts:

- transmitter profile cache,
- transmitter recognition module.

The proposed module (pictured on figure 3.3) would receive data about successfully decoded packets from the packet decoder module. Additionally, signal properties and other attainable data would be provided by advanced blocks like high precision timestamping and signal cancellation module. All

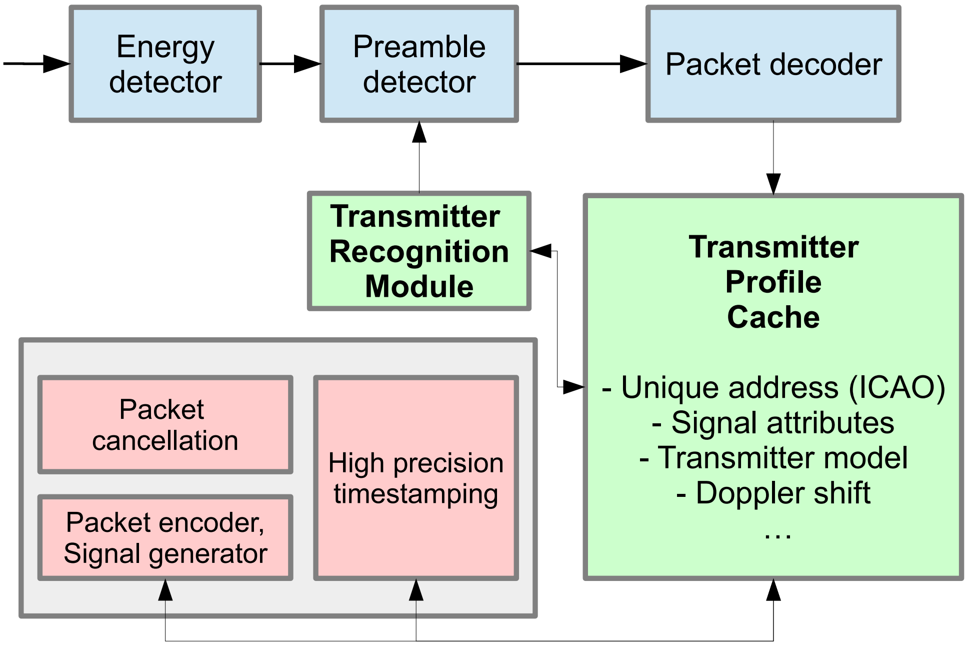


Figure 3.3: Schematic overview of a receiver equipped with transmitter profiling and recognition system.

of the data would be stored in a retrievable format that could be used in any of the modules up or downstream. For example, it could be accessed used when performing cancellation to select an appropriate signal model for a specific transmitter. Furthermore, the gathered data in the transmitter profile cache could be used in the transmitter recognition module to identify or recognize a particular aircraft solely from physical signal properties, like it has been demonstrated in [18] and [20]. This would also augment packet decoding because a part of the payload (notably aircraft unique address - ICAO) would be known in advance even before trying to decode the payload data. This would allow us to introduce iterative decoding techniques that would also report partially correct packets that are still valuable for passive aircraft position tracking.

Chapter 4

Evaluation environment and data acquisition

In this brief chapter we discuss how we have captured a trace of real world data/signals for evaluating the performance of our work.

In order to thoroughly evaluate our work the decision was made to evaluate our work against real world data. Although our work is capable of operating real-time, we have decided to capture the signals using SDR equipment and store it a file. This file, henceforth referred as a signal trace was then fed into our receiver to receive the results. Using a trace rather than live data has multiple benefits that are also requirements for a scientific work. We used the same trace with multiple implementations and other receivers to ensure a fair comparison and reproducibility of results.

4.1 Signal Trace Acquisition

Normally, when there are no special requirements, a signal trace capturing is relatively simple. An SDR device or a SDR dongle is plugged into a computer and connected to the appropriate antenna. However, the method for evaluating timestamping performance posed a requirement: we need to capture two traces of the same signal, but with two separate SDR devices.

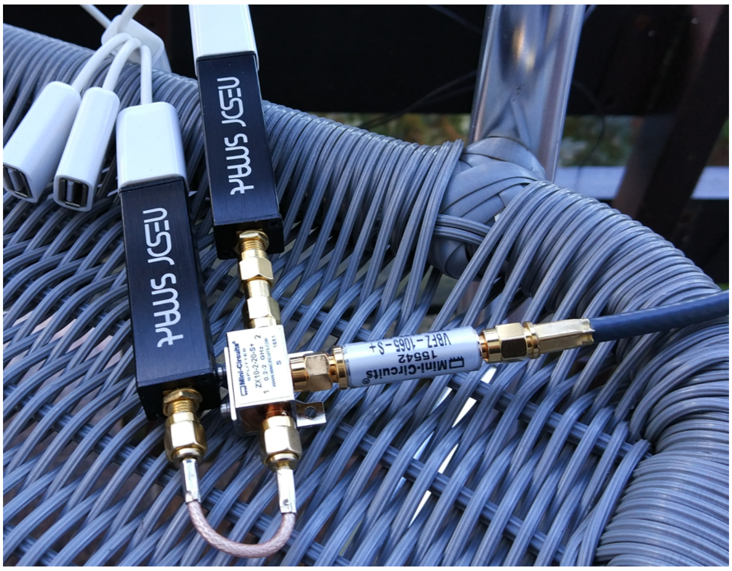


Figure 4.1: Hardware capturing setup: 1090Mhz filter, signal splitter and two SDR dongles

This is due to reasons discussed in the previous chapter. Capturing traces with two devices produces two traces that are coarsely synchronized (in the order of seconds) but are essentially still unsynchronized as each SDR device uses its own oscillator and work independently of each other. The photo on figure 4.1 demonstrated the physical hardware setup during one of the trace capturing sessions. The signal comes from the antenna (pictured on figure 4.2) into a band pass filter that rejects out of band noise and interference. The output of the filter is connected into a signal splitter that provides two (3dB attenuated) outputs that are connected to two SDR dongles. In order to keep coarse synchronization and keep all the settings the same for both dongles, capturing was done on a single computer. The SDR dongles are essentially the same as the popular RTL-SDR, but feature a temperature compensated frequency oscillator that guaranties clock stability of 0.5 ppm. Clock stability of our devices has been tested prior use with methods described in [23].

The antenna (photo on figure 4.2) was purpose-made for receiving signals at 1090 MHz. It is a vertical omnidirectional collinear dipole antenna that

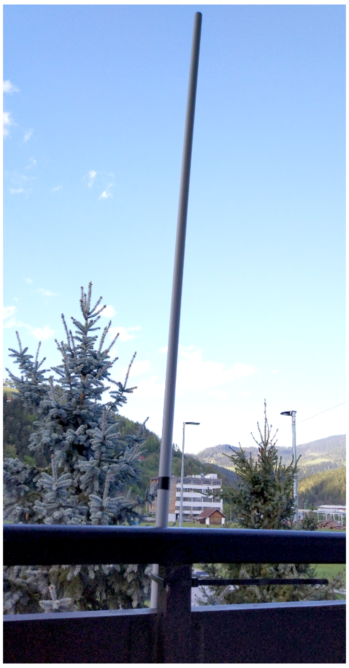


Figure 4.2: Hardware capturing setup: custom made colinear dipole 1090Mhz antenna

features around 6-8 dB gain and a shallow take-off angle that is suitable for receiving aircraft transponder signals that originate further away and consequently at a shallower angle.

4.2 Decoding performance evaluation

An important aspect of receivers overall performance is without doubt the decoding performance. Generally, this is a metric of how good the receiver is able to receive in terms of number of successfully decoded packets. There may only be a single variation or set of configuration parameters for a given receiver and its performance evaluation is straightforward. Our receiver, however, is configurable and the choice of parameter values can affect its decoding performance. In some cases, a set of sub-optimal parameters (in therms of decoding performance) might be preferred because of lighter computational footprint of the receiver with those parameters. In this section we evaluate the decoding performance of our receiver implementation with different sets of parameters and compare the results with those of the widespread Mode-S/ADS-B receiver *dump1090*.

4.2.1 Comparison with dump1090

To perform the comparison of decoding performance of our receiver against the widespread *dump1090*, we used a signal trace captured with an RTL-SDR dongle at the sample rate of 2.4 MHz. The trace contains 18 seconds worth of I/Q samples. Our receiver requires two parameters for its operation:

- Upsampling factor,
- Energy detector SNR threshold in dB.

Upsampling factor determines the factor, by which the whole signal trace is upsampled. Our receiver is configurable and its performance can be tuned by selecting a upsampling that suits the application. The only requirement posed by the internal works of the receiver is that the new sampling *rate* after upsampling must be an integer multiplier of 2 MHz. Due to the fact that our trace has been captured at the sample rate of 2.4 MHz, we have chosen to perform the evaluation with upsampling factors of 5-, 10-, 20- and 40-times.

Energy detector SNR threshold configures the behavior of the energy detector block. The lower the threshold, the more permissive is the energy detector in letting chunks of samples to the next stage in the receiver. If the threshold is set too low, then the energy detector is letting through chunks of samples that are essentially noise. On the other hand, if the energy detector threshold is set too high, then the receiver will not be able to detect week packets. It presents a trade-off between unnecessary work and lost packets. The implementation of *dump1090* does not have the same structure - there is no energy detector block explicitly, yet a condition that only accepts candidates that have SNR higher than 3.5 dB. This condition is equivalent to the energy detector SNR threshold in our receiver and thusly one of the comparison data points is set at the threshold of 3.5 dB to ensure a fair and relevant comparison between the two receivers. Furthermore, the performance metrics were measured also at threshold values of 0 dB, 2 dB and 6 dB.

Receiver version	Decoded Mode-S packets	Unique ADS-B messages
dump1090	2527	979
Our recv. 5x, 3.5 dB	2578	1025
Our recv. 10x, 3.5 dB	2558	1018
Our recv. 20x, 3.5 dB	2554	1016
Our recv. 40x, 3.5 dB	2554	1016

Table 4.1: Comparison of our receiver version against the popular *dump1090*. Table shows results for different upsampling factors at the energy detector SNR threshold of 3.5 dB.

We have measured two metrics that indicate decoding performance: number of successfully decoded Mode-S packets and number of unique ADS-B messages. If we observe the table 4.1 we can see the numerical results for *dump1090* along with our receiver with energy detector threshold of 3.5 dB and varying upsampling factor. We can see that our receiver manages to receive a little more Mode-S packets, however the difference is not really significant. The increase in performance is apparent if we look at the number of unique ADS-B messages. Our receiver manages to receive just little less than 5% more ADS-B messages, no matter what upsampling factor is used. Plots 4.3 and 4.4 display the effects of different upsampling factors and different energy detector SNR thresholds. The magenta square on the plots indicates the performance of *dump1090* We can clearly see that decreasing the threshold value yields more decoded packets and ADS-B messages in almost all cases. Although not so apparent, the same is true for upsampling factor. However, the difference is much smaller. It is clear that our implementa-

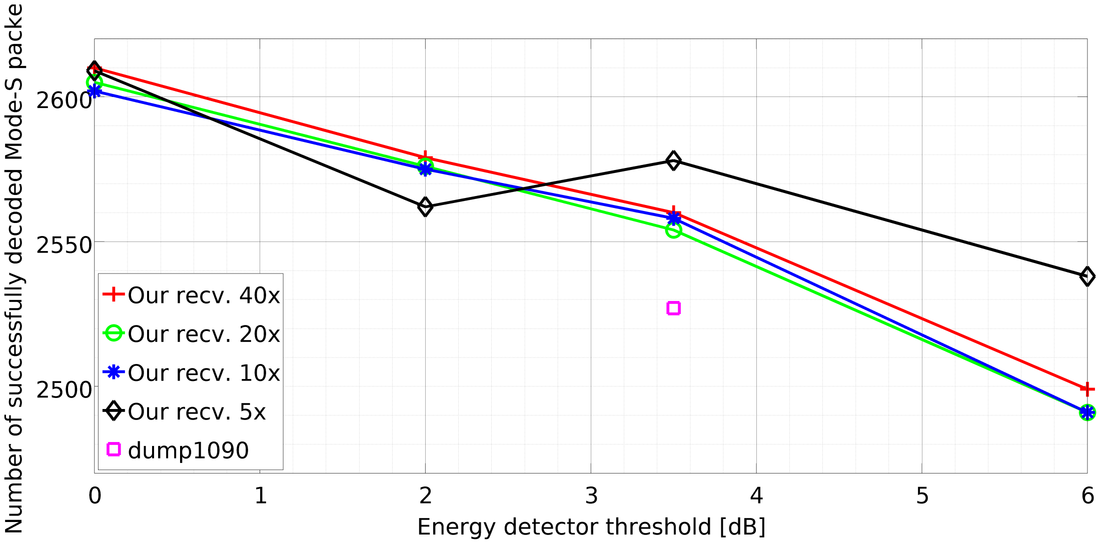


Figure 4.3: Comparison of our receiver against the popular *dump1090*. Plot shows Number of successfully decoded Mode-S packets versus energy detector threshold for different upsampling factors.

tion manages to outperform *dump1090* when compared at a similar energy detector SNR value. We can conclude from the measurements that increasing the upsampling factor and decreasing SNR threshold improves packets decoding performance. However, this comes at a price. Upsampling is a computationally intense operation and running the receiver with wide open energy detector can make a difference of being able to run the receiver in real-time, especially when used on constrained hardware like, for example, the Raspberry Pi.

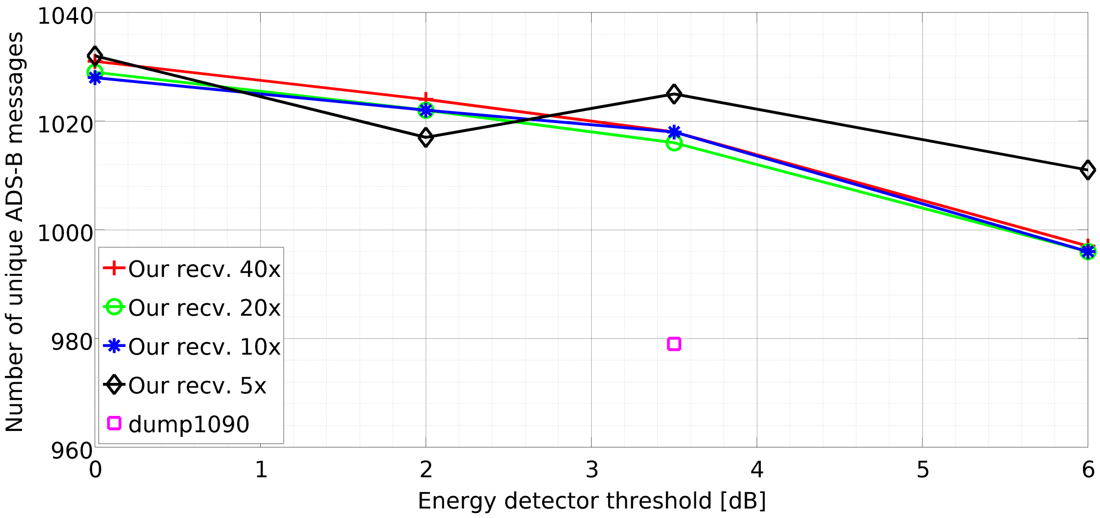


Figure 4.4: Comparison of our receiver against the popular *dump1090*. Plot shows Number of unique ADS-B messages versus energy detector threshold for different upsampling factors.

Chapter 5

High Precision Timestamping

In this chapter we describe our work on implementing and evaluating a method for high precision timestamping. Work done in this thesis regarding timestamping is also basis for a paper that will be submitted to the IPSN'2018 conference.

Timestamping is a process of determining the time of arrival of the packet. Precise time of arrival is very import when considering the decoded packets as data for further processing like multilateration [3] or asynchronous node tracing [24]. In the case of multilateration, better accuracy directly yields more accurate position estimates. Improving the precision and accuracy of timestamping has very beneficial effect for applications and can open new research possibilities.

This work aims highly as the goal for timestamping performance is achieving ToA error variance down to a couple of nanoseconds. When timestamp data is used in triangulation applications, less ToA error variance directly translates to smaller positional error. As a reference, a couple of nanoseconds of ToA error variance translates into a couple of meters of positional precision. However, the goals presents quite a challenge since Mode-S physical format and modulation scheme was only intended to serve the purpose of delivering data, not for precise timestamping. The specifications being rather old, there are quite large tolerance margins regarding pulse timing, mainly due to limited hardware capabilities at the time of developing the system. For example, the position of each pulse might be shifted by ±50 ns with respect to the nominal position. This poses quite a problem when nanosecond-level resolution is the aim of our work. Furthermore, we are facing a system deployed on transponders developed by multiple vendors. These transponders exhibit quite noticeable differences in pulse shapes and timing, yet they still conform to the specifications. Thusly, this poses quite a challenge for this opportunistic use of packet signals.

Generally, there are two different sources of ToA errors:

- Synchronization error,
- Measurement noise.

This work works on reducing the latter. Synchronization error is a separate problem that is not solved by methods described in this thesis. Also, it can be solved by employing GPS precision clock reference as a master clock in order to achieve synchronization. There are also use cases where the application consuming timing data can work entirely with unsynchronized sources, for example tracing a moving node with asynchronous ToA measurements shown in [24]. Therefore, this work focuses on reducing the measurement noise, as it is highly influenced by the method of timestamping.

In following sections we will first look at how the ToA variance was estimated and what evaluation setup requirements were necessary. Next, we will describe two classes of timestamping methods, alternative peak-position based and more canonical correlation based methods. We will describe both classes in detail, show the effects of upsampling factor and evaluate the performance in terms of ToA variance. Moreover, we will show how peak-position based methods can be further improved by classifying packets based on the signal strength and treating each packet class separately.

5.1 Timestamping methods

In this work we have implemented and compared multiple packet timestamping methods. There are two major groups of timestamping methods: the first one utilizes the positions of individual peaks, the second one uses correlation of packet amplitude signal with a template.

5.2 Peak-position based timestamping

The first group of timestamping methods utilizes peaks that are present in the amplitude signal of the packet. In order to utilize the peak positions we must firstly discuss the different peak types that are present in an Mode-S reply packet signal.

5.2.1 Peak types

In Mode-S reply packets there are first 4 preamble peaks and later there are amplitude peaks corresponding with the packet payload as per (B)PPM modulation. However, due to (B)PPM modulation, we are dealing with two different kinds of peaks:

- Type I,
- Type II.

Type I peaks correspond to single peak that are $0.5\mu s$ in length. This peaks occur when modulating (with (B)PPM modulation) bit sequences '00', '10' and '11'. Although not modulated with PPM, preamble peaks are also considered type I peaks, since they are the same length.

Type II peaks correspond to $1\mu s$ long peaks. These peaks occur when modulating the bit sequence '01'. The first bit (1) transforms into the right (latter) timeslot and the second bit transforms into the earlier (left) timeslot forming the pulse sequence of 0110. When pulse shaping is applied, these two pulses merge, forming a single $1\mu s$ long pulse - a type II peak.

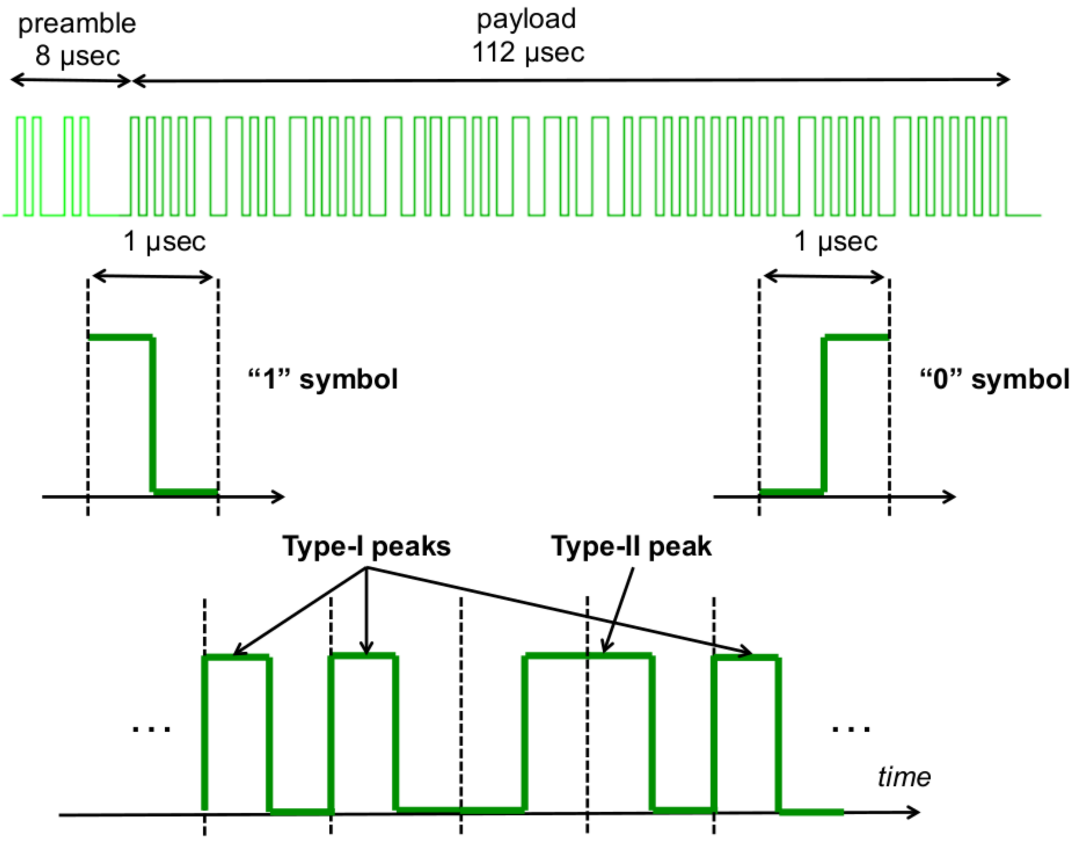


Figure 5.1: (B)PPM modulation and peak types: Type I and Type II peaks

Although not apparent, but this distinction of two peak types is very important for developing a robust timestamping methods. The reason for it will be discussed in the upcoming sections.

5.2.2 Finding peaks

In order to robustly find peaks that only belong to the signal of interest, we employ the following method. Out method requires that the packet in question is correctly decoded (checksum is correct). From the decoded data we are able to re-encode the data where the nominal position of the peaks reside. For each peak we can then search within an interval (width of 1 μs for type I peaks and 2 μs for type II peaks) and determine the maximum value. The sample where the value is maximal is the sample number designated to the peak.

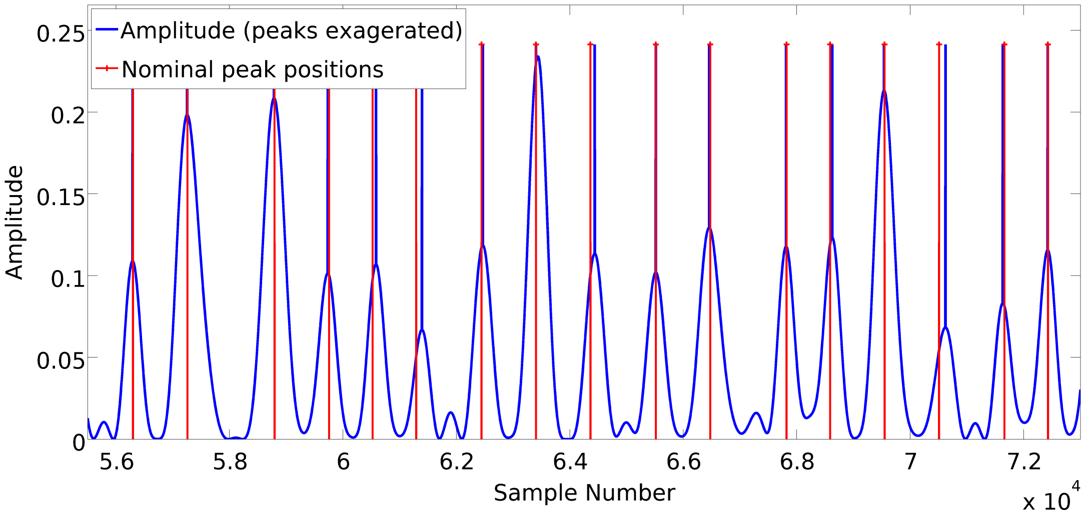


Figure 5.2: Amplitude actual and nominal peak positions

5.2.3 Timestamping using the first peak

Timestamping using the first peak is the most straightforward approach to determine the time-of-arrival of a packet using peaks. The first nominal peak (first preamble peak) is the first local maximum in the packet amplitude signal. This is the simplest and most computationally efficient method.

Finding the position of the first peak also plays a role in other peak based timestamping methods. It serves as a reference. All other nominal peak positions are referenced from the first peak position.

Timestamp value follows the formula:

$$T_s = T_{pp0}; \text{ where } T_{pp0} \text{ is the time/sample of the first preamble peak.}$$

5.2.4 Timestamping using preamble peaks

The next progression of using amplitude peak positions as a timestamping method is to use more peaks. This method uses the first preamble peak position as a reference. Next three preamble peak positions are compared to their nominal positions from the reference and the difference of the peak positions is averaged. The first peak position plus the average peak position difference serve as a timestamp value.

Timestamp value follows the formula:

$$T_s = T_{pp0} + \frac{\sum_{k=1}^{3} (T_{pp_k}^n - T_{pp_k})}{3}$$

where

T_{p0} is the time/sample of the first peak,

T_{pp_k} is the time/sample of the $k-$th preamble peak,

$T_{pp_k}^n$ is the nominal time/sample of the $k-$th preamble peak.

It is important to mention that this timestamping method works even when packet payload is not known or not correctly decoded.

5.2.5 Timestamping using all type I peaks

Once the packet payload is known (correctly decoded), previous method using preamble peaks can easily be extended to the whole packet. At this point some care is warranted as not all peaks in the payload section are the same. The type I peaks are the same in length in time as preamble peaks and can be used directly. Type II peaks, however, are longer and thus have less concentrated information. We see this as a disadvantage and have thus developed a method that only considers type I peaks. Furthermore, type II peaks are not generated the same by all transponders. Some generate a longer, uniform peak, others generate a longer pulse with two peaks. Because the type II peak spans two modulation timeslots, it is unclear how the peak position is to be defined. We have noticed packets where the position of the peak is in the first slot, in between the two slots or in the second slot. This can also vary within the same packet. This hinders the accuracy of timestamping and greater precision was achieved only using type I peaks.

The computation procedure is the same as for preamble peaks. Firstly, the first peak position is determined and used as a reference. Then, positions of all of type I peaks (including preamble peaks) are compared to nominal peak positions and their difference is averaged.

Timestamp value follows the formula:

$$T_s = T_{pp0} + \frac{\sum_{k=1}^{3}(T^n_{pp_k} - T_{pp_k}) + \sum_{k=1}^{N_p}(T^n_{p_k} - T_{p_k})}{3 + N_p}$$

where

T_{p0} is the time/sample of the first peak,

T_{pp_k} is the time/sample of the $k-$th preamble peak,

$T^n_{pp_k}$ is the nominal time/sample of the $k-$th preamble peak,

T_{p_k} is the time/sample of the $k-$th payload peak,

$T^n_{p_k}$ is the nominal time/sample of the $k-$th payload peak,

N_p is the number of payload type I peaks.

5.2.6 Timestamping using type I+II peaks

In order to evaluate if the claim that using type II peaks hinder the performance of timestamping we have also developed a method that uses both type I and type II peaks. The first peak is still used as a reference and peak position differences (from the nominal) are averaged. We have defined nominal position of type II peak to be in between the modulation timeslots or at the $0.5\mu s$ into the peak.

5.3 Correlation based timestamping

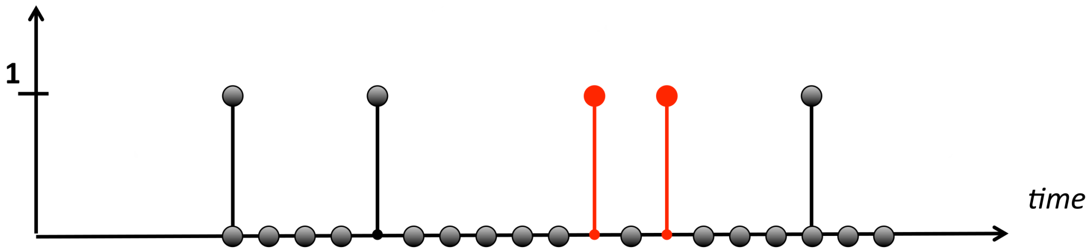
As stated before, Mode-S (and ADS-B) packet signals have loosely defined specifications that allow for a relatively big variance in peak position offset (from nominal position). That makes it less suited for timestamping with canonical approach of correlation with a known template, like for example, WiFi or LTE. In order to support our claim that an alternative timestamping method is in order, we will also evaluate timestamping based on correlation with a template.

Correlation based timestamping methods are different from the peak-position based. They still use the peak positions, although this is implicit, hidden behind the local maximal value of the peak. This methods work by utilizing the whole packet amplitude signal and a generated template. The amplitude signal and the template are correlated with each other and the sample number where the correlation is maximal is used as a timestamp.

$$T_s = \text{argmax } tcorr(\text{signal}, \text{template})$$

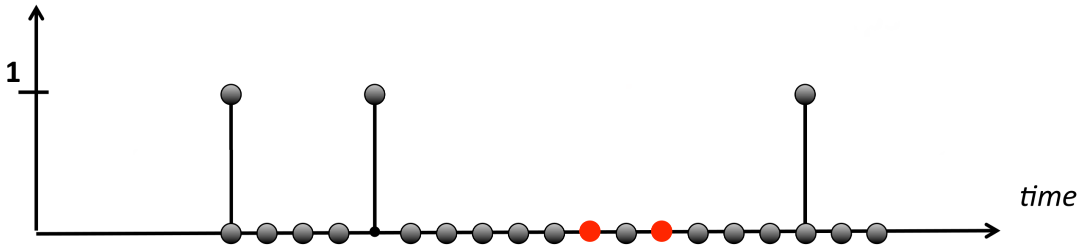
5.3.1 Correlation with template A

Template A consists of impulses of value 1 at type I nominal peak positions, value 1 at type II nominal positions and 0 otherwise.



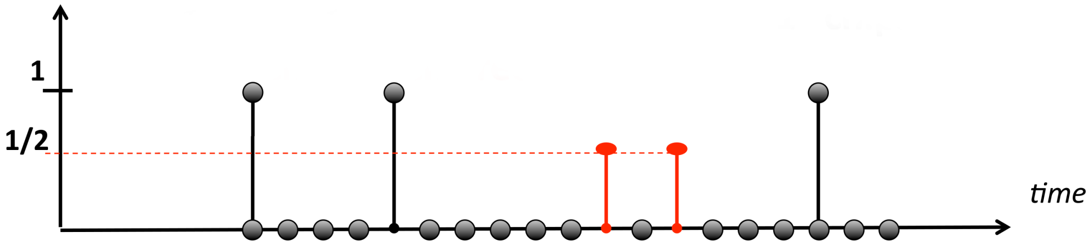
5.3.2 Correlation with template B

Template B consists of impulses of value 1 at type I nominal peak positions and 0 otherwise.



5.3.3 Correlation with template C

Template C consists of impulses of value 1 at type I nominal peak positions, value 0.5 at type II nominal positions and 0 otherwise.



5.4 Results and evaluation of timestamping methods

In previous sections we have revealed multiple methods for determining Mode-S/ADS-B packet ToA. We have shown a canonical correlation with a known template and a more novel approach by utilizing peak positions. To evaluate and compare these methods we had to develop an evaluation methodology that would allow us to estimate the precision of the time-of-arrival measurements. Next section will describe this methodology, followed by results, in-depth analysis of special cases in peak-position-based methods and comparison between the correlation-based and peak-position-based methods, including a possible explanation of the different performance that we have encountered.

5.4.1 Measuring performance of a timestamping method

Evaluating precision of a timetsamping method is a non-trivial task. There are certain limitations that restrict a direct evaluation - difference from the actual packet time-of-arrival. This is a challenge, since we do not have access to high-end measurement equipment (e.g. spectral analyzer and RF signal analyzer) that could serve as a reference for evaluation. Even with the high-end equipment, it would have been non-trivial, since we are aiming at resolution in the order of nanoseconds, and slight measurement error on the equipment might skew or even bias the results.

In order to evaluate our ToA estimation, we must first derive an error model. Lets denote the *true* arrival time of a packet m received by the receiver i with $t_{m,i}$ and the arrival time as *measured* by the receiver i with $t'_{m,i}$. The true ToA $t_{m,i}$ is the time measured by a reference clock that is infinitely precise and has no measurement noise. Mind, such a clock does not exist, but it is a mathematical construct that helps us define our model. There are two main components that contribute to the measured ToA error in respect to the reference clock. These terms are clock error $\xi_i(t)$ at time $t = t_{m,i}$ and measurement noise $\epsilon_{m,i}$:

$$\hat{t}_{m,i} = t_{m,i} + \xi_i(t)|_{t=t_{m,i}} + \epsilon_{m,i} \tag{5.1}$$

The clock error $\xi_i(t)$ is the difference of the measured time at the receiver and the absolute reference clock. We have found that this term is the result of hardware characteristics, more specifically, the stability of the local oscillator of the SDR device in question. Even more importantly, this term is *slowly* varying with time. This opens up the opportunity of modeling and estimating this term with a low-degree polynomial. Unfortunately, we can not directly estimate this term as we do not have the knowledge of the absolute reference time. The remaining term $\epsilon_{m,i}$ represents measurement noise present in TOA estimation process. It can be modelled by a *random variable* with zero mean[1] and variance $\sigma^2_{m,i}$. It is important to note that clock error and measurement noise are independent. If we find a way to accurately estimate the clock error, we can in turn get an estimate for the measurement noise and of course the estimate of precision (inverse of measurement noise variance) for the timestamping method under test. Let us consider the *difference* in time-of-arrival (TDoA) between two receivers ($i \in \{1, 2\}$) that have captured the same packet. The true TDoA can be denoted as $\Delta\hat{t}_m$ and follows the equation:

$$\Delta\hat{t}_m = \hat{t}_{m,1} - \hat{t}_{m,2} \tag{5.2}$$

If we insert the equations for the two right-hand-side terms we get the following:

$$\Delta\hat{t}_m = t_{m,1} + \xi_1(t)|_{t=t_{m,1}} + \epsilon_{m,1} - t_{m,2} - \xi_2(t)|_{t=t_{m,2}} - \epsilon_{m,2} \tag{5.3}$$

and if we rearrange it:

$$\Delta\hat{t}_m = t_{m,1} - t_{m,2} + \xi_1(t)|_{t=t_{m,1}} - \xi_2(t)|_{t=t_{m,2}} + \epsilon_{m,1} - \epsilon_{m,2} \tag{5.4}$$

At this point we can consider our hardware setup (described in the next chapter). The difference between the *true* arrival time $t_{m,1} - t_{m,2}$ at the two receivers is bounded by signal propagation delay. Since the two receivers are connected to the same antenna via a signal splitter (see figure 4.1) and have minimal difference in interconnect cable length, the term $t_{m,1} - t_{m,2}$ is negligible. For all intents and purposes, we can safely consider that $t_{m,1} =$

[1] The assumption of unbiasedness is not critical in this context, since measurement bias, if present, is absorbed by the constant offset term within $\xi(t_{m,i})$ and implicitly compensated along with the latter.

$t_{m,2}$ and can henceforth denote the true arrival time by $t_m = t_{m,1} = t_{m,2}$. Taking into consideration, the equation for $\Delta\hat{t}_m$ simplifies:

$$\Delta\hat{t}_m = \xi_1(t)|_{t=t_m} - \xi_2(t)|_{t=t_m} + \epsilon_{m,1} - \epsilon_{m,2} \tag{5.5}$$

We are left with two groups of terms that can be denoted as $\Delta\xi(t_m) = \xi_1(t)|_{t=t_m} - \xi_2(t)|_{t=t_m}$ and $\Delta\epsilon_m = \epsilon_{m,1} - \epsilon_{m,2}$. Since a difference of two slowly changing terms is still slowly changing, we can approximate $\Delta\xi(t_m)$ with a low degree polynomial. This estimates the temporal profile of the clock error between the two receivers. In our case, degree of $l = 5$ was sufficient to accurately estimate the clock error difference withing a confined time period (of one minute) without the concern of over-fitting. If we denote the estimated clock error difference by $\Delta\xi'(t)$, we can extract an estimate for $\Delta\epsilon_m$ from the TDoA data. TDoA data is computed our by taking the ToA for each packet, received by two receivers. ToA is computed by a timestamping method that is being evaluated. We call the estimate after removing the estimated clock a TDoA residual Δr_m:

$$\Delta r_m = \Delta\hat{t}_m - \Delta\xi'(t_m) \tag{5.6}$$

If we express this in terms of $\Delta\epsilon$ we get:

$$\Delta r_m = \omega_{m,l} - \Delta\epsilon_m \tag{5.7}$$

where $\omega_{m,l}$ is the clock error difference estimation residual from estimation with polynomial regression.

Because TDoA residuals are unbiased by construction, the following holds $E\left\{\Delta r_m\right\} = 0$. Since we are considering identical sensors, with same hardware and software, we can safely assume that the noise variance is the same at both sensors, hence $VAR(\Delta\epsilon_m) = 2 \cdot \sigma_m^2$. Taking the variance of the quantities in (5.7) we obtain:

$$\sigma_{\Delta r}^2 = \nu_\ell + 2 \cdot \sigma_m^2 \tag{5.8}$$

wherein $\nu_\ell \geq 0$ denotes the residual component from the polynomial approximation of the (relative) clock error.

From the data at hand, we can compute an empirical estimate of the variance of TDOA residuals by the Mean Square Error (MSE) $\hat{\sigma}^2_{\Delta r} = \langle \Delta r_m^2 \rangle$ (the symbol $\langle \cdot \rangle$ denoting sample average) and based on (5.8) finally obtain an estimate of the TOA error standard deviation (recall $1/\sqrt{2} = 0.7$):

$$\hat{\sigma}^2_m = \sqrt{\frac{\langle \Delta r_m^2 \rangle - \nu_\ell}{2}} \geq 0.7\sqrt{\langle \Delta r_m^2 \rangle} \tag{5.9}$$

The last inequality of (5.9) stems from the fact that the term ν_ℓ is non-negative. In other words, neglecting such term and taking simply the Root Mean Square (RMSE) of TDOA residuals after ℓth order regression provides a conservative estimate of the TOA precision, i.e., it *overestimates* the true value of σ_m.

5.4.2 Numerical results

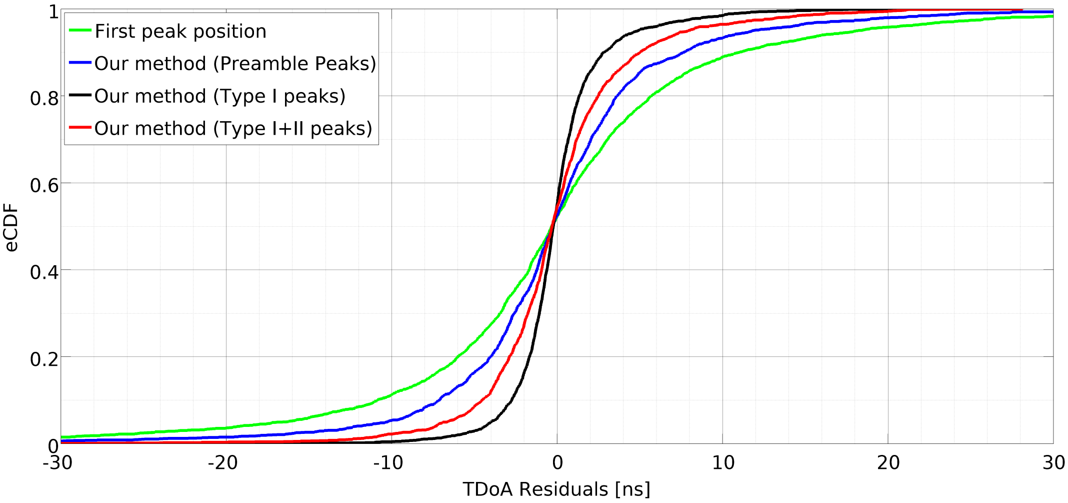


Figure 5.3: Comparison of our peak-position-based timestamping methods (eCDF of residuals)

The performance of various presented timestamping methods have been evaluated with real world data. The two signal traces have been captured

Estimation method	$\hat{\sigma}_\epsilon = 0.7 \cdot \sqrt{\langle \Delta r_m^2 \rangle}$	distance at v_{light}
dump1090	51.3 ns	15.29 m
Correlation template A	13.8 ns	4.14 m
Correlation template B	4.8 ns	1.44 m
Correlation template C	17.3 ns	5.19 m
First peak	8.0 ns	2.40 m
Preamble peaks	5.7 ns	1.71 m
All Type I peaks	2.2 ns	0.66 m
All Type I+II peaks	3.4 ns	1.02 m
All Type I peaks (c0 class)	5.1 ns	1.53 m
All Type I peaks (c1 class)	1.2 ns	0.36 m
All Type I peaks (c2 class)	2.7 ns	0.81 m

Table 5.1: Summary of estimated ToA variance for different timestamping methods. Distance at the speed of light v_{light} is provided as an projected estimate for multilateration precision if a given timestamping method was used to produce ToA data.

using a hardware setup described in chapter 4, using identical SDR hardware. Both traces were processed through our receiver and successfully decoded packets[2] were subjected to various timestamping techniques (including our high-precision peak-position-based method) to measure packet ToA (time-of-arrival). Successfully decoded packets from both traces were matched and TDoA (time-difference-of-arrival) were computed from matched pairs of packets from the two receivers. Totally, we have matched 2060 successfully decoded packets from 18 seconds worth of traces captured at sample rate of 2.4 MHz.

As discussed in section 5.4.1, we can use the values of the TDOA residual RMS $\langle \Delta r_m^2 \rangle$ and the corresponding estimate $\hat{\sigma}_m = 0.7 \langle \Delta r_m^2 \rangle$ to indicate the precision ($\frac{1}{\hat{\sigma}_m}$) of a given timestamping method. Results, including the

[2] Packets that have valid CRC checksum.

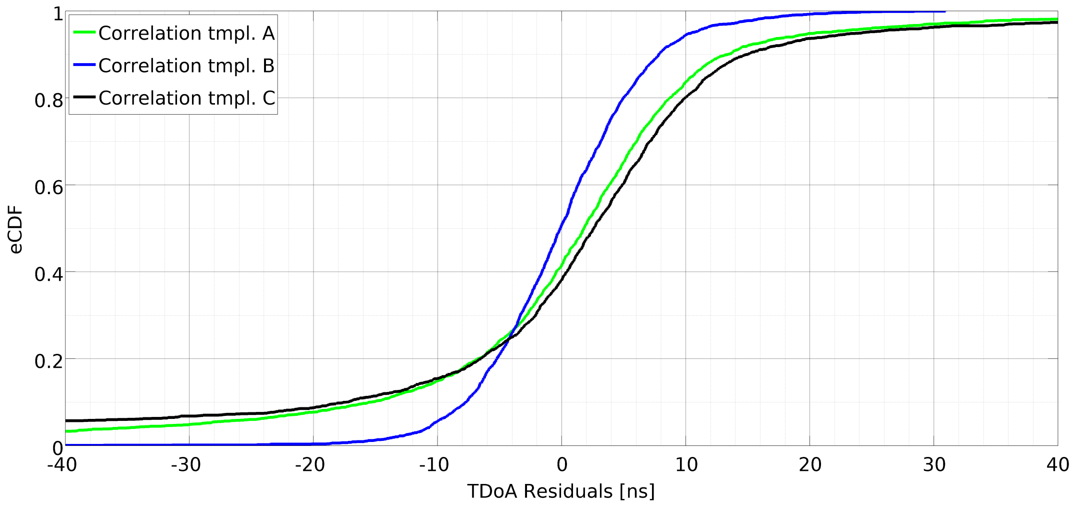


Figure 5.4: Comparison of correlation-based timestamping methods (eCDF of residuals)

values achieved by *dump1090* - included as a reference, are presented in table 5.1. The lower the value of $\hat{\sigma}_m$, the better the precision. Similarly, we can present the results graphically by plotting the empirical cumulative distribution function (ECDF) of the TDOA residuals - the steeper the curve, the higher the precision. We have compared the peak-position based methods with different number of peaks (figure 5.3), correlation based methods (figure 5.4) and finally the best methods of each class with the *dump1090* timestamping precision for reference (figure 5.5).

If we look at the numerical results, we can observe that the best performing method[3] is the peak-position-based, where all type I peaks were used. It achieves deviation of 2.2 ns that would translate in 66 cm of accuracy in multilateration applications, which is a great result. Furthermore, if we look at the values for peak-position-based method with only the first peak and using only the preamble peaks, we can see a clear trend - using more type I peaks benefits the precision. In average, there are 56 type I peaks

[3] Without considering packet strength classes.

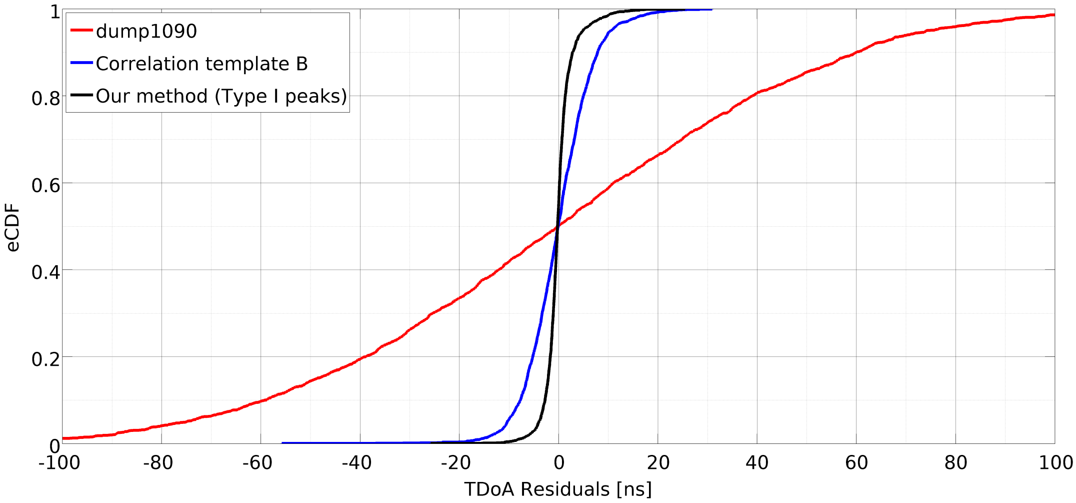


Figure 5.5: Comparison of our best performing methods (of each type) with dump1090 (eCDF of residuals)

in a packet, contrast to only 4 preamble peaks. On the other hand, the inclusion of type II in the peaks hinder the performance of the timestamping method, increasing the deviance to 3.4 ns. It seems that type II peaks also hinder the performance of correlation-based approach. The only correlation method that came close to the results of peak-position-based methods was correlation with template B, that also does not consider the type II peaks. Looking at figure 5.4, we can see that variance of the residuals of correlation with template B conforms to normal distribution, while templates A and C exhibit behaviour that does not conform to normal distribution. This is something that could be a topic of research for future work, however, given that correlation based methods suffer from higher computational complexity and are not as elegant as peak-position based methods, we did not investigate the phenomenon in this thesis. Lastly, comparing the timestamping performance of *dump1090* with our methods, it is clear that we have achieved a massive increase in precision. Nonetheless, we have continued to search for improvements to our method, we discuss that in the next section.

5.4.3 Packet signal strength classes

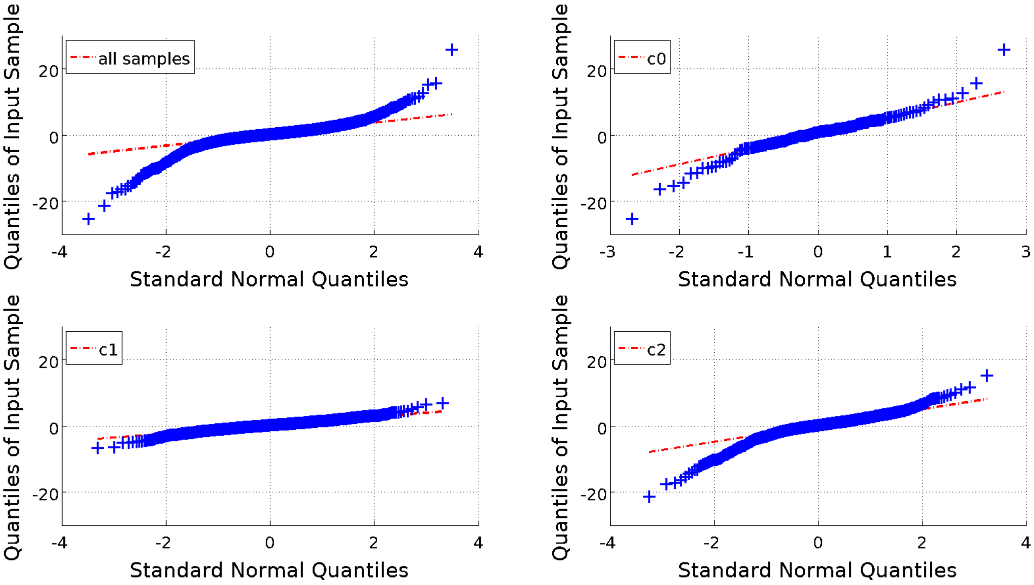


Figure 5.6: Standard quantiles of different packet classes (Type I peak method)

While the basic peak-position-based with type I peaks timestamping method already performs more than admirably, we still noticed that the TDoA residuals did not completely conform to normal distribution. This can be observed on figure 5.6 in the upper left subplot. This subplot shows empirical quantiles of TDoA residuals against the quantiles of a fitted normal distribution (qq-plot). We can see that in the center part it conforms very nicely to the fitted normal distribution, however there are still quite noticeable tails that deviate from the normal distribution. This led us to believe that not all data conforms nicely and that we might be dealing with data that could be better fitted to multiple normal distributions. However, this required us to classify packets into multiple classes, and fit normal distribution over TDoA residuals for each class. After in-depth examination, we have found that packet signal-to-noise ratio could be a good attribute for split-

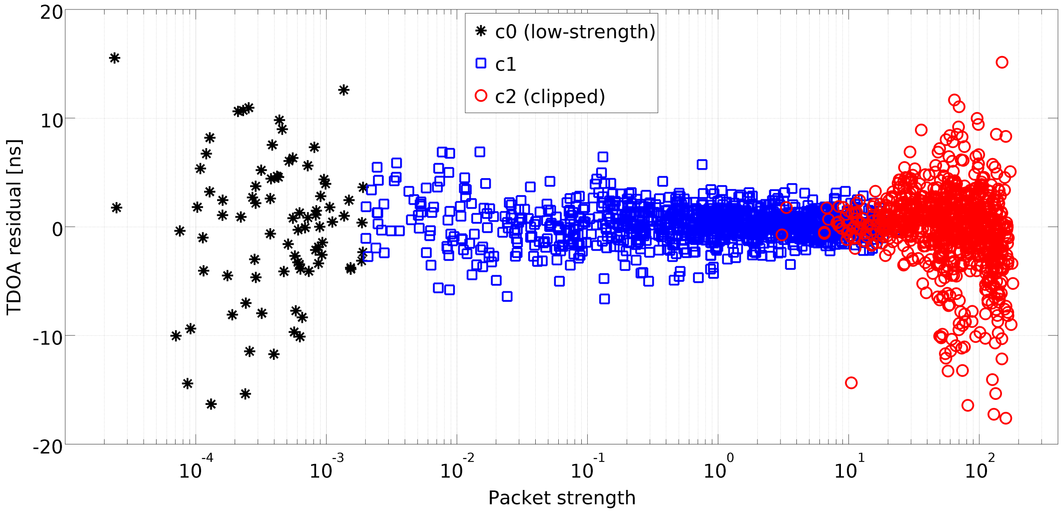


Figure 5.7: TDoA residuals of different packet classes (Type I peak method)

ting our data into multiple classes. This was based on the hypothesis that weaker packet signals (low SNR) would yield a higher measurement noise variance due to the relatively high power of the noise. On the other hand, strong packets (high SNR) will produce lower measurement noise variance. Moreover, we have observed that quite a lot (about 40 %) of packet signal exceeded the ADC dynamic range and resulted in clipping. Since the introduced distortions we predict that this will negatively affect the timestamping performance. To estimate the SNR we used a metric called packet strength that we defined as:

$$\psi_m \stackrel{\text{def}}{=} \sum_{k=1}^{K_m} \rho_k^2$$

where $\rho_k \stackrel{\text{def}}{=} |s'[n]|\big|_{n=\hat{\tau}_k}$ denotes the maximum amplitude of the k-th type I peak in the upsampled signal $s'[n]$. This gave us a reliable measure of packet strength that can be used for classifying packets.

To define classed of packets, we plot in figure 5.7 for every individual packet m the measured TDoA residual Δr_m against the packet strength ψ_m.

We can observe that TDoA residuals deviate further from 0 when packet strength gets weaker. Similarly, TDoA residuals converge closer to 0 the stronger the packet gets. At some point we can observe that TDoA residuals start rapidly diverging away from 0. This is due to packet signals saturating the ADC and the signals get clipped. Accordingly, we define three classes:

c0 : all TDoA samples associated to packet strength below a given threshold, specifically $\psi_m \leq 0.002$, are marked with a black asterisk "$*$".

c2 : all TDoA samples associated to packets that were clipped by at least one of the two sensors, marked with a red circle "$\circ$".

c1 : all remaining packets, labelled with a blue diamond "$\diamond$".

After defining the classes we were able to process each class of packets separately. Figure 5.6 shows that gaussian distribution is very well approximated for the co and c1 classes. Class c2 still exhibits some deviation from the normal distribution. This could indicate that among packets with clipped I/Q samples, there is still room for improvement. Despite this fact, clipping can effectively be avoided by employing automatic gain control algorithms for SDR devices, or by using devices with higher ADC bit-depth. After all, the devices in our testbed use only 8-bit ADC, while devices with 12-bit ADC are readily available, albeit not as low-cost. On the other end of the packet strength spectrum, we have the packets classified into c0. These account for only 5 % of all packets. Packet strength depends upon multiple factors, including gain values in the SDR radio front-end, antenna, geographical location and of course the position and distance of aircraft to the receiver.

When considering each class separately, we obtain remarkable results. The classes c0 and c2 exhibit estimated ToA error variance of 5.1 ns and 2.7 ns, respectively. This is expected as the packets in c0 and c2 are considered of a lesser quality. The most awe-inspiring result comes from the estimated ToA error variance of class c1. We were able to obtain a value as low as 1.2 ns, that corresponds to *only* 36 cm at the speed of light. This is especially impressive when we consider the simplicity and computational efficiency of proposed timestamping method.

Another result is the usefulness of packet strength as an indicator of timestamping precision. This is especially convenient for downstream algorithm (like multilateration) that can leverage the quality indicator and assign higher importance to better quality ToA estimations, consequently increasing the precision of the algorithm. In the case of multilateration, such weighting of the input ToA data would induce higher precision positional estimates.

5.4.4 Effects of upsampling

In order to achieve better timestamping precision and accuracy the packet signal can be upsampled prior executing the timestamping procedure. This has a two-fold effect. Firstly, it improves the resolution of timestamping. Secondly, it increases the processing load as it needs to do more processing. In case of peak-position based timestamping, upsampling brings better resolution when determining positions of peaks. This greatly improves timestamping accuracy at the beginning (upsampling factors from 2 to 20), but later we start to observe diminishing returns. This can be observed on figure 5.8.

The effect of upsampling is more easily quantified when using correlation based methods. The precision corresponds closely to the upsampling factor. This is due to the fact that value used as a timestamp is the sample number where the correlation of amplitude signal and the template is the greatest. More samples there are, more fine-grained the resulting value can be. It is important to note that higher upsampling factor does not automatically yield a more precise timestamp, but a lower upsampling factor can limit the precision due to the resolution of the output value.

In the case of peak-position based timestamping methods this did not present such a limitation because of the averaging. Lower upsampling rate, however, did present a limit to the precision of the peak positions (and thus affected the performance), but the datatype of the resulting value did not as

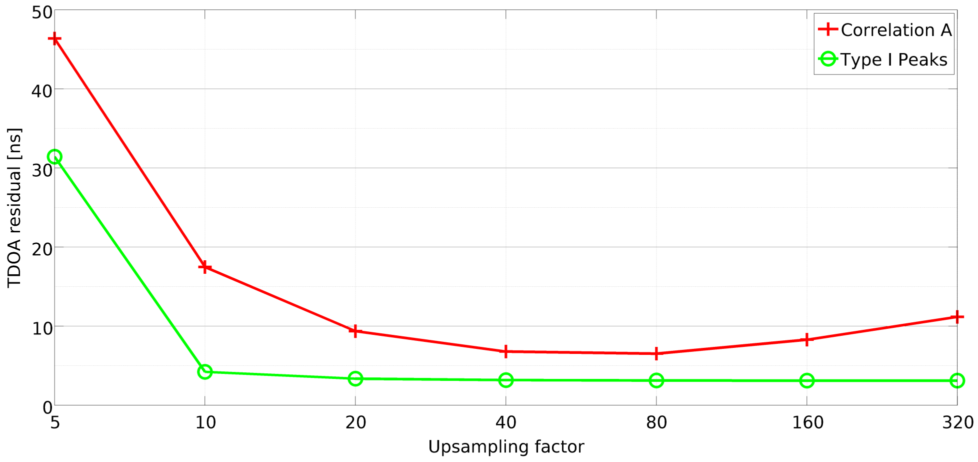


Figure 5.8: TDoA variance versus upsampling factor for correlation and peak-position-based methods.

it was a floating point number.

Figure 5.8 shows how upsampling factor affects correlation based timestamping methods. It can be observed that up to the upsampling factor of 40, there is benefit to upsampling, but further increasing can lead to unnecessary consumption of computational resources and can even hinder the performance of correlation based timestamping methods. The peak-position based methods have a very steep fall to the best TDoA residual value and further upsamling has no benefit.

5.4.5 Comparison of correlation base methods with peak-position based methods

The results clearly indicate that correlation based methods perform worse than peak-position based timestamping methods. Interestingly, peak-position based methods still give better results even when used with as much as 10 times lower upsampling factor.

One of the key reasons for worse performance of correlation based times-

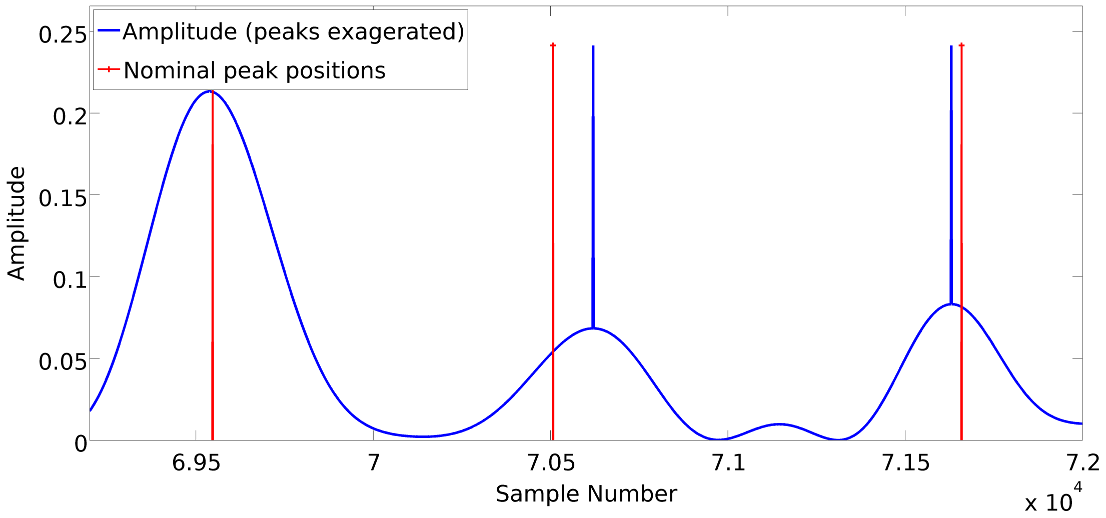


Figure 5.9: Positive and negative peak position offset at neighboring peaks

tamping methods is the fact that they are not robust when used on real world I/Q traces. This is due to the fact that Mode-S transponders aircraft are a lot of times imprecise in timing. Decoding (the intended purpose of the signal) is affected very little by these transmitter imperfections, but for timestamping this can have a huge impact. Figure 5.9 shows two peaks of a packet amplitude signal and their respective nominal positions. We can see that the first peak is late in comparison to the nominal position, however, the second peak is early relative to the nominal position. Note that out correlation based timestamping methods use a fixed template and that the peaks are spaced apart nominally. The case where one peak is early and the other one is late causes the correlation template to miss at least one of them, resulting in a poor performance. Please mind that the occurrence of early/late peak pairs is something we observed in a lot of packets from different aircraft and is not an isolated anomaly. Our peak-position based methods are far less sensitive to the effect of the early/late peaks because peak position differences are averaged and early/late pairs are somewhat canceled out.

Another reason why correlation based methods perform worse than peak-position based methods is that peak amplitudes are not equal - there are some stronger peaks. Since correlation operates on the value/magnitude of the whole signal (contrast to peak-position based method that works on the position/time of the peaks in the signal), it tends to be pulled towards stronger peaks and worsen the fit at lower level peaks.

Furthermore, correlation based timestamping methods work on the whole signal (the whole chunk of samples containing the packet) and are more affected by the noise and interference as the correlation is computed over areas where no useful signal is present. Peak-position based timestamping methods are better in that regard. The peak finding process is limited only to a small interval (around the nominal position) where the signal should be present.

Lastly, correlation based methods suffer greatly when Doppler shift has occurred. Even if peaks were perfectly timed at the transmitter, they would get stretched out by the Doppler effect, and the fixed correlation template would fit very poorly. Peak-position methods are more robust as they still produce a meaningful timestamp, although it can be shifted in time because of the Doppler effect. This could be tracked and counteracted in post-processing, but is out of scope for this thesis.

Chapter 6

Cancellation and improving packet decoding

One of the advantages of software defined radio is the possibility of running multi-pass processing algorithms on the incoming signal, or parts thereof. For example, we can extract only a certain part of the incoming signal and run multiple passes of the same algorithm without penalty of requiring more hardware. Furthermore we can employ computer libraries (for example, polynomial fitting) that would be very difficult or very expensive to implement in hardware or data-flow architectures. Leveraging those benefits we wanted to develop a method for improving packet decoding by resolving certain cases where a normal decoder would not yield useful results. One of those situations is when two aircraft transponders access the wireless channel (transmit) at the same time and both signals are merged together - packets collide.

In order to develop a system for packet collision resolution, we must be able to perform packet cancellation - cancel out of the packets. One of the packets needs to be estimated in enough detail so that cancellation techniques can be applied.

This chapter describes how we estimated signal attributes, how the packet encoding is done and how it all comes together as a method for Mode-S reply packet cancellation.

6.1 Estimating physical signal properties

In order to achieve decent packet cancellation, the cancellation signal that is to be subtracted from the source needs to be as close to the actual signal of the packet. Consider a case where there are two packets colliding. Lets call them packet A and packet B, the source signal S and the cancellation signal C. If the packets A and B have collided the source signal S would contain the sum of both signals. In this case we want the cancellation signal C be as close as possible to A, in order to obtain something resemblant of B when we subtract A from S. If we knew A or B in advance, this would be a trivial task. Because we do not know A or B, we must try to estimate A from the source signal S, packet payload data and external information about the signal format. This includes the modulation scheme used as well as data encoding. We must gather as much properties of A just by observing S. In this work we focused on observing the following properties:

- Amplitude,
- Phase,
- Frequency offset (from the nominal 1090 MHz),
- Peak locations and accurate/precise timestamping,
- Packet payload [1]

6.1.1 Estimating Amplitude

In previous chapter we described the method for high precision timestamping. A part of this method is also determining the peak positions and values at these peaks. This is also useful for estimating peak signal amplitude. Because we already know where the signal is present, we can use these points to sample amplitude values. Nominal peak positions are calculated from the decoded

[1] This is not actually a physical property of the signal. It could also be summarized into the previous bullet point since peak positions are dependent on the payload data.

data, and actual peak positions are determined by searching for a maximal value in the vicinity of the nominal peak positions.

In our approach we have opted to use the average peak amplitude as the signal amplitude estimate. Peak amplitude is kept constant at this level for generating the amplitude template. Because the peak template is generated with the peak amplitude of 1.0, it is only required to multiply the templates with the peak value determined by amplitude estimation.

6.2 Estimating phase and frequency offset

When trying to replicate the signal as closely as possible in order to apply cancellation, the amplitude signal is not the only thing that must be matched closely. Perhaps even more important is the phase progression of the signal. Even if the estimated amplitude shape is identical to the signal, we would not be able to successfully cancel out the packet if phase progression was wrong. In fact, we would likely achieve an opposite effect - making the packet signal even stronger.

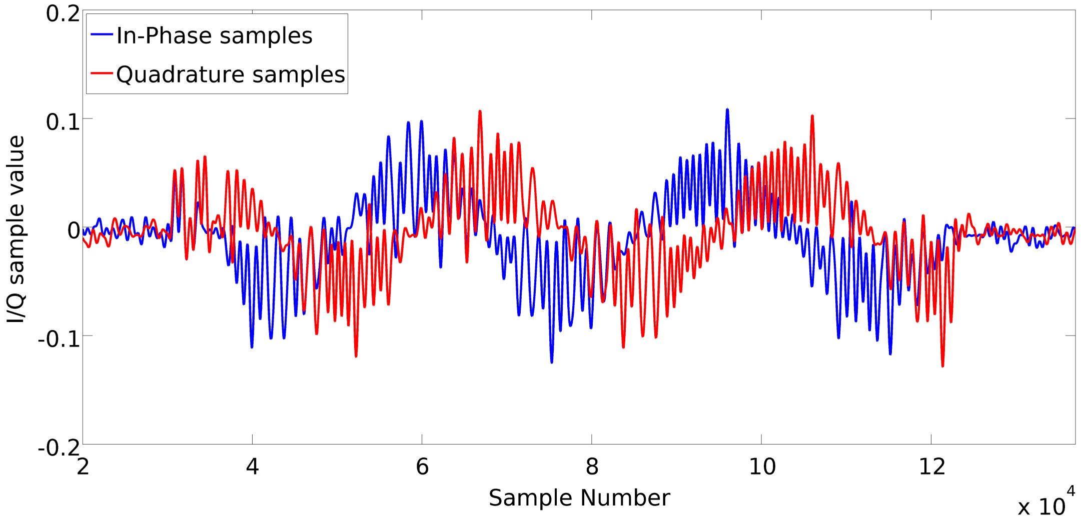


Figure 6.1: Typical packet I/Q signal with frequency offset

It is important to mention that we are dealing with real system that consists of real, imperfect hardware. That includes transmitters (transponders) on the aircraft as well as the receiver used for capturing the signal. Transmitters used on the aircraft are often old, containing imprecise clock oscillators. That can lead to mismatch in transmit and receive frequency, manifesting itself as a non-zero frequency of the received (down-converted) signal. This can also be seen as oscillations of the I and Q sample values though time (figure 6.1). Please note that this frequency offset does not affect the amplitude signal, only the phase progression. Instead of constant phase, there is a mostly linear progression of the phase (constant frequency offset). Because the amplitude is unchanged this is not a problem when only decoding the packet payload, considering that with the particular modulation scheme adopted in Mode-S, namely (B)PPM, the encoded information is borne exclusively by the signal amplitude, not phase.

If there was only one transmitter (aircraft) this issue could be addressed by tracking this frequency offset and adjusting the receive frequency accordingly. However, we are interested in capturing packets from as much aircraft as possible. That poses a restriction on altering the receive frequency - it must remain fixed. To combat the problem of frequency offset we can try to estimate it on per aircraft and per packet basis.

If we observe figure 6.2 where there are amplitude and phase signals of a packet, we can see a regular amplitude shape and a rather chaotic phase signal. Should our estimation be based directly on the phase signal, we would be making a rather large error. The reason behind the big estimation error is the seemingly random phase signal.

This is caused by the fact that the signal of interest is not present all the times. The amplitude of the signal goes below the noise on some parts of the packet. At these spots we are essentially observing the phase of noise, which is almost certainly random (if there is no other interference). The samples with random phase and the actual phase of the packet signal are weighted equally and this causes a large error in estimation and more importantly it

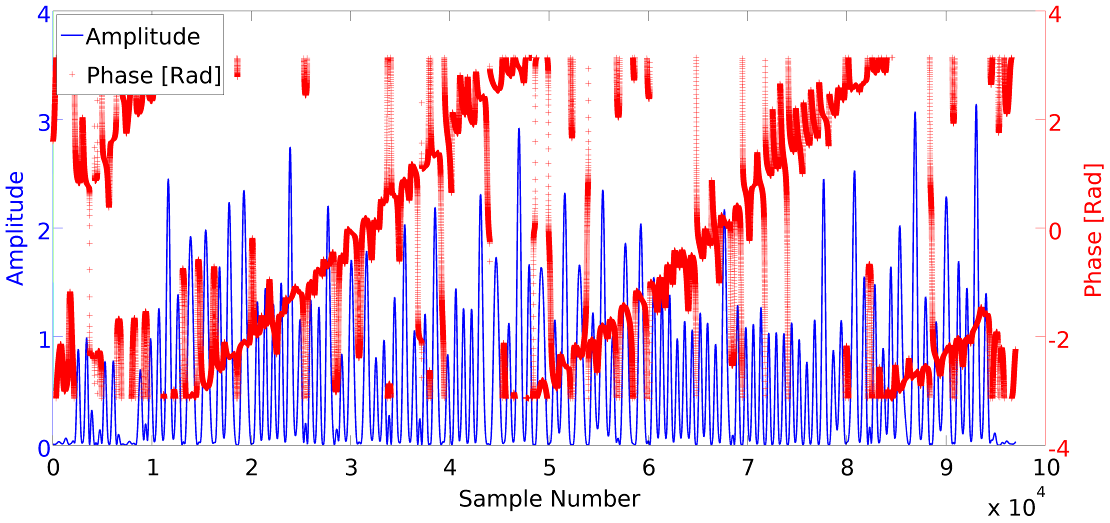


Figure 6.2: Phase of a packet

prevents phase unwrapping. To avoid processing random phase of noise we have taken a slightly modified approach. We already know the peak positions - sample positions where signal of interest is very likely to be present instead of noise. To get a better input for the next stage of phase (and frequency) estimation we only consider the phase samples near the peak positions and only if the amplitude is strong enough (a threshold determined empirically to be 77% of average peak amplitude). On figure 6.3 we can now observe a much clearer phase signal. There are also clearly visible spots where the phase has been wrapped. At this point we were able to unwrap the phase of the peaks and obtain a set of training points for the next stage. Unwrapped phase can be observed on figure 6.4.

When phase training points are determined, we proceed with fitting a polynomial over the training points (figure 6.5). In our case, linear polynomial yielded the best results. Its coefficients give us estimates for the initial phase and frequency offset.

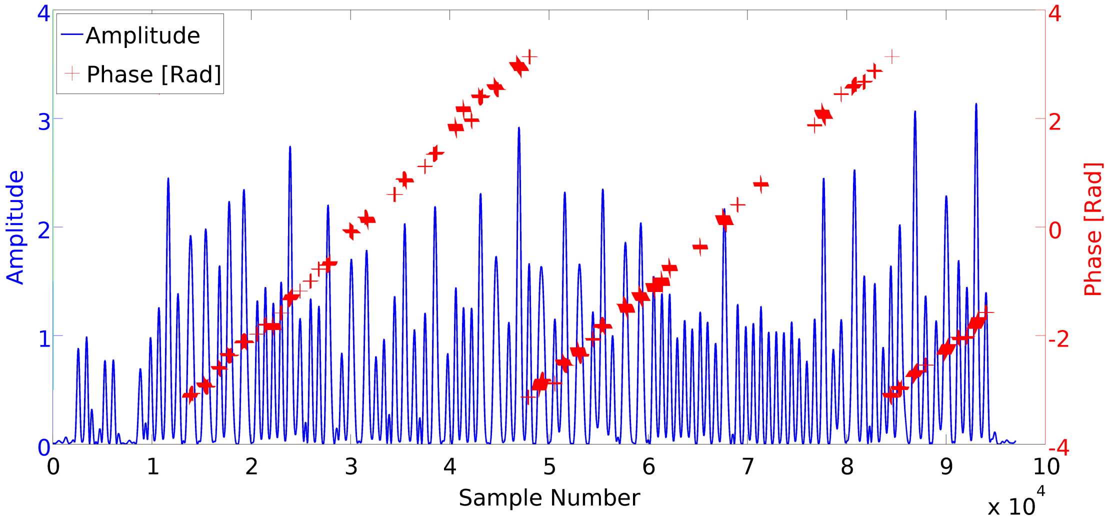


Figure 6.3: Phase of Peaks - Wrapped

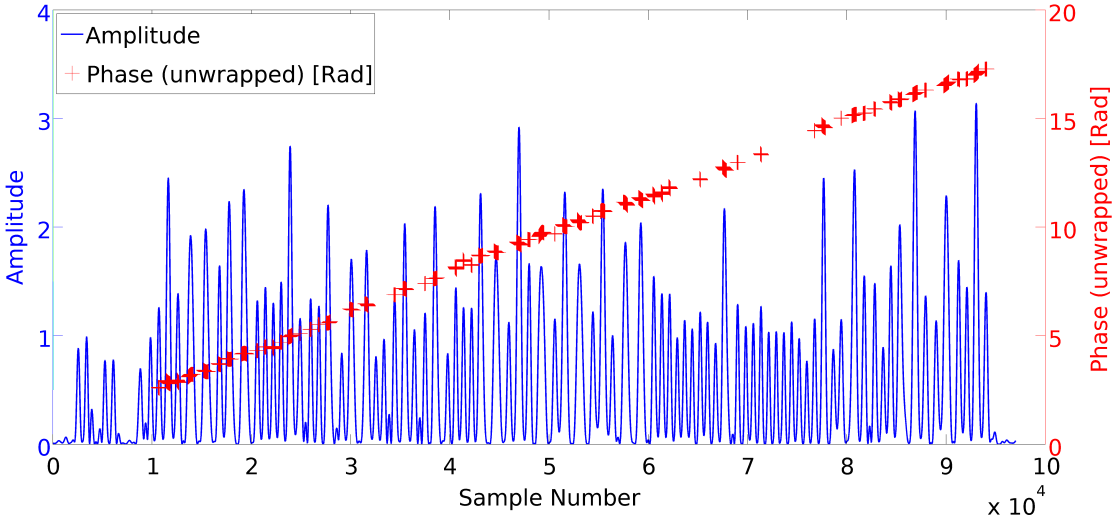


Figure 6.4: Phase of Peaks - Unwrapped

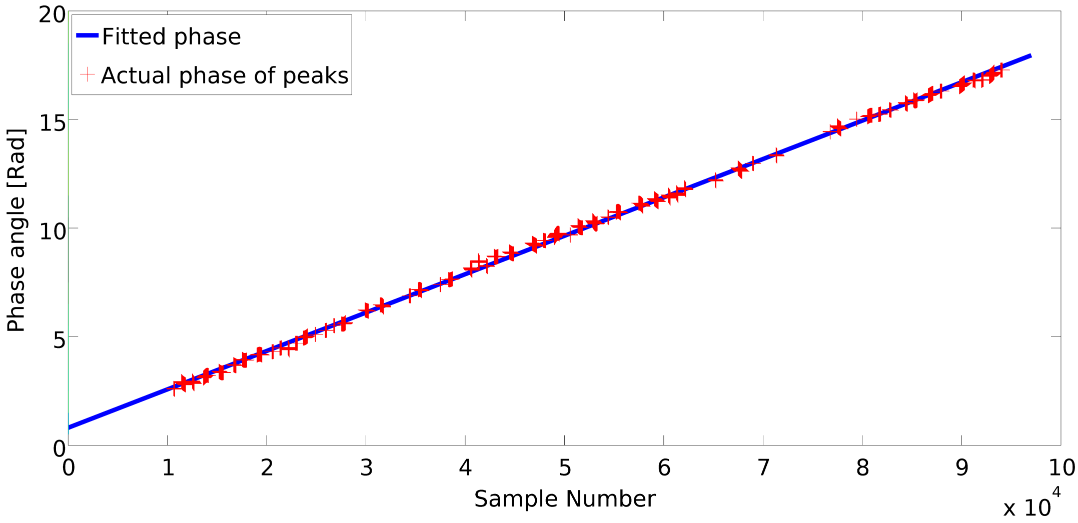


Figure 6.5: Fitted phase on training points

6.3 Packet encoder and signal generator

In order to generate packet signals from estimated signal properties and payload data we must be able to encode the payload data with pulse position modulation, append the preamble and generate pulses according to the estimated properties.

The packet encoding or packet signal generation is done in multiple stages. Firstly we must interpret the payload data from the input encoding and convert it into a string of bits. The string of bits is then encoded with the pulse position modulation. For each bit there are two possible positions that it can occupy within its designated slot. If the bit value is 0, then it will produce a pulse at the earlier (left) position, similarly, if the bit value is 1, it will produce a pulse in the later (right) position. When a string of pulse positions is generated from bits, the encoder prepends the signature preamble (like the one on figure 2.1). This creates a packet frame that is a valid Mode-S reply packet.

The next stage in signal generation is to generate an amplitude envelope from the string of pulse positions created in the previous step. Each pulse should be shaped according to estimated signal properties (amplitude, peak width) or, in case of generating signals for (potential) transmission, with accordance to ICAO specifications [5]. Detailed explanation of pulse templates is described in the next subsection.

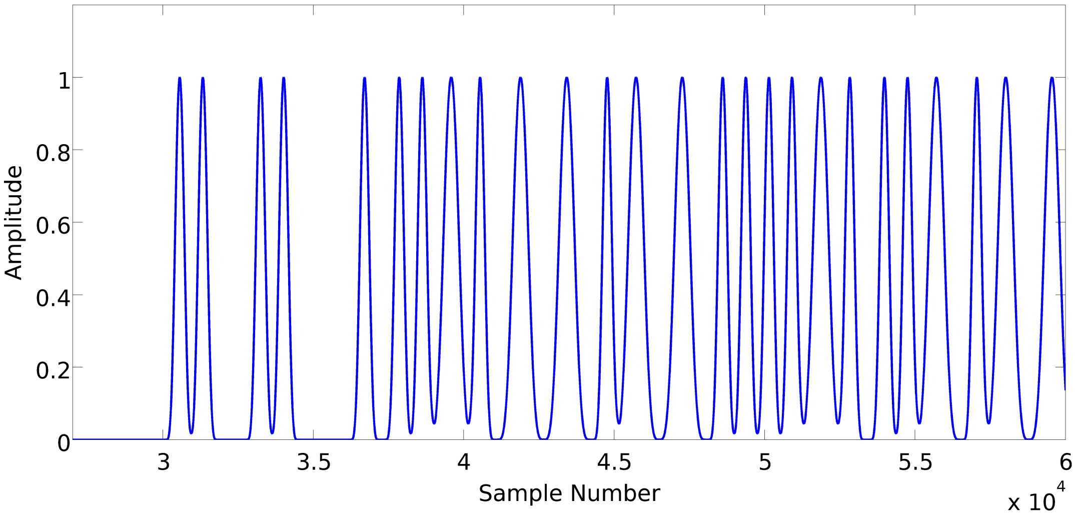


Figure 6.6: Fragment of the generated amplitude envelope

Generated amplitude envelope is valid packet baseband signal with constant phase (figure 6.6). However, in the case of cancellation, we must also apply phase and frequency offset to the signal in order to estimate the received signal. The whole encoding process can be observed on figure 6.7.

6.3.1 Pulse template

In the process of modulating we need to transform the (B)PPM impulses into actual pulses. To achieve this we replace the impulse with a pulse template. To generate a suitable pulse template we use a raised cosine function. This allows us to adjust and tune multiple parameters of a pulse template: rise time and hold time. We can adjust these two parameters to best fit the

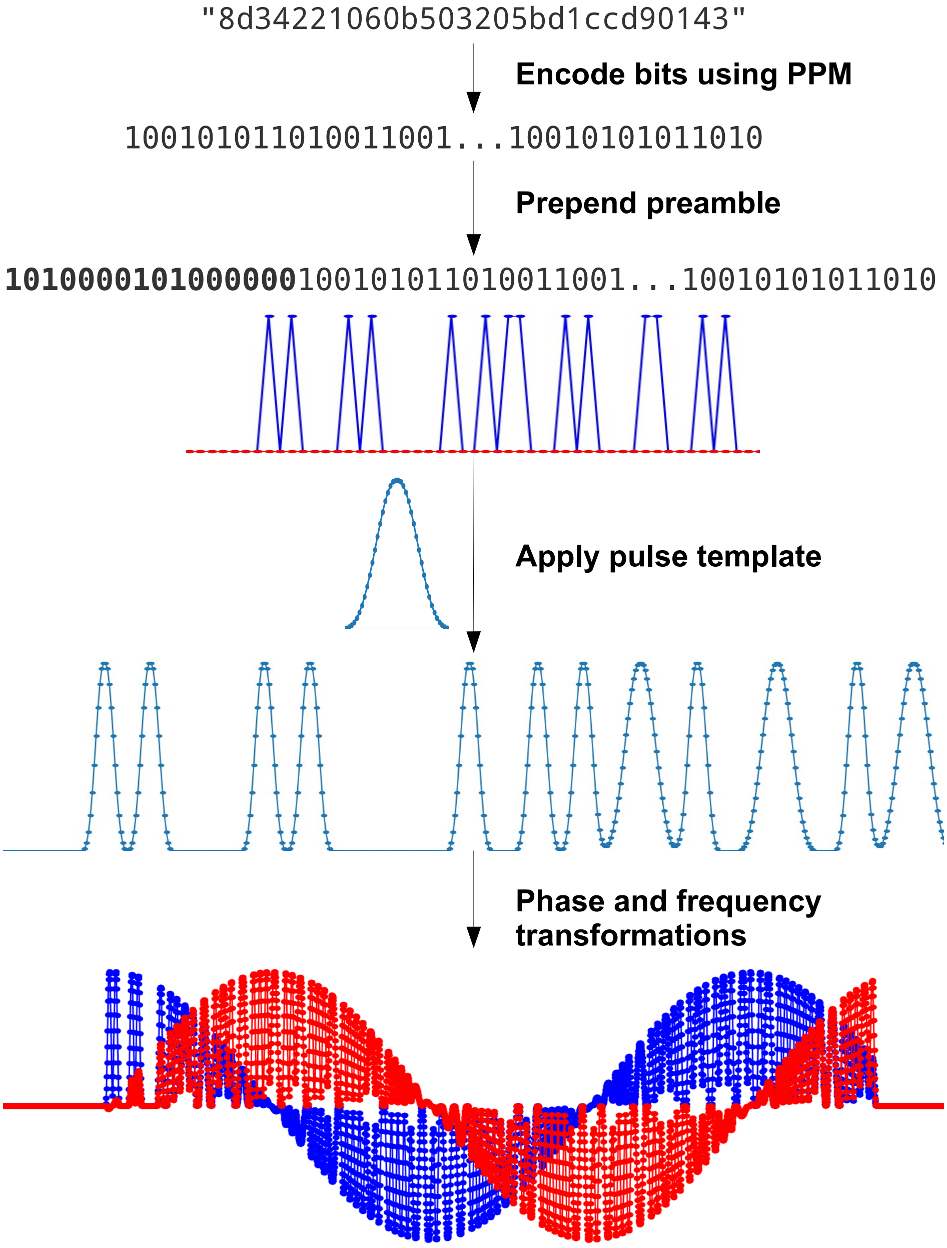


Figure 6.7: Encoding process

source signal when applying cancellation, or when generating a new packet, to keep the signal bandwidth in the limits specified in the ICAO specifications [5]. When generating the template, amplitude is kept at unitary value to allow easier further processing of the signal. We generate two different pulse templates, one for type I peaks and another, wider, for type II peaks.

6.4 Packet cancellation

In the previous sections we have argued and showed how to estimate relevant signal attributes that are essential for packet cancellation. Furthermore, we have described the Mode-S reply packet encoding and signal generation process. In this section we will show how we employ the estimated attributes and signal generation in order to generate a packet signal that we can subtract from the signal trace to cancel out the packet.

The cancellation process consists of multiple steps:

- Successfully decode the packet (or one of the packets if a collision has occurred),
- Estimate amplitude - based on payload, find the peak positions and values,
- Estimate phase and frequency offset,
- Encode the payload into string of pulses and generate amplitude envelope,
- Apply phase and frequency offset transformation,
- Subtract the generated signal from the original trace,
- Process the remainder again through the receiver chain.

The process stated above can be applied to cancel out the packet signal. This is useful for collision resolution that is discussed in the next section.

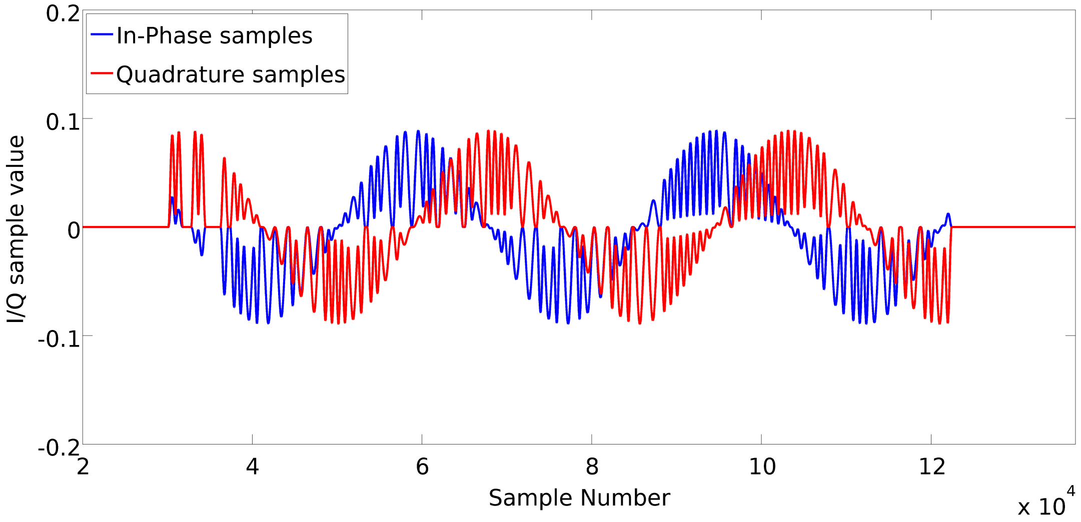


Figure 6.8: Generated packet signal

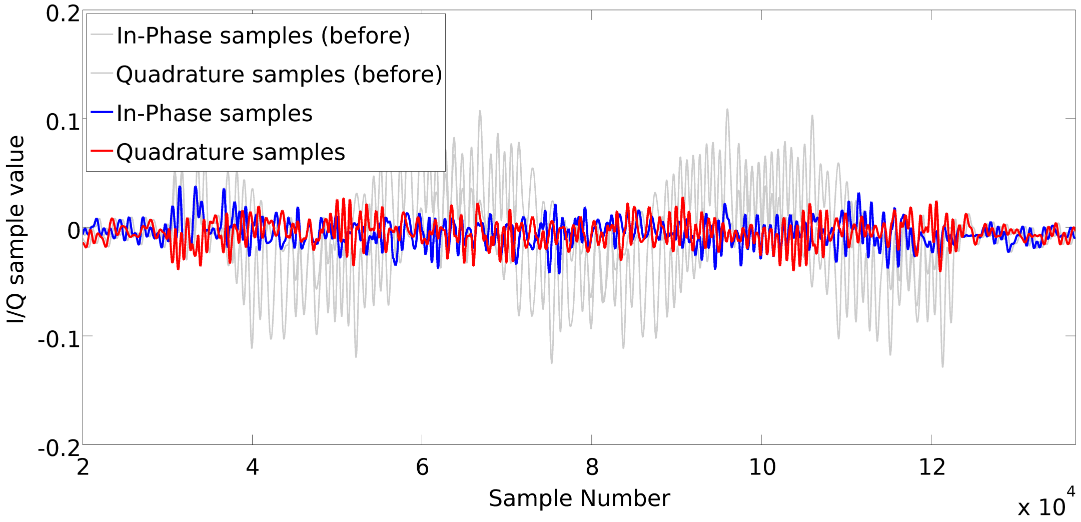


Figure 6.9: Remaining signal after cancellation

However, to be able to use the cancellation method for collision resolution, it must be able to cancel out packets when there are no collisions. Figure 6.1 shows a packet that can be cancelled out by the method. Signal attributes are estimated like discussed above and a signal approximation is generated (figure 6.8) that is used to subtract from the original signal. The result is visible on figure 6.9. We can observe that the signal power is greatly reduced and is comparable to noise. There is, however, some residual signal, but this is due to the difference of estimation and the real signal. Improving the quality of the signal estimation is a possible future work improvement.

6.5 Packet collision resolution

The main application of packet cancellation is packet collision resolution. In areas with lots of air traffic (around busy airports) it commonly happens that two aircraft transmit at roughly the same time. This results in both signals being added together. In cases where both signals have similar amplitude and/or are completely or mostly overlapped, the receiver has a very little chance of successfully decoding even one of the packets. However, there are cases when one of the packets is quite stronger than the other, as it is often encountered when aircraft are at very different distances (one very far, one very close) from the receiving antenna. The signal received from the closer aircraft[2] is much stronger and the receiver is able to successfully decode the packet. In conventional receivers, the weaker packet would be lost because it was overridden by the stronger packet. This opens up a potential for collision resolution by employing packet cancellation.

6.6 Results, discussion and future improvements

As a proof of concept, lets observe the results on a synthetic collision of two packets on figure 6.10, where a stronger and a weaker packet have collided. The receiver was able to successfully decode the stronger packet in the first

[2] Due to free-space signal loss, the further-away aircraft has much lower signal, and similarly, the closer aircraft produces much stronger signal.

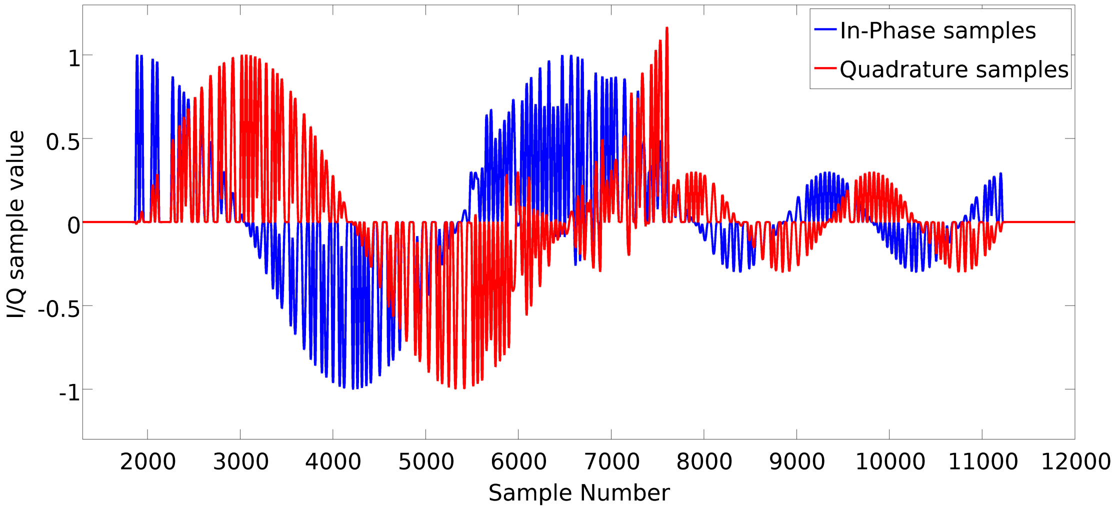


Figure 6.10: Packet collision - Stronger (first) packet collided with a weaker (second) packet

pass. The packet payload data and estimations of signal attributes were used to generate an estimation of the signal of the first packet. Then, packet cancellation method described in previous section was applied. The result can be seen on figure 6.11. We can observe that the amplitude of the stronger (first) packet has been greatly reduced. Furthermore, the weaker packet is now fully exposed and in the second pass our receiver was able to decode the payload successfully. This shows that packet cancellation techniques can be successfully applied in the sense of packet collision resolution.

Because we shown that collisions can be resolved in certain scenarios, and have also shown that our cancellation techniques work on real world packet signals[3], we applied our cancellation procedure to every successfully decoded packet from the captured trace. After cancellation, the trace was processed again through the receiver chain. Unfortunately, we have observed no *new* successfully decoded packets after the second pass (after cancellation). This indicates that the developed method was unable to resolve collisions between

[3] *Packet cancellation* section show cancellation on real world packet signal.

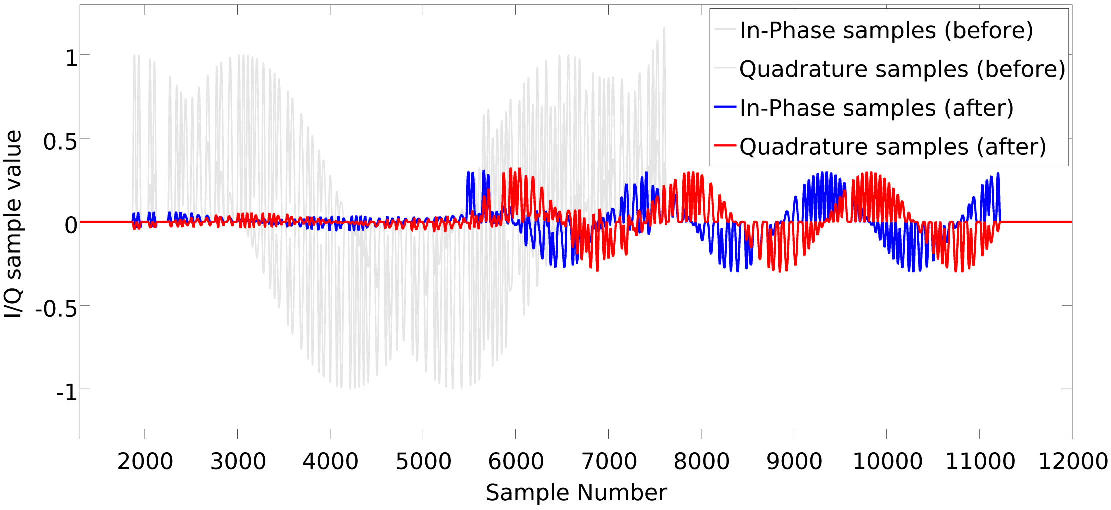


Figure 6.11: Resolved packet collision - second packet now visible and decodable

real world packets. We are certain that the lack of success is due to rather simplistic cancellation technique. This goes to show that generating a cancellation signal i.e. reconstructing a signal that closely matches the one that was transmitted, is a complex task. We have noticed multiple deviations from our model in real world packets.

Firstly, the amplitude peak height can vary quite significantly and sporadically throughout the packet. Use of a constant amplitude estimate works well on certain (well behaved) transmitters, but fails on some. We have considered that amplitude peak values gathered in the timestamping process could be used to reconstruct the amplitude on a per-peak basis, but this approach would fail when the peak values are influenced by the signal of the other colliding packet.

Additionally, for certain transmitters it is very hard to estimate the phase and frequency offsets, due to presence of strong phase noise. In certain cases it is significant enough that it can be used to fingerprint aircraft or transponder models based on it, as it was demonstrated in the article [18].

It is important to note that as long as the amplitude of the signal is formed correctly, the transmitter complies with the specifications. Phase noise is therefore not a problem for decoding the packets, however, it makes the task of reconstructing the signal significantly harder. In the future, we consider developing multiple phase estimation models and a heuristic that would allow us to choose the best model on a per aircraft basis. The best fitting model, once determined, could be remembered and recalled when a new packet from the same aircraft is encountered and processed for cancellation.

Another reason to why the developed cancellation method was unsuccessful can be sought in the fixed pulse template used to construct the pulses. Due to relatively permissive specifications, there are appreciable differences in pulse shapes between different transmitters. We believe that employing a customized pulse template that suit only a particular aircraft or transmitter model would greatly reduce the residual packet signal left behind after cancellation. This goes hand in hand with gathering and learning transmitter signal properties proposed in section 3.3 and plays well with the evolvable design of our receiver.

On top of the discussed things, there might still be an unknown phenomenon at play in the real world data that was not considered while producing synthetic data. This poses a new challenge and presents an opportunity that is worth investigating in future research work.

It is also worth investigating the possibility of performing partial cancellation i.e. performing cancellation only on a part of a signal. This would alleviate the requirement that the packet that is considered for cancellation is decoded correctly. Given a facility for identifying the aircraft without successfully decoding the payload, we could leverage previously learnt information about the aircraft's signal properties (we would know the preamble and at least the unique address of the aircraft) to perform partial cancellation. On top of that, collision resolution could be performed iteratively, gaining more and more information at each iteration, in turn gaining enough information to successfully decode one of the colliding packets. This could be used to resolve even the type of collisions, where none of the packets have been successfully decoded, making our receiver not only evolvable but also cognitive.

Chapter 7

Conclusion and future work

In this thesis we presented an advanced software-defined receiver, capable of receiving and decoding ADS-B/Mode-S packets that are emitted by aircraft and used by traffic control. We described in-detail the components of the software receiver chain and laid out the basis for researching, designing, implementing and evaluating advanced features not present in state of the art ADS-B/Mode-S receivers. These features include high precision packet timestamping (that can be used in conjunction with multilateration algorithms for passive aircraft position tracking) and packet collision resolution by means of packet signal cancellation. We have specifically designed our receiver to be extendable and evolvable and have implemented the two advanced features to demonstrate that. Additionally, we have presented our vision of a cognitive receiver and stated the features and improvements that would make it so, specifically the cognitive cache that would hold information about each aircraft/transmitter and would allow for development of iterative decoding procedures and could provide the necessary data for timestamping using only partial packet payload data.

To show the evolvability and extensibility of our receiver we implemented several timestamping methods. This includes correlation-based methods and our novel approach of peak-position-based method, that is not only rather simple and elegant, but also relatively computationally efficient. We have shown that our peak-position-based timestamping method can achieve timestamp error variance of 2.2 ns (equivalent to 66 cm at the speed of light) and when considering the packet strength classes, down to awe-inspiring 1.2 ns, which is equivalent to 36 cm at the speed of light. This would allow for high-precision passive positional pinpointing and tracking of aircraft when coupled with appropriate downstream algorithms.

Furthermore, we have investigated packet collision resolution. We have developed a proof of concept technique for packet signal cancellation by estimating physical signal attributes and recreating the packet signal from them. We have shown that the present implementation of our packet cancellation method works in a synthetic (proof of concept) environment, but struggles on real world data. This limited success allowed us to gain invaluable insight into the complex problem of packet signal cancellation. We have realized that signal model needs to be built separately for each individual aircraft. This goes hand in hand with the proposed future development of transmitter profile cache that would allow the receiver to learn the attributes of individual transmitter and apply cancellation techniques accordingly. On top of that we have argued about the methods (suitable for future research) that would perform iterative and partial decoding and allow timestamping and cancellation techniques to be used on partial data and thus alleviate one of the main shortcomings of traditional receivers and making a step towards a cognitive receiver for ADS-B/Mode-S messages.

Bibliography

[1] Gnuradio mode-s/ads-b radio, `https://github.com/bistromath/gr-air-modes`, accessed: 2016-11-17.

[2] Gnu radio, `http://gnuradio.org/`, accessed: 2016-11-17.

[3] R. Kaune, C. Steffes, S. Rau, W. Konle, J. Pagel, Wide area multilateration using ads-b transponder signals, in: Information Fusion (FUSION), 2012 15th International Conference on, IEEE, 2012, pp. 727–734.

[4] G. Galati, M. Leonardi, I. Mantilla-Gaviria, M. Tosti, Lower bounds of accuracy for enhanced mode-s distributed sensor networks, IET Radar, Sonar & Navigation 6 (3) (2012) 190–201.

[5] ICAO, Aeronautical Telecommunications Volume IV, Surveillance and Collision Avoidance Systems - annex 10 to the convention on international civil aviation, 2007.

[6] G. Baldini, T. Sturman, A. R. Biswas, R. Leschhorn, G. Godor, M. Street, Security aspects in software defined radio and cognitive radio networks: A survey and a way ahead, IEEE Communications Surveys & Tutorials 14 (2) (2012) 355–379.

[7] L. Michael, M. Mihaljevic, S. Haruyama, R. Kohno, Security issues for software defined radio: Design of a secure download system, IEICE Transactions on Communications E85-B (12) (2002) 2588–2600.

[8] S. Grönroos, K. Nybom, J. Björkqvist, J. Hallio, J. Auranen, R. Ekman, Distributed spectrum sensing using low cost hardware, Journal of Signal Processing Systems 83 (1) (2016) 5–17.

[9] C. J. Bernardos, A. De La Oliva, P. Serrano, A. Banchs, L. M. Contreras, H. Jin, J. C. Zúñiga, An architecture for software defined wireless networking, IEEE wireless communications 21 (3) (2014) 52–61.

[10] R. Berezdivin, R. Breinig, R. Topp, Next-generation wireless communications concepts and technologies, IEEE Communications Magazine 40 (3) (2002) 108–116.

[11] F. Guerriero, V. Loscrí, P. Pace, R. Surace, Neural networks and sdr modulation schemes for wireless mobile nodes: A synergic approach, Ad Hoc Networks 54 (2017) 17–29.

[12] Radarcape - ads-b receiver, `http://www.modesbeast.com/radarcape.html`, accessed: 2016-11-29.

[13] E. G. Piracci, G. Galati, M. Pagnini, Ads-b signals reception: a software defined radio approach, in: Metrology for Aerospace (MetroAeroSpace), 2014 IEEE, IEEE, 2014, pp. 543–548.

[14] Flightaware - flight tracker / flight status / flight tracking, `https://flightaware.com/`, accessed: 2017-09-04.

[15] Flightradar24.com - live flight tracker!, `https://www.flightradar24.com`, accessed: 2017-09-04.

[16] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, M. Wilhelm, Bringing Up OpenSky: A Large-scale ADS-B Sensor Network for Research, in: Proceedings of the 13th International Symposium on Information Processing in Sensor Networks, IPSN '14, Berlin, Germany, 2014, pp. 83–94.

[17] dump1090, https://github.com/antirez/dump1090, accessed: 2016-11-21.

[18] D. Moser, P. Leu, V. Lenders, A. Ranganathan, F. Ricciato, S. Capkun, Investigation of multi-device location spoofing attacks on air traffic control and possible countermeasures, in: ACM Conference on Mobile Computing and Networking, MOBICOM, 2016.
URL http://lenders.ch/publications/conferences/mobicom16.pdf

[19] M. Schäfer, V. Lenders, J. B. Schmitt, Secure Track Verification, in: IEEE Symposium on Security and Privacy (S&P), S&P '15, San Jose, CA, USA, 2015.

[20] M. Strohmeier, I. Martinovic, On passive data link layer fingerprinting of aircraft transponders, in: Proceedings of the 1st ACM Workshop on Cyber-Physical Systems-Security and/or Privacy (CPS-SPC), 2015.

[21] M. Eichelberger, K. Luchsinger, S. Tanner, R. Wattenhofer, Indoor localization with aircraft signals.

[22] G. Galati, M. Leonardi, P. De Marco, L. Menè, P. Magarò, M. Gasbarra, New time of arrival estimation method for multilateration target location, Proc. JISSA.

[23] R. Calvo-Palomino, F. Ricciato, D. Giustiniano, V. Lenders, Ltess-track: A precise and fast frequency offset estimation for low-cost sdr platforms, in: Accepted for the 11th ACM Workshop on Wireless Network Testbeds, ACM WiNTECH'17, 2017.

[24] F. Ricciato, S. Sciancalepore, G. Boggia, Tracing a linearly moving node from asynchronous time-of-arrival measurements, IEEE Communications Letters 20 (9) (2016) 1836–1839.