Synchronisation of MPEG-2 based digital TV services over IP networks

Master Thesis project performed at Telia Research AB

by

Björn Kaxe

Preface

This Master Thesis in Electrical Engineering has been carried out at Telia Research AB, Communication Services, Farsta, from May 1999 to January 2000.

I would like to thank my supervisor at Telia, Per Tholin, for his assistance, patience and encouragement and also for interesting discussions with him as well as with Mats Ögen which have helped me during this period. I would also express my gratitude to Bo Sjöberg, Gunnar Betnér and Per Ola Wester. Many thanks to my roommate Fredrik Ydrenius who has put up with me for more than seven months.

Finally, I would like to thank my examiner at RIT, Department of Teleinformatics, Gunnar Karlsson for reading my report one last time and thereafter giving me valuable ideas in order to improve it.

Abstract

This thesis deals with the problem of handling delay variations of MPEG-2 audio-visual streams delivered over IP-based networks. The focus is on high quality digital television applications. A scheme to handle delay variations (jitter) has been designed and evaluated by simulations. The results have been compared to the expected requirements of an MPEG-2 decoder and an ordinary consumer TV set. A simple channel model has been used to simulate the IP-based network, where the jitter process is uniformly distributed with a peak-to-peak delay variation of 100 ms. The main focus on the scheme is where the MPEG-2 decoder is "fully" synchronised, i.e. there is a nominal constant delay from the A/D converter to the D/A converter.

From simulations it has been shown that it is possible to design a dejittering scheme capable of filtering 100 ms of peak-to-peak IP-packet delay variation, producing a residual jitter amplitude in the order of a microsecond. Such a low jitter amplitude is obviously well below the MPEG-2 RTI specification of ±25 μs. The scheme also matches the performance requirements that can be expected of a consumer TV set. It has also been shown that it is possible to combine an extreme low-pass filtering with a sufficiently small additional delay added by the dejittering scheme.

If the scheme is to be implemented in a real system some further investigations have to be made, especially concerning issues around real time support of common operating systems.

1 Introduction

1.1 Overview

In this section, an introduction to this thesis "Synchronisation of MPEG-2 based digital TV services over IP networks" will be given. First of all, a background to the problem will be presented. Then an introduction to the problem follows and the purpose of the thesis will be described. Finally, an overview of the structure of the thesis with reading instructions will be given.

1.2 Background

Already in the late 19th century the research in representing images with electrical signals began. In 1897 the cathode ray tube was invented, which still is the most widely used technique in TV sets and computer monitors. But the possibility of transmitting audio-visual information first became possible with the arrival of television in the early thirties. The first television broadcast took place both in Berlin and Paris in 1935 and the first public television service was started in New York in 1939. In the forties, television services started in more and more countries in Europe, but each country developed its own standard. It was not until 1952 that a single standard was proposed and progressively adopted for use in Europe. Now modern television was born [Peters 85].

Apart from gradually improving quality of sender and receiver equipment, three major innovations have characterised the development of television since the fifties: the introduction of colour television in the mid-fifties, high definition television in the late seventies, and digital television in the nineties.

One major problem with analogue television is its high demand of bandwidth. Thanks to advanced image coding, data compression techniques and digital representation this bandwidth can be significantly reduced. Typically, about six digital TV channels fit into the bandwidth of a single analogue TV channel. One major advantage of digital television over analogue, apart from the reduced bandwidth, is the possibility of interaction between the receiver and the sender.

Today digital television is delivered over dedicated broadcast networks, by satellite, cable and terrestrial transmission. The most widely used video coding standard used in these networks is MPEG-2. It is for example used in the DVB standard for broadcasting of digital television, which are the most widely used standards in Europe, but is also used in storage of digital video for example on DVD.

Today's broadcast transmission methods give almost no interactivity to the viewers. To enable some sort of interactivity, the networks have to provide support for an information flow from the receiver to the sender. Therefore, there is a large interest in providing new, interactive TV services over data communications networks, like IP networks.

In order to provide interactive TV services over data communications networks, a lot of work has been done during the nineties around QoS issues. Especially, ATM networks have been studied in this as respect. An overview of the issues of asynchronous transfer of video over packet switched networks is given in [Karlsson 96].

Since the Internet has grown and developed enormously in the last few years, one can expect that in a near future more services, like high quality digital television, will be offered beyond the usual data transmission that the Internet was first designed for. An Internet provider can provide both broadband connection to the Internet, digital television and IP-telephony on the same cable.

The transmission of digital television over IP-based network will provide opportunities for interactive services for the viewers, for example video on demand, (where the viewer decides when to watch a certain movie or TV program).

There are some problems with real time transmission of audio-visual information over IP based networks because these types of networks were not designed for those sorts of applications. But today there is ongoing work to support real-time services over IP-based networks. There exist some real time streaming products for audio and video over IP, like Real Player, but they do not provide the quality required for high quality digital television.

1.3 Introduction to the Problem

As mentioned earlier, IP-based networks were not initially designed for real time transmission of audio-visual information. Traditionally IP-based networks behave as classical packet switched networks, providing no guarantees regarding delivery of the information on a "network level". When the network is heavily loaded, i.e. congested, some data may be lost or significantly delayed during the transmission. Audio-visual data are generally vulnerable to data loss because the coding techniques used, for example the most commonly used subsets of MPEG-2, generate bitstreams with limited resilience to packet losses. Another major problem is that end-to-end delay is variable, which depends on the load of the network. In order to deliver MPEG-2 audio and video streams in real time with high quality, these delay variations have to be reduced at the receiving end, or the decoder will not operate correctly. This problem will be explained in later parts.

This thesis will deal with the problem of delay variations of MPEG-2 audio-visual information delivered over IP-based networks. In this thesis a scheme to handle delay variations will be presented, which will restore the packet intervals of an MPEG-2 stream, delivered over an IP network. It will be mainly aimed at multicast applications of digital television where MPEG-2 audio and video are streamed in real-time. The scheme should be implementable in software in a set-top-box or on an ordinary PC. It should work both with constant and variable bit rate coded MPEG-2 streams.

1.4 Delimitation

The designed scheme will not be implemented in a real set-top-box or on a computer due to limited amount of time. It will be evaluated by simulations only.

It is not the purpose of this thesis to characterise and model delay variations of real IP networks, and create a realistic channel model. Instead, a very simple channel model, that can illustrate a "worst case" scenario will be used in the simulations.

In the simulations an assumption that the MPEG-2 streams are delivered over an ordinary 10 Mbit/s Ethernet interface, is made.

1.5 Structure of the Report

In the first sections, Sections 2 and 3 of this thesis, the basics of analogue video signals and parts of the MPEG-2 standard will be described. These parts are crucial in the understanding of why delay variation is a problem in real-time streaming of video. A short overview of video coding according to MPEG-2 will also be given in Section 3.

After that, in Section 4, a description of IP networks and why delay variations occur in these networks, is given. In the same section all protocols that a real system is assumed to use will be briefly described. Then, in Section 5, the concept of real-time streaming will be defined and explained. In Section 6, a more thorough description of the problem is presented and in the same section an overview of the research field will be given. Thereafter, in Section 7, a mathematical description of the problem will be provided and in the same section the proposed scheme will be described. In Section 8 the simulations of the proposed scheme is presented and some conclusions are made from these simulations. Section 9 further discusses the results and provides some more general conclusions. In addition, some recommendations on future work will be given.

2 Analogue video

2.1 Overview

In this section there will be a description of how an analogue video signal is built up. This is crucial in the understanding of the problem of synchronisation of video signals and other problems investigated in this thesis.

2.2 Video Signal

An analogue television picture is built up of lines. In the PAL standard the number of lines per frame is 625 while in NTSC it is 525. These pictures or frames are updated with a certain frequency. In Europe it is standardised to 25 Hz whereas in USA it is 30 Hz.

2.2.1 Monochrome Video Signal

In Figure 2.1 is shown how a TV frame is "drawn" on the TV screen when the traditional television-picture tube is being used, see [Enstedt 88]. In the tube an electron gun is firing electrons on a fluorescent material which emits light when it is exposed to the electrons. The electron ray draws each line by moving from left to right. When a whole line has been drawn on the screen the electron ray is moved back quickly to the left in order to start drawing the next line. When this movement (line return) is made the electron ray must be blanked in order not to make this visible on the screen. Therefore so-called line blanking pulses must be put into the video signal. In Figure 2.1 these line returns are shown with dashed lines and active lines are solid. Each line is drawn on the screen in turn from the upper left corner down to the lower right one, which is also shown in the figure.

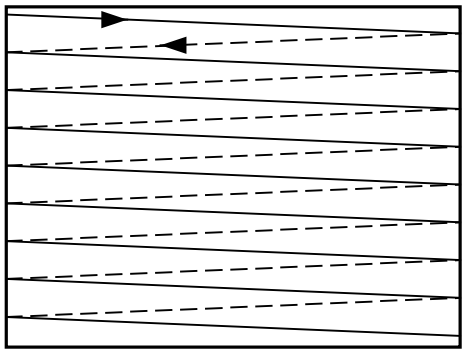

Figure 2.1 Line drawing and line return

There is also another type of blanking pulses which is used for the vertical return, that is when a new picture is to be drawn on the screen, called picture blanking pulses.

To make it possible for the TV to know when to make line returns as well as picture returns, so-called synchronisation pulses are put in the video signal, line and picture synchronisation pulses, respectively. These synchronisation pulses are put in the blanking intervals. Figure 2.2 shows how a monochrome video signal is built up with blanking and synchronisation pulses. The figure shows the last three lines in a picture and the two lines in the following picture. The figure is highly simplified and only aims

at giving an idea of where the blanking and synchronisation pulses are put in the video signal. In reality the picture synchronisation consists of many short pulses.

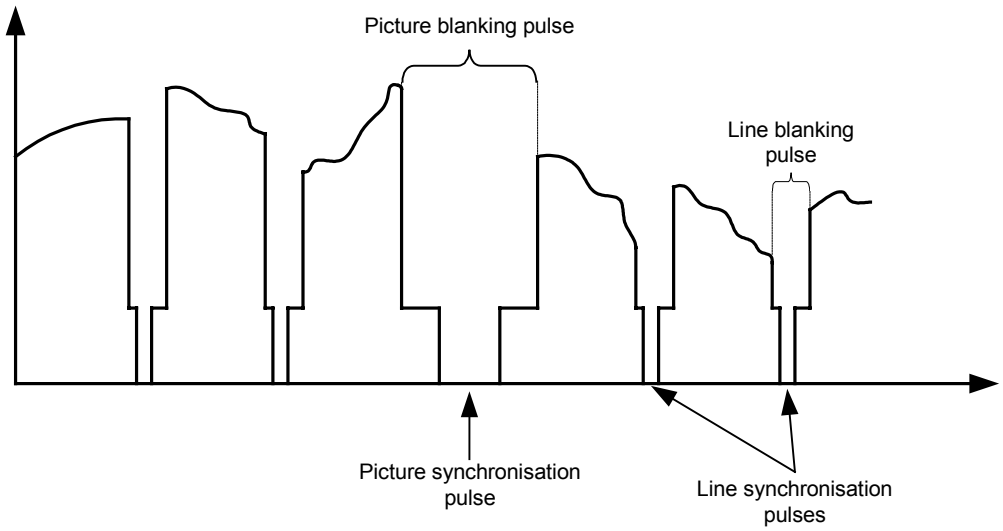

Figure 2.2 Monochrome video signal

As mentioned earlier the frame update frequency in Europe is 25 Hz. At such a low frame rate the flicker in the TV picture is annoying. A way to solve this problem would be to increase the number of updates per second, to say 50 Hz. Principally there are no obstacles to do that, but it would result in some practical problems. One problem would be that the bandwidth of the video signal has to be increased. In TV transmission techniques, another way to solve this problem is used. This is called interlace. In interlace a frame is displayed as two fields, one consisting of the odd lines and the other one of the even lines. Illusory, this will give an update frequency of 50 Hz, without increasing the number of lines per second (in 25 Hz, PAL the number of lines per second is 15625).

2.2.2 Composite Colour Video Signal - PAL

So far, only the monochrome video signal is described. A monochrome video signal has a bandwidth of about 5 MHz. In the frequency spectrum of the monochrome signal there are some unused regions, which are used for the colour information. To do this a modulation method with a so-called sub-carrier is used. To make it possible for the oscillator of the TV to synchronise to this carrier frequency a colour synchronisation burst is put in the video signal. This signal is made up of 9-11 periods of an unmodulated colour carrier wave with a fixed phase. This is inserted into each latter part of the line blanking pulses after the line synchronisation pulses in the colour video signal, see Figure 2.3.

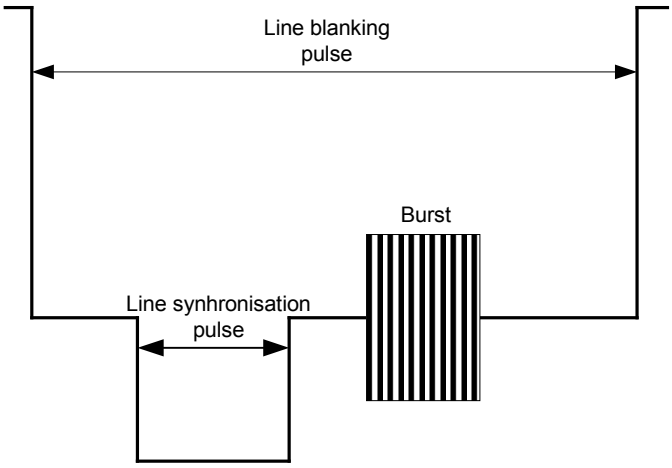

Figure 2.3 The position of the burst in the line blanking interval.

2.2.3 Component Video Signals

The last section described how the monochrome video signal was extended with colour information. This type of signal is called a composite signal since all information, including the luminance, the chrominance, and the synchronisation information, are contained in the same signal. In this type of signal the chrominance information actually consists of two components called U and V, whereas the luminance component is called Y. The video signal can then be represented by three or four separate signals Y, U, and V, and potentially a separate synchronisation signal. This format is called component format.

A more commonly used format than Y, U, V is R, G, B (Red, Green, Blue), which for example is provided in a scart-connector of a modern TV set. One of several reasons to use this type of signal is that a typical colour video camera optically captures these three colour components.

2.2.4 Requirements of a Video Signal

In order to make the TV display the video signal correctly, the receiver has to synchronise to the line and picture synchronisation pulses, respectively. The TV also has to synchronise to the colour sub-carrier frequency to extract the colour information correctly. For the TV to do so the video signal has to be accurate and stable in frequency.

The ITU-R recommendation [ITU-R 624] specifies different frequency and phase requirements for video signals. These requirements are the minimum a receiver should handle.

The accuracy and stability requirements for the colour sub-carrier are the most stringent and therefore they will be discussed below.

The central sub-carrier frequency of PAL-B is 4.43361875 MHz. The frequency requirements of the colour sub-carrier for PAL-B, specify a tolerance of ± 5 Hz (which corresponds to ± 1 ppm). This requirement defines the minimum accuracy of the oscillators for the modulators and thus the minimum range a receiver should handle. There are also requirements for the short- and long-term frequency variations. The

maximum short-term variation for a PAL-B signal is 69 Hz within a line. This corresponds to a variation of the colour frequency of 16 ppm/line. If this requirement is satisfied, we can get a correct colour representation for each line. The maximum long-term frequency variation (also called clock drift) a PAL signal must meet is 0.1 Hz/s.

It should be noted that these requirements are stated for broadcast equipment. If the signal is to be displayed on a consumer TV set, these requirements can be reduced significantly, [Andreotti 95]. In fact, home receivers can handle a much wider range of frequency deviation and drift while ensuring good quality likely in the region of 100 ppm deviation. However, such figures are not standardised.

3 Video Coding

3.1 Overview

First in this section, a short description of some video compression methods that are used in modern audio-visual coding standards will be given. Then, some standards, which are used today, are mentioned. After that, the generic audio-visual coding standard MPEG-2, which is used in this thesis, will be treated in more detail. The details of the video compression methods used in MPEG-2 will not be mentioned, and only the so-called MPEG-2 Systems Layer, which is responsible of synchronisation and multiplexing, will be described.

3.2 Background

A high quality digital version of a 25 Hz video signal is typically made up of 576 lines of 720 pixels. The video signal is normally divided into one luminance component called Y and two chrominance components U and V, see Section 2.2.3. One common way of digitising an analogue video, that is suit for TV broadcasting quality requirements, is to sample the luminance with all 720 pixels per line, while the chrominance components are subsampled by a factor of 2, giving 360 pixels per line. The resolution of the samples is normally 8 bits and this gives an average of 16 bits per pixel of all 720 pixels per line. This will give a data rate of approximately 170 Mbit/s (576*720*25*16 bits ≈ 170 Mbit/s). An ordinary movie of 1.5h would then use approximately 115 GB of storage space. This is an enormous amount of data and most storage media cannot store this amount. Neither can it deliver it at such high transfer rates. Some sort of compression has to be used in order to keep cost down. It is a fact that video sequences contain a lot of both statistical and subjective redundancy. There are several ways to compress video signals, both in temporal and spatial domain, while causing very limited reduction in quality.

3.3 Video Compression Methods

In a sequence of still pictures making up a video signal, much of the picture-area, e.g. the background, will remain the same, while objects may move around. Instead of encoding each frame individually, it makes sense to utilise the frame by frame correlation by using a temporal prediction. The previous frame may then be used to "guess" the current frame. However, since some areas have moved, a motion compensation is added to the temporal prediction, improving the performance of the predictor. This coding method is often referred to as motion compensated temporal prediction. It is one part in many modern coding techniques, like MPEG.

There are also spatial methods to reduce the redundancy in the pictures. Usually, the frames are transformed into the frequency domain using the Discrete Cosine Transform (DCT), where the frequency components can be manipulated. For example, high frequency components of the frames usually have low amplitudes and can be discarded with almost no perceivable loss of quality.

After using these two methods, an entropy-coding algorithm is used, for example Huffman encoding that takes advantage of the statistical distribution of the bits in the

data steam. These methods can reduce the bit rate without any loss of information, while the two other operations above loose information in the encoding process.

Current compression algorithms combine all of these methods into what is called hybrid coding and this class of algorithms is used for example in MPEG-2. The interested reader can find further information in [Forchheimer 96].

3.4 Video Coding Standards

There exist many video-coding standards, like H.263 and its predecessor H.261 that is used for videoconference applications and MPEG-2 that is used for higher quality applications.

MPEG-4 is a new standard that uses a lot of new compression methods. It is supporting very low bit rates down to 5 kbit/s. This is particularly interesting in mobile networks applications, like video-conference over cellular phones.

3.5 The MPEG-2 Audio-Visual Coding Standard

In 1988 the MPEG (Moving Pictures Experts Group) committee was started. The immediate goal of the committee was to find a standardisation of video and audio on CD-ROMs. This resulted in the MPEG-1 standard in 1992. The MPEG-1 standard is optimised to a data rate of about 1.4 Mbit/s. This data rate will give a quality comparable to an ordinary VHS video tape recorder. A shortcoming of the MPEG-1 standard is that it lacks specific support for interlaced formats, explained in Section 2.2.1.

In 1994 the MPEG-2 standard was finished. Its main purpose was the transmission of TV quality video, but now includes supports for High Definition Television (HDTV) as well. This standard is an extension of the MPEG-1 standard and supports interlaced formats and a wider range of data rates from less than 1 Mbit/s to 100 Mbit/s.

MPEG-2 can be used and is used in many applications, such as videoconference, satellite TV and DVD because of its generality. Today MPEG-2 is the leading standard in broadcasting of digital TV.

As mentioned above, MPEG-2 uses a hybrid coding technique, including both temporal prediction and transform coding. The details of the compression techniques of MPEG-2 will not be examined further. The interested reader can read more in [Haskell 96].

3.5.1 MPEG-2 Systems Layer

The MPEG-2 standard is divided into two main layers:

- Compression layer (includes audio and video streams)
- Systems layer (including timing information to synchronise video and audio as well as multiplexing mechanisms)

The Compression layer handles compression of the audio and video streams. The processing of this layer generates so-called elementary streams, (ES). This is the output of the video and audio encoders.

The Systems layer in MPEG-2 is responsible for combining one or more elementary streams of video and audio as well as other data into one single stream or multiple streams, which are suitable for storage or transmission. The Systems layer supports five basic functions, see [MPEG2 Sys]:

- synchronisation of multiple compressed streams on decoding,
- interleaving of multiple compressed streams into a single stream,
- initialisation of buffering for decoding start up,
- continuous buffer management,
- time identification.

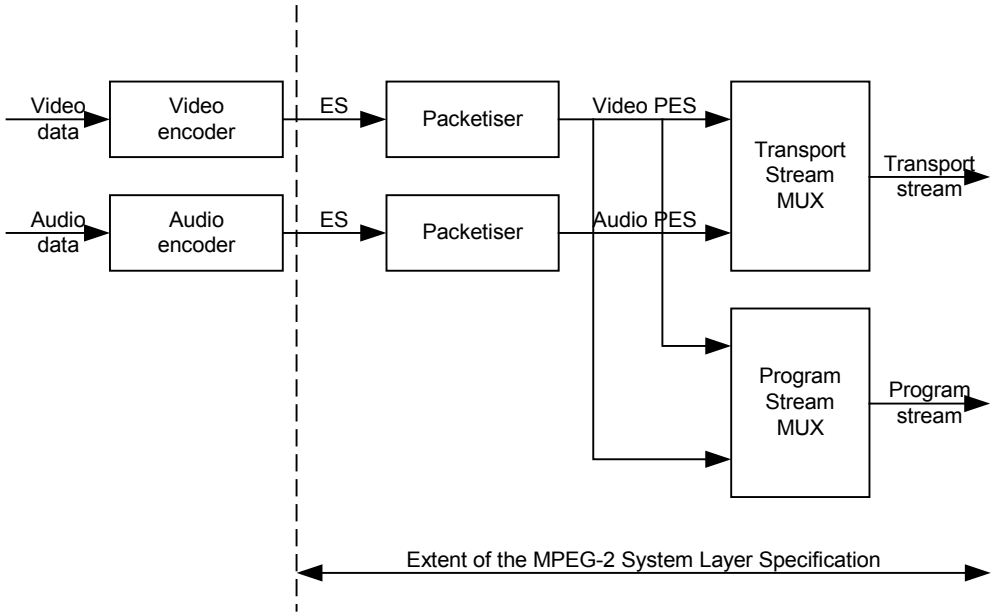

Figure 3.1 Model for MPEG-2 Systems in an implementation, where either of the Transport stream or the Program stream is used, [MPEG2 Sys].

A model of the Systems layer on the encoding side is shown in Figure 3.1. Each elementary stream, generated by the video and audio encoders, are first mapped into so called packetised elementary stream (PES) packets, see Figure 3.2.

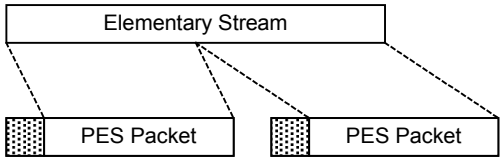

Figure 3.2 Mapping of ES into PES.

The headers in the PES packets hold among other things timing information when to decode and display the elementary stream. Another rather important functionality is the

possibility to indicate the data rate of the stream, which is used to determine the rate at which the stream should enter the decoding system.

The packetised elementary streams (PES) are multiplexed into either a program stream (PS) or a transport stream (TS), see Figure 3.1. A program stream supports only one program, whereas a transport stream may include multiple programs. Elementary streams of a single program typically share a common time base. The time base is the clock that determines among other things the sampling instances of the audio and video signals and is used when the elementary streams are generated. A program can for example be a television channel including a video stream and an associated audio stream.

In program streams, only elementary streams with common time base are multiplexed. Program streams are designed for use in almost error-free environments and are suitable for applications, which may involve software processing. Program stream packets may be of variable and relatively great length.

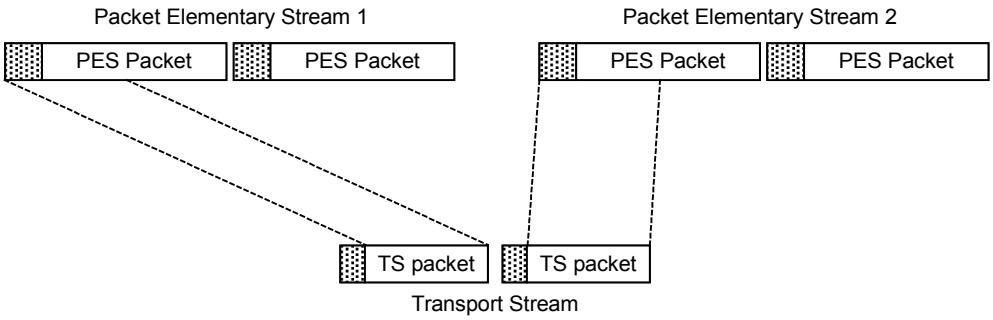

Figure 3.3 Mapping of two PES packets into one TS packet.

Both elementary streams with common time base (programs) and elementary streams with independent time base can be multiplexed into transport streams. Transport streams are designed for use in environments where errors are probable, such as storage or transmission in lossy or noisy media. Transport stream packets are fixed size, 188 bytes long.

3.5.2 MPEG-2 System Clock

When the sampling and encoding is done in the video and audio encoders, a sampling clock called system time clock (STC) is used. It has a frequency of 27 MHz ± 30 ppm. The STC is normally synchronised to the line frequency of the incoming analogue video signal. The STC is represented by a 42-bit counter. Two types of time stamps derived from this clock is inserted in the PES, presentation time stamps (PTS) and decoding time stamps (DTS). The PTS indicates to the decoder when to display the contents of the PES. The DTS indicates to the decoder when to remove the contents of the PES from the receiving buffer and decode it. These time stamps have to be inserted in the PES with an interval not exceeding 0.7 seconds.

3.5.3 System Clock Recovery

The decoder side has its own version of the STC, which is used in the decoding process of the audio and video streams. This clock has to be synchronised with the STC of the encoder side or the buffer of the decoder will over- or underflow. To do so the decoding

system may recover the frequency of the STC of the encoder. In order to do so, time stamps of the STC is inserted in the transport stream or the program stream, that the decoder side can extract. In the TS case these time stamps are called program clock reference (PCR) and in the PS case system clock reference (SCR). The TS can include many programs with its own time base and therefore separate PCRs for each of these programs have to be included in the TS. The SCR has to be sent with a maximum interval of 0.7 seconds, while the PCR has to be sent at least every 0.1 seconds.

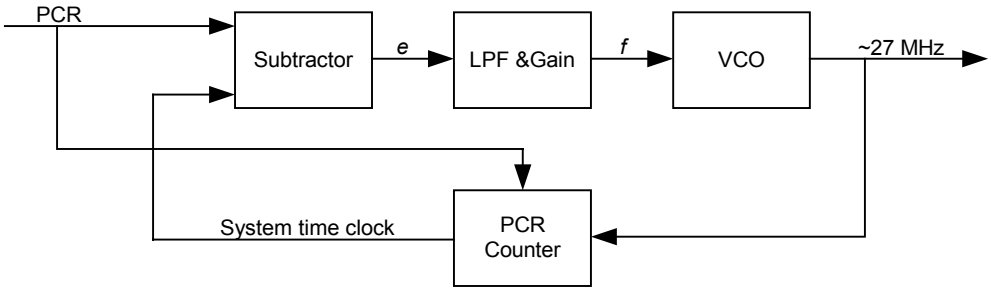

Figure 3.4 Clock recovery in MPEG-2 decoder, from [MPEG2 Sys]

Typically a digital phase-locked loop (DPLL), see [Best 93], is used in the MPEG-2 decoder to synchronise the clock of the decoder to the STC of the encoder. A simple PLL is shown in Figure 3.4. It works as follows: Initially, the PLL waits for the first PCR to arrive. When the first PCR arrives it is loaded to the PCR counter. Now the PLL starts to operate in a close loop fashion. Each time as a PCR arrives it is compared to the current value in the PCR counter. The difference gives an error term *e*. This error term is sent to a low pass filter (LPF). The output from the LPF, *f*, controls the frequency of the voltage-controlled oscillator (VCO) whose output provides the system clock frequency of the decoder. The output of the VCO is sent to the PCR counter. The central frequency of the VCO is approximately 27 MHz. After a while the error term *e* converges to zero which means that the DPLL has been locked to the incoming time base.

The requirements on stability and frequency accuracy of the recovered STC clock depend on the application. In applications, where the output from the decoder will be D/A converted to an analogue video signal, the STC clock is directly used to synchronise the signal. The colour sub-carrier and all synchronisation pulses will be derived from this clock, see Section 2.2. In this case the STC must have sufficient accuracy and stability so that a TV set can synchronise correctly to the video signal. In other applications, for example when the decoder is built into a video card in a computer and the output will be displayed on the computer screen, the video signal feeding the computer monitor normally is not synchronised to the STC, but uses a free running clock.

4 Network & Protocols

4.1 Overview

This section will describe the behaviour of a packet switched network and the problems that occur when real time audio-visual information is streamed over these types of networks. After that an overview of the protocols that a real system is assumed to use is given. The end of this section will describe how MPEG-2 is to be transmitted over IP-based networks.

4.2 Packet Switched Networks

4.2.1 Introduction

Communication networks can be divided into two basic categories: circuit-switched and packet-switched. These classifications are also sometimes called connection oriented and connectionless.

In circuit-switched networks dedicated connections are formed between peers that want to communicate. The existing telephone networks are typical circuit-switched systems. One advantage of these types of networks lies in its guaranteed capacity: once a connection is established, no other network activity will decrease its capacity. On the other hand, this can also be a disadvantage: even if the communicating peers do not transmit any information at the moment, the guaranteed capacity is kept by them.

Packet-switched networks take an entirely different approach. When data are to be transferred over a packet-switched network, they are divided into small pieces called packets. These packets also carry identification information, which enables the network nodes to send them to the intended destination. One advantage of these networks compared to circuit-switched networks is that they use the available capacity more efficiently. All communicating peers share the same capacity. However, when the number of communicating peers grows, each one will get a smaller share of the available capacity.

4.2.2 Delay Variations

When packets are sent over packet-switched networks the delay will vary over time. This means that the original inter-packet interval of the stream will not be maintained, but a delay variation will be introduced. There are many different reasons why these delay variations occur. The load on the networks varies over time, which may cause a time varying fullness of the queues of the routers or switches present in the end-to-end path. The source itself can also introduce some delay variations in the output stream of packets.

The delay variation (also called jitter) is the difference in the delay of a packet compared to the instant of time, when the packet should have arrived, if it experienced only the minimum fixed delay of the network. This is the definition of jitter that is used in this thesis.

There are also other definitions of packet delay variations in use, like interarrival jitter that is sometimes used by IETF (Internet Engineering Task Force). In the Internet

draft defining the RTP protocol, [Schulzinne 99], (see Section 4.3.5), there is a definition of how this jitter shall be calculated, which uses the delay variation that two consecutive packets experience. The absolute value of this difference is filtered to some sort of mean value, which is the calculated jitter value. This value is calculated on the run. It should be noted that this jitter value does not capture slow delay variations because time instants of only two consecutive packets are used in the algorithm.

A hypothetical probability distribution of packet delay is shown in Figure 4.1 (note that the probability distribution curve does not correspond to any real jitter distribution, but rather serves to illustrate the concept). In this thesis the peak-to-peak value of the delay variation is used as the jitter amplitude, see Figure 4.1.

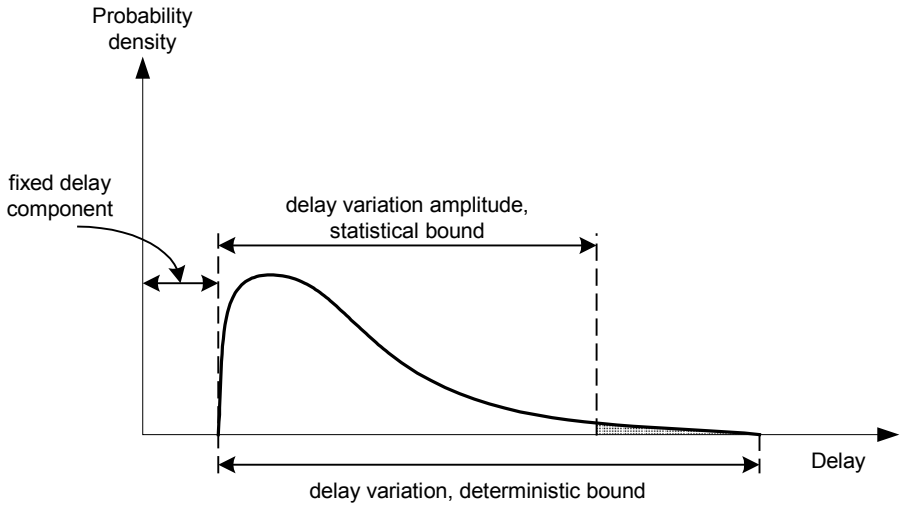

Figure 4.1 Distribution of hypothetical packet delay.

When audiovisual information is streamed over a network (see Section 5.2 for a definition of streaming), the jitter amplitude can occasionally be larger than the maximum delay variation the application is capable of absorbing. Packets that are delayed more than this maximum delay will then be thrown away by the application/terminal since they arrive too late to be useful. This maximum delay is denoted the statistical bound in Figure 4.1. The shadowed area under the curve in Figure 4.1 is the probability that this bound is exceeded.

As discussed later, the distribution used in the simulations is truncated, which means that the deterministic bound and the statistical bound actually coincide, see Figure 4.1.

Delay variations can be described with their spectral characteristics. Two different terms are sometimes used to denote delay variations. One may talk about high frequency and low frequency delay variations, where the first one is called jitter and the second one is called wander.

In this thesis, the terms delay variation and jitter is used interchangeably, and both will refer to delay variations irrespectively of spectral properties. However, when analysing the simulations, see Section 8, a distinction between "slow" and "fast" delay variations is made, since they affect the video signal in different ways.

4.2.3 IP-based Networks

The most widespread protocol for computer network communication is the Internet Protocol, IP for short. Networks using the Internet Protocol are usually called IP-based networks for short. This protocol is a member of the TCP/IP suite, which is used in all communication over the Internet.

Internet is a collection of networks and computers to form a global virtual network. The networks connected to Internet use different network techniques like packet and circuit switching. But all information sent over Internet is encapsulated in packets, like in packet switched networks.

In IP-based networks data are sent with "best effort". This means that the networks give no guarantee that the information will arrive at the receivers. The packets could be lost, or arrive out of order. They will also experience some uncontrollable delay variations. Several different techniques to overcome these problems to reach some quality of service, QoS, have been proposed, see Section 5.3 for a definition of QoS.

4.3 Protocols

4.3.1 TCP/IP Layering

Network protocols are usually developed in layers, where each layer is responsible for different distinct functions. In the TCP/IP suite case there are four different protocol layers as shown in Figure 4.2, see [Stevens 94].

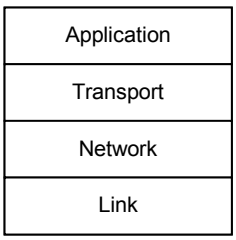

Figure 4.2 The four layers of the TCP/IP suite

1. The link layer, also called the data-link layer, normally includes device drivers and network interface in the computer. This layer is concerned with the access to as well as the routing data across a network for two peers attached to the same network. The purpose of this layer is that higher layer protocol need not be concerned about the specifics of the network to be used. Sometimes this layer is divided into two layers, The physical layer and the network access layer see [Stallings 97]. Ethernet is an example of a link layer protocol.

2. The network layer is responsible for transferring data between peers on different networks. IP, ICMP and IGMP are the network protocols in the TCP/IP protocol suite.

3. The transport layer provides a flow of data between two peers, for the application layer above. TCP and UDP are the transport protocols in the TCP/IP protocol suite.

4. The application layer handles all the details of the particular application.

Some of the protocols mentioned above will be treated in the following sections. The rest of them are described in [Stallings 97].

4.3.2 Ethernet

Ethernet is the predominant LAN technology used with TCP/IP today. It uses a medium access control technique called CSMA/CD.

The maximum transfer unit, MTU of Ethernet packets is 1500 bytes. The currently most used one is the 10 Mbit/s version but faster versions are available like Fast Ethernet that operates at 100 Mbit/s.

4.3.3 IP, Internet Protocol

As mentioned earlier IP is the network layer protocol used for all data traffic over the Internet. The current version used is IPv4 but a newer version IPv6 is to replace it, see [Stallings 97].

4.3.4 UDP, User Datagram Protocol

UDP is a simple, datagram-oriented transport layer protocol. Each output operation by a process produces exactly one UDP datagram, which causes one IP datagram to be sent. This is different compared to a stream oriented protocol such as TCP where the amount of data written by an application may have little relationship to what actually gets sent in a single IP datagram. It is up to the application to split the output data stream into convenient packet sizes.

UDP provides no reliability. It sends the datagrams that the application writes to the IP layer, but there is no guarantee that they will reach the destination. It is up to the application to handle problems of reliability, such as lost packets, duplicate packets, out-of-order delivery and loss of connectivity.

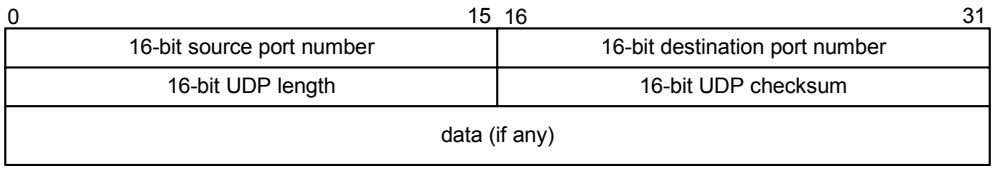

Figure 4.3 UDP header.

The port numbers, see Figure 4.3, are used to demultiplex the incoming packets to the correct application.

4.3.5 RTP, Real Time Protocol

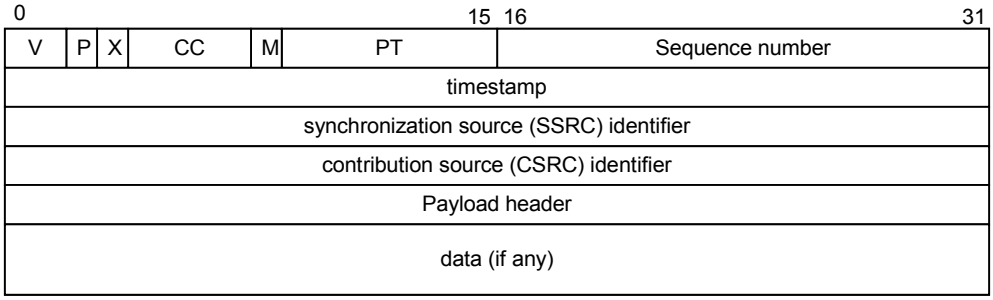

Figure 4.4 RTP header.

RTP is the Internet standard protocol for the transport of real time data, see [Schulzinne 99]. It is mainly intended to be used on top of UDP/IP, but can also be used with other protocols, for example AAL5/ATM. An RTP packet encapsulated in a UDP/IP is shown in Figure 4.5.

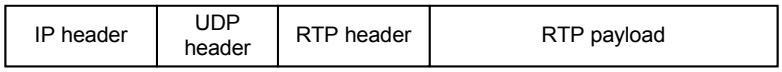

Figure 4.5 Encapsulation of RTP in a UDP/IP packet

RTP provides functionality that is suitable for applications transmitting real-time data, such as audio/video over multicast or unicast networks. These functions include: content identification of payload data, sequence numbering, timestamping, and monitoring QoS of data transmission. In the UDP/IP case, UDP provides the checksum and the multiplexing.

The sequence number, see Figure 4.4, is incremented by one for each RTP packet. It can be used to detect packet losses and out-of-order delivered packets.

The timestamp is a 32-bit number and typically reflects the sampling instant of the first byte of data in the RTP packet (as described later in Section 4.4 the timestamps of RTP may actually be used in two different ways). It can be used to synchronise the receiver to the sampling clock of the sender to determine the playout time and to measure packet interarrival jitter, (as described in Section 4.2.2). The frequency of the clock generating the timestamp is dependent on the data format carried in the payload. In the MPEG-2 case the frequency will be 90 kHz, see Section 4.4.

RTP actually consists of two protocols, RTP and RTCP (Real Time Control Protocol). RTP is used for the transmission of data packets. RTCP provides support for the real-time conferencing of groups. This support includes source identification and support for gateways like audio and video bridges as well as multicast-to-unicast translators. It offers QoS feedback from receivers to the multicast group as well as support for the synchronisation of different media streams.

There are several RTCP packet types to carry a variety of control information. It is not within the scope of this thesis to describe all of them but two of them can be interesting to mention, SR (Sender Report) and RR (Receiver Report). SR is used for transmitting information from active senders to participants that are not active senders. One interesting information provided in SR packets, in the matter of synchronisation, is a mapping between NTP timestamps and RTP timestamps. Another information

provided in both SR and RR packets is an estimate of the statistical variance of the RTP data packets interarrival time.

In their normal use the timestamps of RTP are actually not suited to measure jitter. For a timestamp to be used to get a correct measurement of the jitter, it should indicate the transmission moment. As mentioned earlier the timestamps usually reflect the sampling instant of the first byte of payload. One problem with these types of timestamps appears when video coding is used. When the encoding is done the number of bits per frame will vary, depending on the information contents of the frames. Another problem is that the timestamps will not always be monotonically increasing. For example when a motion compensated temporal prediction is used, like in MPEG-2, the frames will not necessarily be sent in time order.

4.4 MPEG-2 Video over RTP/IP

RFC 2250 specifies how to packetise MPEG-1 and MPEG-2 video and audio streams into RTP packets, see [RFC2250]. Two approaches are described. The first one specifies how to packetise MPEG-2 Program streams (PS), Transport streams (TS) and MPEG-1 system streams. The second gives a specification on how to encapsulate MPEG-1/MPEG-2 Elementary streams (ES) directly into RTP packets. The former method then relies on the MPEG systems layer for multiplexing, whereas the latter method makes use of multiplexing at the UDP and IP layers.

4.4.1 RTP Encapsulation of MPEG-2 Transport Stream

Each TS packet is directly mapped into the RTP payload, see Figure 4.6. To maximise the utilisation multiple TS packets are aggregated into a single RTP packet. The RTP payload will contain an integral number of TS packets. In the Ethernet case, where the MTU is 1500 bytes, there will be seven TS packets in each RTP payload (RTP payload size=1316), and every IP packet will have a size of 1384 bytes.

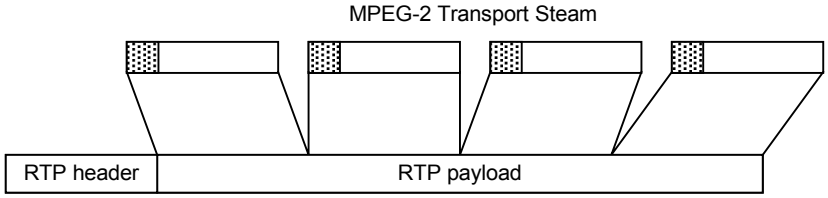

Figure 4.6 Mapping of TS packets into RTP payload.

Each RTP packet header will contain a 90 kHz timestamp. This timestamp is synchronised with the STC of the sender. The timestamp represents the target transmission time of the first byte of the payload. This time stamp will not be passed to the decoder and is mainly used to estimate and reduce jitter and to synchronise relative time drift between the transmitter and the receiver.

In the MPEG-2 Program stream case there is no packetisation restrictions. The PS is treated as a packetised stream of bytes.

In Figure 4.7, the protocol architecture for TS over IP networks is illustrated. For each protocol, it is also shown, which TCP/IP protocol layer it belongs to. In the TCP/IP suite the MPEG-2 Systems layer is considered to belong to the Application layer.

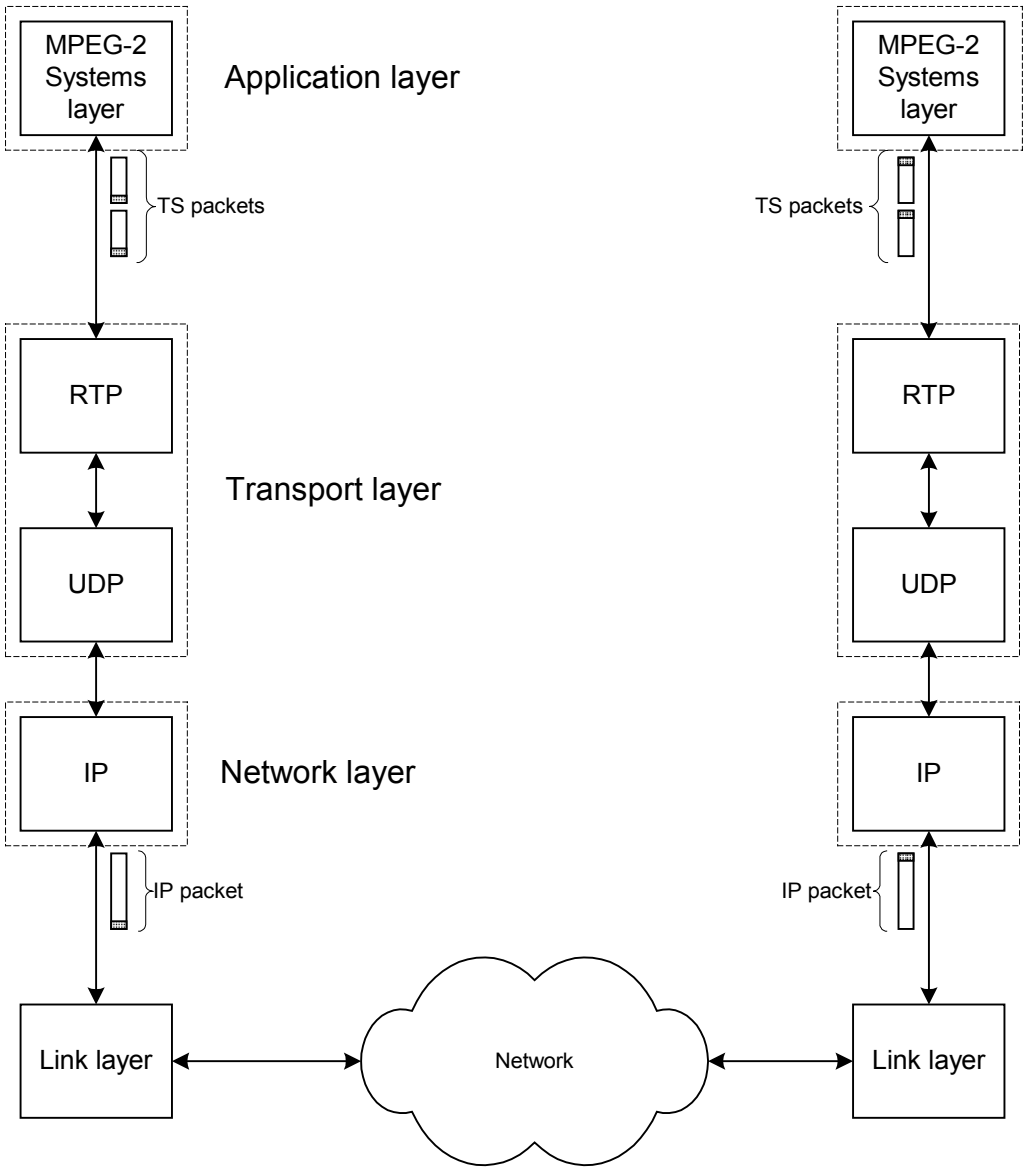

Figure 4.7 Protocol architecture for MPEG-2 TS over IP networks

4.4.2 RTP Encapsulation of MPEG-2 Elementary Stream

The second approach described in [RFC2250] is to packetise MPEG-1/MPEG-2 elementary streams (ES) directly into RTP packets. Audio ES and Video ES are sent in different streams and different payload type is assigned to them. Both audio and video streams have their own payload header that provides the information that the MPEG-2 System layer normally provides. It is not in the scope of this thesis to describe them.

One big difference in synchronisation and dejittering issues compared to the encapsulation of TS and PS, is the timestamp used in the RTP header. In this case the timestamp in the RTP header represents the presentation timestamps (PTS) in MPEG-2 Systems layer, see Section 3.5.2. In this case the timestamp is both used for reduction of jitter and in the decoding process.

5 Real-time Streaming Applications

5.1 Overview

First in this section some definitions are made, concerning real time streaming. Thereafter, the concept Quality of Service is described. Then some classifications are made of different audio-visual streaming services. At the end of the section the concept of synchronisation is defined and described.

5.2 Definitions

This introduction to real-time streaming is mainly based on the definitions suggested by [Kwok 95].

Information can be classified as time-based or non-time-based. Time-based information has an intrinsic time component. Audio and video are examples of information that has a time-base, because they generate a continuous sequence of data blocks that have to be displayed or played back consecutively at predetermined time instants. For example a video sequence is made up of frames generated at regular time instances and these frames have to be displayed at the same rate as they were generated. Examples of non-time-based information are still images and text.

A real-time application is one that requires information delivery for immediate consumption, in contrast to a non-real-time application where information is stored at the receiving point for later consumption. For example, a telephone conversation is considered a real-time application, while sending an electronic mail is considered a non-real-time application, see [Kwok 95]

It is important to distinguish between the delivery requirement (real-time or non-real-time) and the intrinsic time dependency (time-based or non-time-based), because they are sometimes mixed up. For example, a transmission of a video file is a non-real-time application even though the information is time-based, while browsing a web page is considered a real-time application even though the page has only non-time-based information.

A real-time streaming application is an application that delivers time-based information in real-time. For example, a transmission of a radio channel, that is played back at the same time as it is received, is considered a real-time streaming application.

5.3 Quality of Service

The notion of quality of service, for short QoS, originally emerged in communications to describe certain technical characteristics of the data delivery e.g. throughput, transit delay, error rate and connection establishment failure probability. These parameters were then mostly associated with lower protocol layers and were not meant to be observable or verified by the applications. These types of parameters are still sufficient to characterise communication networks transferring non-time dependent data.

When time dependent data, such as real time streaming of audio-visual information, are transferred over communication networks, a broader view of the concept quality of

service has to be used, where the entire distribution system must participate in providing the guaranteed performance levels, see [Vogel 95].

The following definition of QoS is provided by [Vogel 95]:

"Quality of service represents the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application".

For real-time applications the most important properties, according to [Rudkin 97], are temporal properties such as delay, jitter, bandwidth and synchronisation, and reliability properties such as error-free delivery, ordered delivery and fairness. The desired values for these QoS parameters are determined by the limits of human perception. For example, if round trip speech delays exceed 300 ms, conversation can become disjointed [ITU-T G.114].

Sometimes these parameters result in conflicting requirements. For example, selecting a low statistical bound of the delay variations in the dejittering buffer is preferred to minimise the delay. On the other hand, that might cause a too high packet loss ratio, as part of the dejittering process.

5.4 Classification of Real-time Audio-Visual Streaming Services

One can divide real-time streaming applications into different categories, depending on the service it provides and its tolerated delay.

5.4.1 Information Retrieval Services

These types of services include video-on-demand, where the viewer decides when to watch a specific TV program or movie. Usually these services are not very delay sensitive. The viewer can accept to wait maybe a second from the moment that he/she presses "play" and the video sequence is displayed. These services are usually only suited for unicast.

5.4.2 Communicative Services

These types of service include videoconferencing and videotelephony. Communicative services are sensitive to delay and response time. For videoconferencing, the end-to-end delay should not be more than 150 ms, see [Wolf 97]. Actually different authors suggest different delay limits. (The one suggested by Wolf should be regarded as a quite stringent requirement.) These services can be either of type unicast or multicast.

5.4.3 Distributive Services

These types of services include broadcasting/multicasting of e.g. video, of a digital TV service. Distributive services might be delay sensitive. An example of this is a TV program where viewers can call in live to the program and take part in e.g. a quiz show or other competitions. Movie channels are less delay sensitive. However it should be noted that excessive buffering at the receiver may introduce a too long channel change time.

5.5 Principles of Streaming

There are two different principles of streaming of audio and video over networks. The synchronisation problem is very different in these two cases.

5.5.1 Push Method

In the push case the source is controlling the rate of the stream of data. The sink has to estimate the time-base of the source and slave its play back rate to that of the incoming stream. This method is suitable for distributed services and the only method that can be used in broadcast/multicast applications, but it can also be used in unicast.

5.5.2 Pull Method

In the pull case the sink is controlling the time-base/rate of the data from the source. Some sort of flow control protocol has to be used in this case. The source is assumed to transmit at a rate higher than the "normal play back speed", and the sink will then fill its buffer up to a certain level. When this level is reached the sink issues a "stop transmit" command back to the source, which temporarily halts the transmission. The receiver buffer level will then decrease, and when a certain level is reached, the sink will issue a "continue transmission" command, and another cycle starts. This method is suited for retrieval services. The pull method can only be used in unicast.

5.6 Synchronisation

This is the definition of synchronisation given in [Class 97]:

"The task of synchronisation of multimedia data is to guarantee that all time dependent presentation units are only presented within their valid time interval. The valid interval for each presentation unit is specified within the synchronisation specification of multimedia data."

A presentation unit (PU) contains the atomic information of a media stream that can be presented e.g. an audio sample or a video frame.

One can distinguish between two different synchronisation problems, intra-stream synchronisation and inter-stream synchronisation.

5.6.1 Intra-stream Synchronisation

For single data streams, a stream consists of consecutive logical data units (LDU's). An LDU can be a single PU or blocks of these PU's transferred together from a source to one or more sinks. These LDU's have to be presented at the sink with the same temporal relationship as they were captured giving so called intra-stream synchronisation. An example of this type of synchronisation is the synchronised display of pictures of an MPEG-2 decoder, which uses PTSes to determine when each frame is to be presented and the PCRs to recover the time-base, see Section 3.5.2 and 3.5.3. If the video signal is not sufficiently synchronised one can have problems displaying the decoded video signal on a TV set, as discussed in Sections 2 and 3. An insufficiently synchronised audio stream, with having too much jitter in the output signal will have variable pitch, which can be disturbing.

This thesis mainly considers the problems of intra-stream synchronisation. See Section 6 for a description of the problem handled in this thesis.

5.6.2 Inter-stream Synchronisation

Inter-stream synchronisation is defined as the synchronisation of related media streams together, for example when video and audio have to be displayed together. This is also called "lip synchronisation". The time difference between related audio and video LDU's is known as the skew. An experiment made on 107 test persons showed that most of them could not notice skews of up to ±80 ms, see [Steinmetz 96]. In broadcast applications more stringent requirements are typically used. (40 ms, audio lag video, and 20 ms, video lag audio). In general, intra-stream synchronisation involves relationships between all kinds of media including pointers, graphics/images, animations, text, audio, and video.

6 Audio-visual Synchronisation Issues and Presentation of the Problem

6.1 Overview

This section will present the synchronisation issues of a general audio-visual communication system in some more detail. After this there is a description of some research work done in the area so far. Finally, a detailed description of the specific synchronisation problem studied in this thesis will be given.

6.2 Synchronisation of High Quality Video and Introduction to the Dejittering Problem

First, a description of a typical system including both the transmitter and the receiver in a distributive service providing MPEG-2 based digital television over an IP network will be given. UDP/RTP are used to carry the MPEG-2 transport stream. The receiver can be an ordinary personal computer or a set-top box.

In Figure 6.1, a simplified overview of the transmitting side is shown, which sends the video stream over an IP-based network. The figure describes a live system, which might be one sending from a TV studio in real time. First, the camera outputs the analogue video signal, which may be in RGB format, see Section 2.2.3. This signal is analogue to digital (A/D) converted. The camera also generates a separate synchronisation signal. (Note that other formats could be used, like PAL where the synchronisation information actually is part of the video signal itself, see Section 2.2.1. However that would not change the block diagram.) This signal has a frequency of f_{tx} and constitutes the line frequency of the analogue video signal. A synchronisation circuit synchronises to this signal, typically an ordinary PLL, shown in the figure. This PLL outputs a clock, which is used to determine the sampling instances in the A/D conversion process. In reality the line frequency f_{tx} will not be constant but varies in time because of temperature drift etc. Therefore, f_{tx} is a function of time $f_{tx}(t)$.

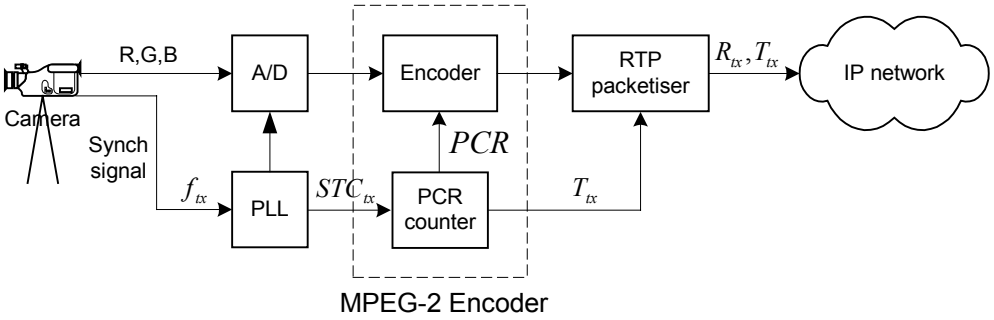

Figure 6.1 The transmitting side

After the video signal has been A/D converted, it is sent to an MPEG-2 encoder, which compresses the video signal and encapsulates the bit stream in transport stream (TS) packets, as described in Section 3.5.1. As described in Section 3.5.2 the encoder makes use of a counter the PCR counter, driven by the STC. The PCRs represent the time to which the DTSes and DTSes refer, which determines the decoding and presentation time of each frame. The frequency of the STC clock signal is denoted

$STC_{tx}(t)$, in the figure and is derived from the synchronisation signal of the analogue video signal. As mentioned in Section 3.5.2 timestamps based on PCRs are put into the MPEG-2 transport stream and later used in the decoding process.

The TS is then packetised into RTP packets, as described in Section 4.4.1, which in turn are put into UDP/IP packets. The packet stream is sent out on the IP-based network with a packet rate of $R_{tx}(n)$, where $R_{tx}(n)$ is the packet rate of the transport stream. (In this section, the variable n is only meant to indicate that functions or signals only exist in discrete events, and are deliberately carelessly used to represent different discrete time domains only to simplify this section.) As mentioned in Section 4.4.1, when TS packets are encapsulated in RTP packets, the RTP timestamps are synchronised to the STC of the MPEG-2 encoder and indicate the transmission time. The sequence of these timestamps then creates a discrete signal, denoted $T_{tx}(n)$. (Note that packet rate need not be synchronised to the STC.)

An overview of a general receiver is shown in Figure 6.2, which receives the packet stream from the IP network. The packet stream has experienced a delay variation when transferred over the network, as described in Section 4.2.2. Therefore the rate of the received packet stream of IP packets will not be the same as that of the transmitted stream. The packet rate of the received stream is denoted $R_{rx}(n)$ in the figure. The RTP timestamps of this packet stream are denoted $T_{rx}(n)$.

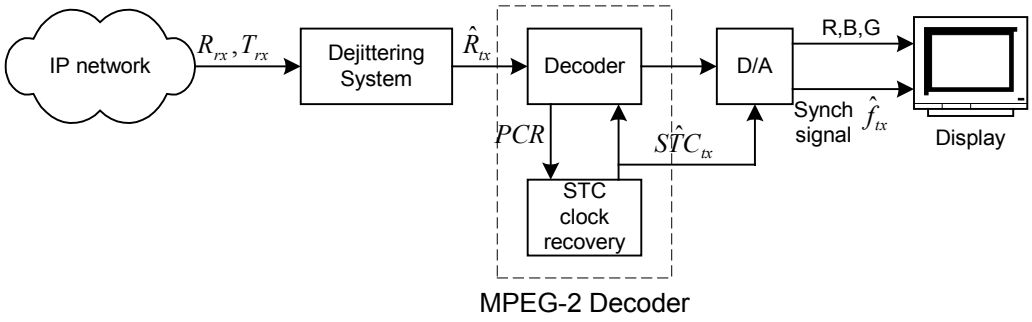

Figure 6.2 The receiving side

As mentioned in Section 3.5.3 the MPEG-2 decoder has its own STC to represent time, which it uses in the decoding process. The decoder has to make a time base recovery from the incoming PCR timestamps of the TS. It may also include a "true" recovery of the STC frequency, typically implemented by a DPLL, as described in Section 3.5.3. (Note that a pure software implementation would not include the DPLL, but only recover the PCR.) The frequency of this clock is an estimate of the STC of the transmitter and is denoted $S\hat{T}C_{tx}(t)$ in the figure. As mentioned earlier, $S\hat{T}C_{tx}(t)$ is also used in the digital to analogue (D/A) conversion to determine the sampling instances. Therefore, all variations of $S\hat{T}C_{tx}(t)$ will directly affect the frequency of the analogue video, resulting from the A/D conversion. The frequency of the synchronisation signal of this analogue video signal is proportional to $S\hat{T}C_{tx}(t)$, i.e. $\hat{f}_{tx}(t) \sim S\hat{T}C_{tx}(t)$. E.g. a 20 ppm frequency error of $S\hat{T}C_{tx}$ will directly result in a 20 ppm frequency error in $\hat{f}_{tx}$.

As discussed in Section 2.2.4 an analogue video signal has to be accurate and stable in frequency, (realistic requirements of the analogue video signal will be discussed later

in Section 8). Therefore, the STC recovery function of the decoder will put certain requirements on the input jitter. In the RTI specification of MPEG-2, see [MPEG2 RTI], there is a recommendation that a decoder should handle at least delay variations (jitter) of ±25 μs. For the MPEG-2 decoder of Figure 6.2 to recover the STC of the transmitter "correctly", i.e. $STC_{tx}(t)$, the delay variations of the incoming transport stream should then be within the region ±25 μs. If the incoming packet stream from the network should suffer larger delay variations than these, the delay variations have to be reduced in some way before the transport stream is input to the MPEG-2 decoder. This is done in the dejittering system, which actually makes an estimate, denoted $\hat{R}_{tx}(n)$, of the transmitted packet rate, $R_{tx}(n)$.

If the variations of $\hat{R}_{tx}(n)$ are slow, i.e. the dejittering system has a much longer time constant than the time constant of the clock recovery circuit of the decoder, $\hat{STC}_{tx}(t)$ will be approximately proportional to the estimated packet rate, i.e. $\hat{STC}_{tx} \sim \hat{R}_{tx}$. (This only holds if the transmitted packet rate $R_{tx}(n)$ is constant.) Therefore variations in $\hat{R}_{tx}(n)$ will directly result in variations of the analogue video signal. Ideally, $\hat{f}_{tx}(t)$ exactly follows $f_{tx}(t)$ with a constant delay, i.e. $\hat{f}_{tx}(t) = f_{tx}(t-\tau)$ where τ reflects the total delay of the system including the delay in the MPEG-2 encoder and decoder, the delay of the network and the addition delay introduced by the dejittering system.

The dejittering system described above is what is going to be designed and evaluated in this thesis.

6.3 Different "Degrees" of Decoder Synchronisation

One can distinguish between different "quality degrees" of the synchronisation of the decoder. These classes are described below. It should be noted that this distinction could be made in many different ways and ours is only one way to do it.

- Class A: Fully synchronised: In this case the decoder makes an exact clock recovery of the transmitted time base, including the sampling frequency and in this case there is a nominal constant delay from the A/D converter to the D/A converter. The audio will be played back at the same pitch as the encoding side and the decoded frames will be played back with the same frame interval as they were sampled. Typically, a clock recovery is implemented by a DPLL, for example the recovery of the 27 MHz STC in an MPEG-2 decoder. This case is suited for high quality applications like digital TV/HDTV.

- Class B: Almost synchronised: This case is like class A but no recovery of the sampling frequency is made. E.g. in the MPEG-2 TS case, the decoder will use a free running clock driven from the PCR counter, and use the PCRs received in the transport stream to update the counter a regular basis. In this case frame/sample slips can occur at certain intervals, the interval depends on the difference in frequency between the clock of the encoder and the decoder. To avoid audio sample slip, e.g. adaptive resampling may be used. This class is typically used by a PC

based streaming client, including those using a hardware MPEG-2 decoder. (Note that there are boards which belong to class A.)

- Class C: Only inter-media synchronised: In this case only an inter-stream synchronisation is made, but the end-to-end delay is not constant. Delay variations of the presented audio and video need to be accepted and frequent slips will occur. E.g. a videoconferencing application used over the Internet will typically fall into this category.

- Class D: Non-synchronised: In this case not even an inter-stream synchronisation is made. The decoder plays back the audio and video access units as soon as they are received. Few modern implementations fall into this category.

6.4 Work done so far in the Area

In this section a brief overview of the area of dejittering and synchronisation of MPEG-2 video delivered over packet networks will be given. Some of the articles mentioned below will also be used later in this thesis.

One can distinguish between two different approaches to solve the dejittering problem, one based on buffer fullness and the other one based on timestamps.

The first one, often denoted adaptive buffer, monitors the fullness of the input buffer and determines the playout rate according to some low pass algorithm, e.g. [Singh 94] or [Parekh 97]. These algorithms can only be used when the video stream is transferred with a constant bit rate (CBR). The results from [Parekh 97] will be discussed later in Section 8.4.7.

The other approach, which can be used both with constant bit rate (CBR) and variable bit rate (VBR), is to use some sort of timing information, e.g. timestamps, in the packet stream, which can be used in the dejittering process, see for example [Andreotti 95] or [Tryfonas 99]. The scheme of this thesis falls into this category.

In [Andreotti 95] ordinary DPLLs are used to dejitter MPEG-2 stream, which uses the PCR timestamps in the MPEG-2 Systems layer. In the simulations it is assumed that the MPEG-2 stream is delivered over a network with small delay variations, and therefore uses a peak-to-peak jitter (delay variation) amplitude of up to 1 ms. Amplitudes of delay variations in these regions are regarded as very low in IP-based networks. This scheme is therefore not suited for IP networks, and will not be further discussed in this thesis.

Also in [Tryfonas 96] DPLLs are used to dejitter MPEG-2 streams delivered over ATM networks, but in these cases peak-to-peak amplitudes of delay variations up to the region of 20 ms are used (which is more realistic in IP-based networks). The results from the simulations made in [Tryfonas 96] will also be discussed later in the section on simulations.

There is not much research work done regarding the synchronisation problems of MPEG over IP-based networks. But one study, which is comparable to the problem dealt with in this thesis, is described in [Noro 99a] and [Noro 99b]. In these articles linear regression on received timestamps is used, to estimate the time-base of the

transmitter. Noro et al are comparing their results with ordinary first order DPLLs and their method is concluded to be superior to ordinary PLLs. The results from the simulations made in [Noro 99b] are difficult to evaluate because there is vague information about the jitter process used to simulate the IP-based network except that it is normally distributed and with a peak-to-peak jitter amplitude. Especially the spectral components of the jitter process are very interesting. Very simple DPLLs are used, against which they match the performance of their algorithm. (Noro uses first order DPLLs, see [Andreotti 95] for a description of DPLLs of different orders.) Therefore their conclusions are somewhat questionable and will not be further discussed in this thesis.

6.5 Principal Functionality of the Scheme

The scheme designed and evaluated in this thesis is meant to have a quality degree of class A, as defined in Section 6.3. Therefore the MPEG-2 decoder must be able to make an exact clock recovery of the time-base of the encoder including the STC clock frequency of the encoder.

As mentioned in Section 6.2 the task of the dejittering system is to estimate the packet rate of the transmitter, $R_{tx}(n)$ and to send the packet stream to the MPEG-2 decoder at this estimated packet rate, $\hat{R}_{tx}(n)$.

A principal model of the function of the scheme is shown in Figure 6.3. The incoming packet stream from the network is put in an input buffer. The control system indicated in the figure then controls when packets are read from the buffer and sent to the decoder. The buffer has to be at least the size of the jitter amplitude (the statistical bound) that one wants to absorb. Packets that experience more delay than the maximum jitter amplitude will be discarded.

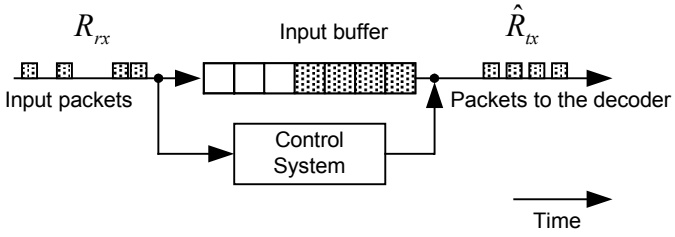

Figure 6.3 Principal model of the dejittering system

The dejittering system should work with both constant bit rate (CBR) and variable bit rate (VBR) streams. As mentioned in Section 6.3 a system that is only designed to work with CBR streams can use the buffer fullness to control the packet rate from the buffer. In the case of VBR some sort of timing information has to be included in the packet stream that can be used to estimate the packet rate of the transmitter, $R_{tx}(n)$. In the case treated here, the RTP timestamps can be used to estimate this packet rate. As mentioned earlier these timestamps are mainly used to estimate and reduce jitter. They are not meant to be used in recovering the STC in MPEG-2, see Section 3.5.2 and this is what they are used for in this thesis. In Section 7 the problem is described further, with a mathematical approach and in the same section the proposed scheme, which is tested later in simulations, is described.

6.6 Specific questions and performance requirements

In this section a specific question is listed, which this thesis tries to answer or discuss. Also some performance requirements of the scheme, which are going to be designed, are given.

- Is it a realistic goal to achieve a fully, class A, synchronisation (where the following decoder makes an exact PCR clock recovery) of high quality MPEG-2 audio-visual streaming over IP-based network, if the delay variations may reach amplitudes of 100 ms?

- The dejittering system will obviously introduce at least a delay corresponding to the jitter amplitude it is set to absorb. However, it will also add some additional delay due to the control system. (This will be further described in the following section.) Since the scheme should be usable in live transmission of television the additional delay the scheme introduces should be small compared to the jitter amplitude.

- The scheme is to be used on consumer equipment in digital television. Therefore the performance on the scheme is dictated of the requirements of a typical consumer TV equipment, regarding frequency accuracy and stability.

- In the real time interface (RTI) of MPEG-2, see [MPEG2 RTI], there is a recommendation that all MPEG-2 decoders should handle at least jitter amplitudes of ±25 μs. Therefore the scheme should be able to reduce the 100 ms of input jitter to this region.

7 Simulation model

7.1 Overview

In this section a mathematical description of the problem will be given, where a lot of definitions are made, which are used later in the thesis. After that a thorough description of the proposed scheme and a mathematical analysis of the system will be given.

7.2 Mathematical Description of the Problem

7.2.1 Time-Bases

Both the transmitter and the receiver have their own time-bases. These are typically driven by the internal hardware clocks that both sides use in the transmission and reception of the video stream. (However, an MPEG-2 real time encoder typically synchronises its time-base to that of the video source, e.g. a video camera). In reality, oscillators always have some frequency deviation from their nominal frequency, and therefore these time-bases will not be the same. (In addition, the frequency deviation is not constant, but may vary, which will be discussed later.) In Figure 7.1 the two time-bases are shown. The two functions correspond to the timestamps of the two clocks at time instants *n*.

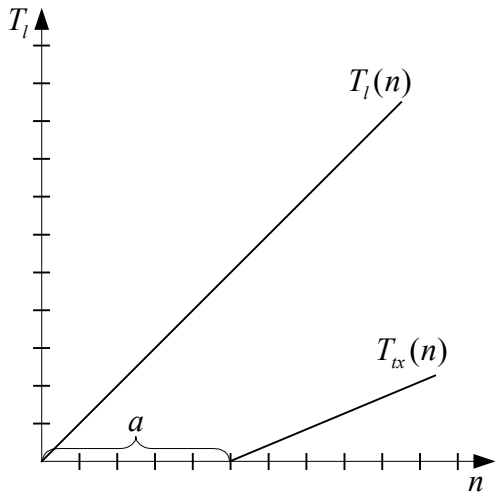

Figure 7.1 Time-bases

The time-base of the receiver is denoted $T_l(n)$ and the transmitter $T_{tx}(n)$. In reality, there will always be a phase difference between the two time-bases, this phase difference at $n = 0$ is denoted *a*. In this description the time-base of the receiver is used as a reference time-base. All other events are compared relative to this clock. For the purpose of the simulations performed here, this simplification does not mean any restrictions. It should be noted, however, that in real implementations, the $T_l(n)$ is not an "ideal" time-base, but will vary with time. All time-bases are assumed to be describable in discrete time. You can think of these discrete events as the minimum resolution of the time-base of the receiver.

$$T_l(n) = n \quad [7.1]$$

In Figure 7.1 the clock of the transmitter has a lower frequency than the receiver. (Note that in reality the difference in frequency between the two clocks is small.) This can be described with equation [7.2],

$$T_{tx}(n) = (1-\varepsilon)\cdot(n-a) \quad [7.2]$$

where ε is a small positive constant, which describes the difference in frequency between the two clocks and *a* is the phase difference between the two clocks at $n = 0$. In reality oscillators will always have some drift and the frequency deviation between the two clocks will therefore not be constant. Then ε will not be a constant, but depends on n, $\varepsilon = \varepsilon(n)$.

In the rest of the description an assumption that *a*=0 is made. This simplification does not mean any restrictions but only serves to simplify the rest of the description.

7.2.2 Jitter of the Arrival Timestamps

When the audio-visual stream is transmitted over the network, the packets will arrive at the receiver with variable delay, see Section 4.2.2. The RTP timestamps in the stream will therefore be affected by this jitter, see Figure 7.2.

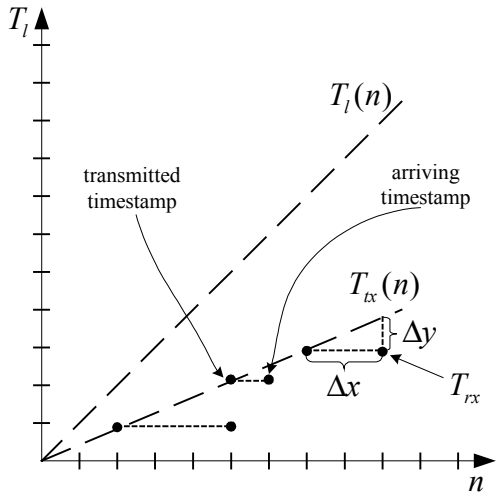

Figure 7.2 Induced jitter on the timestamps

The "signal" formed by the sequence of received timestamps is called T_{rx} and the jitter process $T_j = \Delta x$. If one imagines that T_{rx} exist in every discrete point of time, (this is obiously a simplificaion, but implies no restriction in the following analysis), one can write,

$$T_{rx}(n) = T_{tx}(n - T_j(n)) \quad [7.3]$$

or by using the notation from Figure 7.2 and using [7.2] with a=0.

$$\left.\begin{aligned} T_{rx}(n) &= T_{tx}(n) - \Delta y \\ &= T_{tx}(n) - (1-\varepsilon)\cdot \Delta x \\ &= T_{tx}(n) - (1-\varepsilon)\cdot T_j(n) \end{aligned}\right\} \quad [7.4]$$

[7.4] only holds if ε is constant. If ε is small, one can make the approximation.

$$T_{rx}(n) \approx T_{tx}(n) - T_j(n) \quad [7.5]$$

Now one can see the jitter process as an additive process.

7.2.3 Description of the Dejittering Problem.

From [7.5] one can now describe the dejittering problem with ordinary signal processing theory. To get an estimate of $T_{tx}(n)$ called $\hat{T}_{tx}(n)$, one has to filter out the jitter process $T_j(n)$ from $T_{rx}(n)$, shown in Figure 7.3. (Note that $\hat{T}_{tx}(n)$ need to be delayed compared to $T_{tx}(n)$, which will be further discussed below.)

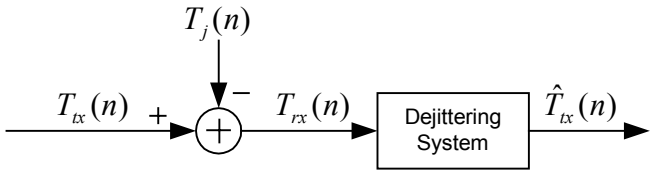

Figure 7.3 Model of the dejittering problem.

7.3 Description of the Proposed Scheme

7.3.1 Overview

In this section the proposed scheme is described. When the scheme was designed it was meant to use the principles of a DPLL, but the scheme should be implementable in software. The inner part of the proposed scheme, the dejittering system which will be further discussed below, can be regarded as a DPLL, but there is one big difference between a DPLL and this system. By nature a DPLL is a non-linear control system and is difficult to analyse mathematically, but the dejittering system is designed as an ordinary linear discrete system and can be easily analysed.

The scheme is shown in Figure 7.4. It works like this. From each RTP packet that arrives from the network the RTP timestamp, T_{rx}, is read and sent to the dejittering system and then the RTP packet is put into the input buffer. From the dejittering system an estimate of the 90 kHz clock of the sender is given, $\hat{T}_{tx}$. This timestamp minus a constant dejittering offset is compared with the RTP timestamps of the packets in the Input buffer. The RTP payload of the packets is sent to the MPEG-2 decoder at the time indicated by this timestamp. The dejittering offset corresponds to the maximum jitter amplitude we want to handle in the system, see Section 7.3.5. In this model, the dejittering offset is assumed to be constant. However, a more advanced scheme could let this vary slightly, depending on the estimated "network conditions". Such schemes are often referred to as using an adaptive playout point, see for example [Ramjee 94].

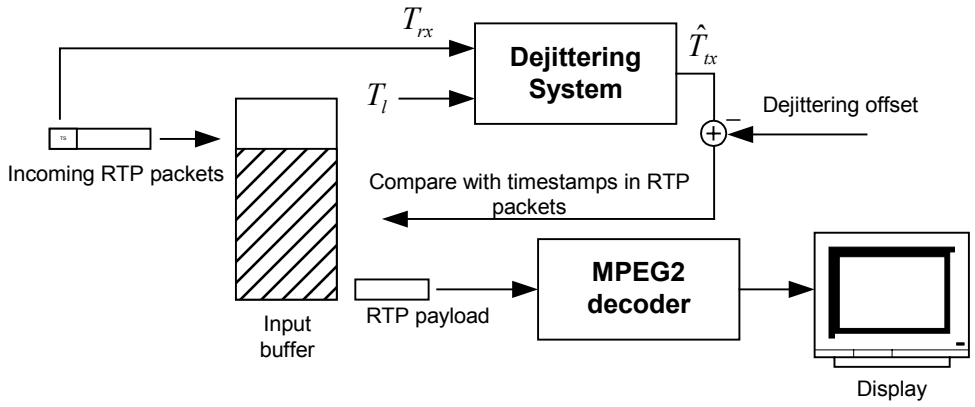

Figure 7.4 Model of the whole system

7.3.2 The Dejittering System

A model of the dejittering system is shown in Figure 7.5. It is run with a constant clock frequency of f_s. The time instants of the system are denoted n_s. The dejittering system has two input signals, the arriving timestamps T_{rx} and the clock of the receiver T_l. The system works likes this. Every new timestamp T_{rx} is loaded to the interpolation system. At every f_s clock tick a new timestamp is given by the interpolation system, (explained in Section 7.3.3). The new timestamp, $T'_{rx}(n_s)$ is compared with the current value of the estimate of the clock of the transmitter $\hat{T}_{tx}(n_s)$. The difference gives an error term $e(n_s)$. This term is the input to a low pass filter, H(z). The filter is used to average the fluctuations of the error term, and corresponds to a loop filter of a DPLL. The output from the LPF is sent to a discrete integrator G(z). The transfer function of G(z), in Z-domain, is given in [7.6].

$$G(z) = \frac{1}{1 - z^{-1}} \quad [7.6]$$

The output from G(z) is then added to the current value of $T'_l(n_s)$. This sum gives the estimate of the clock of the transmitter $\hat{T}_{tx}(n_s)$. $T'_l(n_s)$ is the clock of the receiver, T_l minus a constant, Initial phase offset, explained in Section 7.3.4. It can be shown that the combination of G(z) and the clock $T'_l(n_s)$ corresponds to the VCO and the PCR-counter in the DPLL.

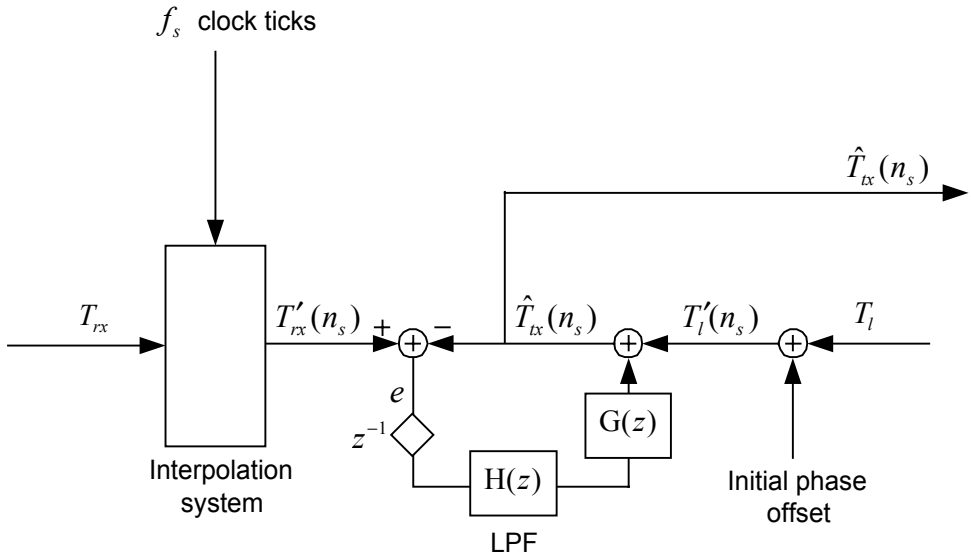

Figure 7.5 Model of the dejittering system

7.3.3 Interpolation of the Input Timestamps

Packet arriving at the receiver will normally not arrive exactly at the clock ticks of the dejittering system n_s, see Figure 7.6.

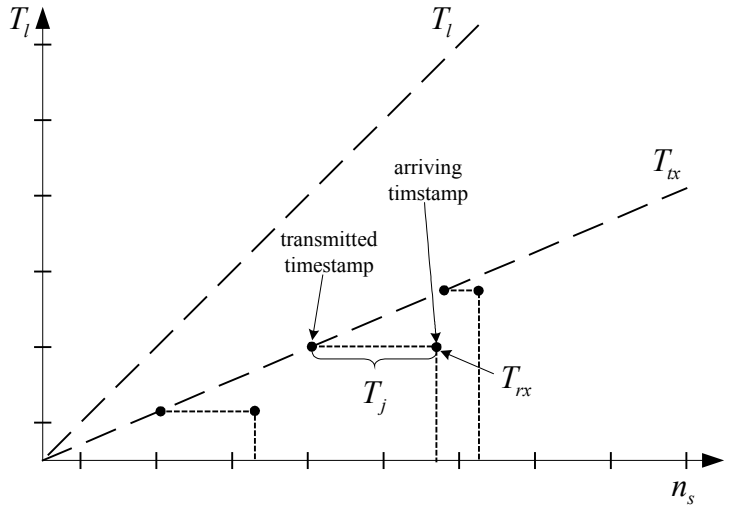

Figure 7.6 Induced jitter on the timestamps

Now the question is what timestamp is to be sent to the dejittering system at the clock ticks, n_s? Some kind of interpolation has to be done of the timestamps. The easiest way to do that is simply to measure the time interval from the arrival of the most recent timestamp to the current clock tick of the dejittering system. This time interval is now added to the most recent timestamp and the new timestamp T'_{rx} is then sent to the system. The clock of the receiver T_l is used when the measurement of the interval is made, see [7.7].

$$T'_{rx}(n_2) = T_l(n_2) - T_l(n_1) + T_{rx}(n_1) \quad [7.7]$$

$n_1 \in \{n\}$ is the arrival time of the last timestamp and $n_2 \in \{n_s\}$ is the current clock tick of the dejittering system. The interpolation is illustrated in Figure 7.7.

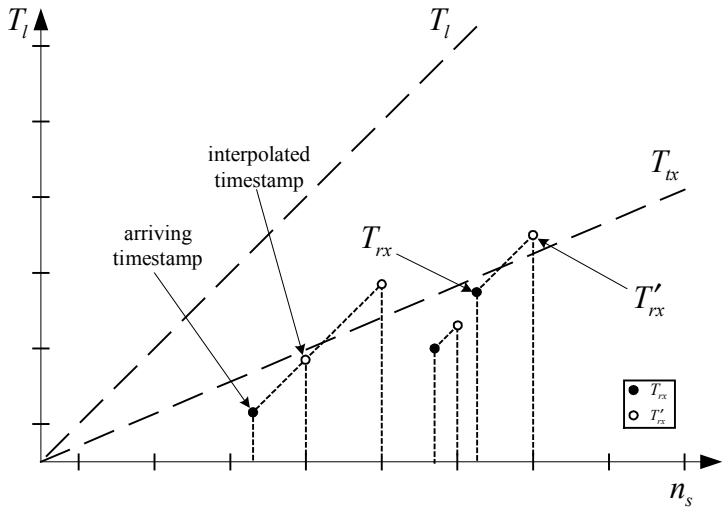

Figure 7.7 Interpolation of incoming timestamps

7.3.4 The Initial Phase

Ideally the dejittering system will lock to the clock of the transmitter plus the mean value of the jitter process T_j, described by [7.8].

$$\hat{T}_{tx}(n) = T_{tx}(n) + E\{T_j\} \quad [7.8]$$

$n = 0$ is defined as the arrival time of the first packet. To get a good initial value of $\hat{T}_{tx}$ the first time the dejittering system is run, T_l' has to be set to an estimate of the current value of the clock of the transmitter plus $E\{T_j\}$. One easy way to do this is to calculate the difference between $T_{rx}(0)$ and $T_l(0)$ and add this offset to T_l.

$$T_l'(n) = T_l(n) - (T_l(0) - T_{rx}(0)) \quad [7.9]$$

A problem with this method is that if the first arriving packet at the receiver has been delayed by, let us say, the maximum jitter amplitude, the initial phase of $\hat{T}_{tx}$ will then get an error at the same size as the jitter amplitude. It will take a long time for the dejittering system to compensate for this. (This will be show in simulations later in Section 8.) Another method is to collect some timestamps at the beginning and make an estimate of the initial phase after that. If T_{tx} and T_l have almost the same frequencies, you can estimate the initial phase by the average:

$$\text{inital phase offset} = \frac{1}{m}\sum_{n=0}^{m-1}(T_l(n) - T_{rx}(n)) \quad [7.10]$$

where *m* is the number of packets to estimate over and $n = 0$ is the time when the first packet arrives at the receiver. Now T_l' is given by [7.11].

$$T_l'(n) = T_l(n) - \frac{1}{m}\sum_{n=0}^{m-1}(T_l(n) - T_{rx}(n)) \quad [7.11]$$

7.3.5 The Input Buffer

The input buffer is used to absorb the delay variations of the incoming packet stream. The size of the input buffer has to be at least the size of the statistical bound of the delay variations, see Section 4.2.2. This bound is something that is chosen depending on the maximum tolerated delay of the system and the packet loss rate. The dejittering offset mentioned in 7.3.1 is used to control this statistical bound. The dejittering system itself will also add some delay. Therefore this extra delay has to be added to the dejittering offset. Say for example that the statistical bound is chosen to 200 ms and that the extra delay from the dejittering system is 10 ms. Then the dejittering offset has to be 210 ms or expressed in RTP 90 kHz timestamps, 18900 timestamps (90000∗0.21=18900). If then the data rate is 5 Mbit/s the total buffer size has to be ≈128 kb ($5\cdot10^6/8*0.21/1024\approx128$).

7.4 Mathematical Model of the Dejittering System

The system from the input signals T_l' and T_{rx}' to the output signal $\hat{T}_{tx}$ has been designed as an ordinary discrete linear system, seen in Figure 7.5. It can therefore be described with transfer functions in the *Z*-domain. $H(z)$ is a low pass filter (LPF) that filters out high frequency components of the error signal *e(n)*. $H(z)$ can be written as

$$H(z) = K\frac{h_b(z)}{h_a(z)} \quad [7.10]$$

where $h_a(z)$ and $h_b(z)$ are polynoms in z.

$$h_b(z) = \sum_{k=0}^{n} b_n z^{-n}$$

$$h_a(z) = 1 + \sum_{k=1}^{n} a_n z^{-n} \quad [7.11]$$

The coefficents of $h_b(z)$ are normalised like this:

$$\frac{\sum_{k=0}^{n} b_k}{1+\sum_{k=1}^{n} a_k} = 1 \quad [7.12]$$

Now K is the gain of $H(z)$ at zero frequency. $G(z)$ is a discrete integration described in [7.6].

The system can now be described with the following transfer function,

$$\hat{T}_{tx}(z) = \frac{1}{z^{-1} \cdot K \cdot h_b(z) + h_a(z) \cdot \left(1 - z^{-1}\right)} \cdot \left[z^{-1} \cdot K \cdot h_b(z) \quad h_a(z) \cdot \left(1 - z^{-1}\right) \right] \cdot \begin{bmatrix} T'_{rx}(z) \\ T'_l(z) \end{bmatrix} \quad [7.13]$$

which has two input signals $T'_{rx}(z)$ and $T'_l(z)$ and one output signal $\hat{T}_{tx}(z)$. (See appendix A.1 for a thorough derivation of [7.13]).

In a linear system with one input signal and one output signal the stability is determined by the position of the poles of the transfer function of the system. If the system has more than one input signal or more than one output signal the pole polynom of the system is determined by the least common divider, LCD of all under determinants of the transfer function of the system, see [Glad 97]. The pole polynom of the system is now

$$K \cdot h_b(z) + h_a(z) \cdot \left(1 - z^{-1}\right) \quad [7.14]$$

The system will be stable if all roots of the pole polynom are inside the unit circle $|z|<1$.

7.4.1 Different Low Pass Filter in the Loop

In the simulations introduced in Section 8 different type low pass filters are tested, but one can distinguish between two different types of them. The first one is an ordinary low pass filter with poles near the 1 in the *Z*-domain (or near the origin in the *S*-domain) and zeros at -1 in the *Z*-domain. A transfer function of a second order filter of this type will look like this

$$H(z) = K \frac{(1 + z^{-1})^2}{1 + a_1 z^{-1} + a_2 z^{-2}} \quad [7.15]$$

A problem with this type of filter is that the system will always have a constant steady state error of the phase. The steady state error can be described with the following equation,

$$E\{e\} = \frac{1}{K \cdot f_s} \cdot \left(\frac{f_{tx} - f_l}{f_l} \right) \quad [7.16]$$

where f_{tx} is the frequency of T_{tx}, f_l is the frequency of T_l. f_s is the clock frequency of the dejittering system and K is the gain of the low pass filter at zero frequency. (See Appendix A.2 for a derivation of [7.16].) $E\{e\}$ is the mean value of the error term e when the dejittering system has locked, see Figure 7.5. One problem with this error is that you cannot predict what size of the input buffer that is required, see Figure 7.4. Another problem with these types of filters is that the dejittering system cannot follow relative drift between the clocks of the transmitter and the receiver. In this case the steady state error will not be constant.

If you want a zero steady state error and a system that can compensate for a ramp drift in the clocks of the transmitter and the receiver, K has to be infinitely large. The way to accomplish this is to place a pole at 1 in the *Z*-domain (or at the origin in the *S*-

domain). This is called a filter with integral compensation, taken from Control Theory, [Cheng 93]. A first order filter of this type will look like this

$$H(z) = K' \frac{1 + b_1 z^{-1}}{1 - z^{-1}} \quad [7.17]$$

The zero at $-b_1$ is required to get the dejittering system stable, can be shown from [7.13], and should be placed near the point 1 in the *Z*-domain to get good high frequency rejection. K' is a constant gain factor. A better rejection of high frequency components of the jitter can be obtained by using a second order filter like this

$$H(z) = K' \frac{(1 + z^{-1})(1 + b_1 z^{-1})}{(1 - z^{-1})(1 + a_1 z^{-1})} \quad [7.18]$$

You can see this filter as a combination of two first order filters, one like the first order filter with integral action in [7.17] and a second one like an ordinary first order filter with one pole at $-a_1$ and one zero at the point -1. A problem with filters with integral action is that it is more difficult to get the dejittering system stable because of the instable pole at the point 1 of the filter, see [Cheng 93].

An even better high frequency rejection can be obtained with a third order filter like this

$$H(z) = K' \frac{(1 + z^{-1})^2 (1 + b_1 z^{-1})}{(1 - z^{-1})(1 + a_1 z^{-1} + a_2 z^{-2})} \quad [7.19]$$

However, it is really difficult to design this type of filter in such a way that the system is stable with good control of the transient behaviour and a bandwidth that is low enough with this and higher orders of the filter. For the simulations presented in this thesis, filters of type [7.15] and [7.18] have been used.

8 Simulations

8.1 Overview

In this section the simulations of the dejittering system will be presented. First the assumptions made in the simulations will be discussed. Thereafter a short description of the simulation platform will be given. Then the results of the simulations will be presented. At the end of the section some conclusions will be drawn from these results.

8.2 Assumptions and Conditions

8.2.1 The Packet Stream from the Source

In these simulations, the packets have been assumed to arrive over an ordinary 10 Mbit/s Ethernet interface. The source is assumed to send 250 packets per second in regular intervals with a size of 1500 bytes (which is the MTU of Ethernet). This will give a data rate of about 3 Mbit/s, which is regarded as the lowest rate used for a high quality TV service. (This will give a constant bit rate (CBR), but the system will work just as well with a variable bit rate (VBR) stream.) With the packet size used, the minimum inter-arrival time of the packets in the stream is about 1.2 ms, assuming a 10 Mbit/s interface. Therefore the sampling rate of the dejittering system f_s is chosen to 900 Hz, so that all arriving RTP timestamps are used in the dejittering process. Another reason to choose this rate is that the 90kHz RTP timestamps clock does exactly 100 clock ticks per sampling interval, which facilitates the simulation implementation and the analysis.

8.2.2 Model of the Channel

It is not the purpose of this thesis to characterise and model delay variations of real IP networks, and create a realistic channel model. Instead, a very simple channel model, that can illustrate a "worst case" scenario, has been used in the simulations.

The jitter process T_j is modelled with a uniform distribution from 0 to 100 ms. Furthermore, an assumption is made that no disordering of packets will take place in the network. To accomplish this, the uniform distribution is low-pass filtered. The filter used is a Butterworth filter of third order with a cut-of frequency of 115 Hz (which has been tested to fulfil this requirement). In Figure 8.1, a histogram of the delay variations of the channel is shown, calculated over $2 \cdot 10^6$ packets. Later on it will be shown that this low pass filtering will not affect the result of the dejittering system, because the jitter above 115 Hz will be effectively filtered out.

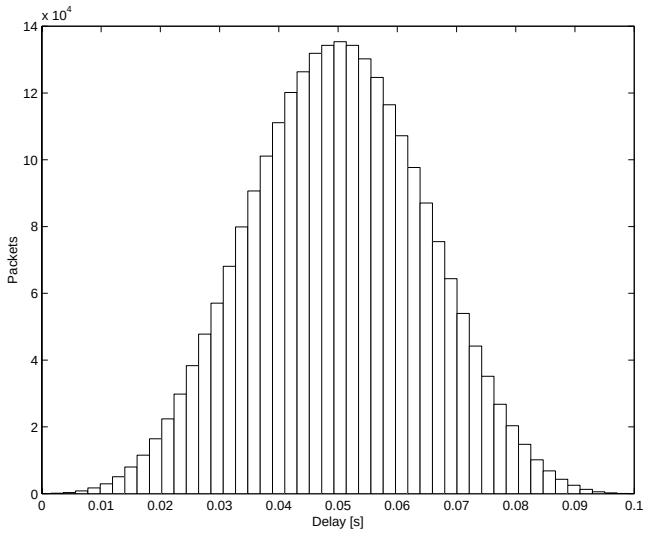

Figure 8.1 Histogram of the delay variations.

It should be noted that in these simulations the deterministic bound and the statistical bound mentioned in Section 4.2.2 actually coincide, (=100 ms).

As shown in Figure 8.1 the jitter can be characterised by a normal distribution. This can also be shown by the Central Limit Theorem, see [Blom 89], which says that the sum of a large number of random variables of arbitrary distribution approaches a normal distribution.

Another simplification, which is made, is that there are no duplicate packets and no loss of packets in the network. This is obviously not in line with true IP networks behaviour but implies no limitations for the study of the dejittering system. If packet losses were present in the network, it would of course affect the interpolation process (described in Section 7.3.3), but where it is assumed to be neglectable.

8.2.3 Accuracy of the Oscillators

In the simulations an assumption is made that the difference in frequency of the transmitter and the receiver system clocks is 100 ppm. This value has been chosen to reflect the performance of oscillators typical to a modern personal computer or workstation. However, 100 ppm is no general upper bound for real systems, a certain implementation may of course exceed this value. The effect of such a "low" performance oscillator is briefly discussed in Section 9.

A ramp shaped clock drift with the amplitude of about 50 ppm is also simulated. This is to simulate the effect of e.g. temperature drift. (In real systems the variations is expected to be lower than this.)

8.3 Simulation Platform

8.3.1 Simulation Tools

In these simulations a simulation program called Ptolemy is used. This program is freeware and is developed mainly at Berkeley University of California. When the development started in January 1990 it was aimed at digital signal processing (DSP)

simulations, but in the current version 0.7.1, a lot of new methods of simulations have been implemented in it, extending its use to other areas.

Now Ptolemy can be used for a broad range of applications including signal processing, telecommunications, parallel processing, wireless communications, network design and hardware/software co-design, see [Ptolemy]. The strength of Ptolemy is that it can combine different computational models very flexibly.

In the simulations of this work two of these models are used: synchronous dataflow (SDF) and discrete event (DE). SDF is mainly aimed at signal processing and in the simulations described below it is used for simulating the dejittering system. DE domain is used for time oriented simulations. Every event has an associated time stamp and is processed in chronological order. This domain can be used for queuing networks, communication networks and high-level models of computer architectures. In these simulations this domain is used to simulate the interpolation of the input RTP time stamps and to drive the dejittering system in the SDF domain with the simulated clock at the receiver, T_l.

Only the receiving peer is used with this program. The generation of the packet streams and the simulations of the communications channel representing the networks are made in Matlab 5.3 and all analyses of the results from Ptolemy simulations are also made in Matlab. All of this can be made in Ptolemy but the tools analysing the data are more flexible in Matlab. An overview of the simulation platform is shown in Figure 8.2.

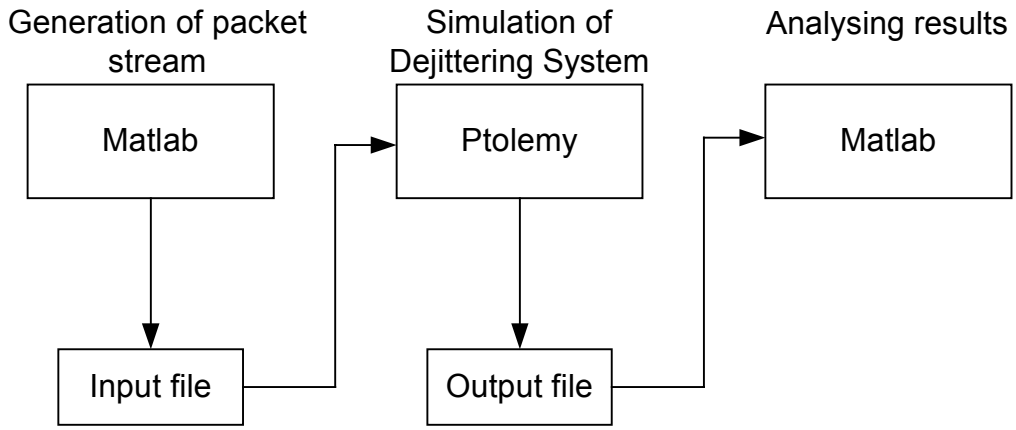

Figure 8.2 Overview of Simulation platform

First of all, the packet stream is generated by a Matlab program, which also implements the introduction of packet delay variation. This data is written to a file, the input file in Figure 8.2. Thereafter, this input file is read by Ptolemy, where the actual simulation of the whole system is made. The results from the simulations in Ptolemy are written to a file, the output file in Figure 8.2. Then the output file is read by Matlab, where all analysing of the results is made and the figures, shown in the following sections, are generated.

8.4 Simulations

8.4.1 Introduction

To study the behaviour of the dejittering system, different types of input signals have been used. They serve different purposes as described below.

First of all, in these simulations the difference in *settling time* of a frequency step, with different initial *phase error* will be shown. This is especially meant to illustrate the importance of an estimation of the initial phase as described in Section 7.3.4. (In general, the response to a step in frequency is an important characterisation of the transient behaviour of any synchronisation system designed to lock to an external timer base.)

Both filters with and without integral compensation have been used in the simulations of the dejittering system. First, results from the simulations of filters without integral compensation will be shown. In this case an ordinary Butterworth filter with second order will be used, which is in line with [Trynfonas 96]. Then, simulations of filters with integral compensation will be introduced, which is also used in [Andreotti 95].

8.4.2 Definitions of Parameters

In the analysis of the results from the simulations, some parameters will be used, which are defined below:

- *Phase error*: This parameter is defined as the error in the estimated clock $\hat{T}_{tx}(n)$ compared to an ideally reconstructed clock. (This means a clock that exactly follows the clock of the transmitter with a fixed delay corresponding to the constant mean of the jitter process.) This parameter can be directly compared to the fullness of the input buffer. A minimised phase error is preferred to minimise the required buffer size and the delay of the system.

- *Frequency error*: This parameter is an approximate of the frequency error of the estimated clock $\hat{T}_{tx}(n)$ compared to the clock of the source T_{tx}. The frequency is defined as the derivative of the phase, which is approximated with the expression in [8.1]. T in [8.1] is the sampling time of the dejittering system.

$$\frac{\hat{T}_{tx}(n) - \hat{T}_{tx}(n-1)}{T} \quad [8.1]$$

- *Frequency change rate*: This parameter is defined as the derivative of the *frequency error*. The parameter is meant to be compared to the drift requirement of the STC in MPEG-2 or the requirements of the colour sub-carrier in PAL. (The maximum tolerated drift of the STC in MPEG-2 is 75 mHz/s or 0.0028 ppm/s, see [MPEG2 Sys] In studio PAL the maximum tolerated drift of the colour sub-carrier is 0.1 Hz/s or 0.023 ppm/s), see [ITU-R 624]. To filter out high frequencies of this estimate of the drift, the mean value of this derivative is computed over a time window of 40 seconds, which is also used in [Tryfonas 96].

- *Jitter*: In these simulations the *jitter* is defined as the residual variations of the phase in frequencies above 0.25 Hz. (There is no clear line between *wander* and *jitter* and in this thesis this line is chosen to be 0.25 Hz. For example, in a Tektronix VM700A, an instrument used to measure on analogue video signals, jitter is defined as frequencies down to 0.25 Hz. In [Parekh 98], jitter is defined as spectral contents of delay variations above 10 Hz.) To measure this parameter the estimated clock $\hat{T}_{tx}(n)$ is filtered with an ordinary second order Butterworth filter with a cut-off frequency of 0.25 Hz. This parameter is meant to be measured after eventual transient phases of the system.

- *Overshot*: Is used in the frequency step responses and is defined as the maximum frequency or phase error minus the steady-state value.

- *Rise time*: Is defined as the time for the frequency to rise from 0 to 90% of its steady-state value. The rise time is denoted t_r.

- *Settling time*: It is the time for the frequency step response to reach and remain within ± 10 ppm of its steady-state value. The *settling time* is denoted t_s.

8.4.3 Effects of Integral Compensation on Transient Behaviour and Drift

Below, the difference between a dejittering system with and without integral compensation in the low pass filter $H(z)$ will be described. Two reference filters will be used in this introduction, one with and one without integral compensation. The one without integral compensation is an ordinary low pass Butterworth of second order with a transfer function according to [8.2].

$$H(z) = K\frac{(1+z^{-1})^2}{1+a_1z^{-1}+a_2z^{-2}} \quad [8.2]$$

A cut-off frequency of 4.5 mHz has been selected in this case. The gain factor K is $5\cdot10^{-6}$. (This filter is denoted no. 2 in Appendix B.1).

The second filter used is a second order low pass filter with integral compensation. It has been designed in *S*-domain and then bilinearly transformed to the *Z*-domain, see Appendix B.2. Its transfer function in the *S*-domain is

$$H(s) = K'\frac{\frac{500s}{3}-1}{s(\frac{56s}{3}-1)} \quad [8.3]$$

and bilinearly transformed to the *Z*-domain will give the following transfer function:

$$H(z) = K'\frac{(1+z^{-1})(1+b_1z^{-1})}{(1-z^{-1})(1+a_1z^{-1})} \quad [8.4]$$

The gain factor K' is 10^{-7}. (This filter is denoted no. 4 in Appendix B.2).

First the results from simulations without any jitter and without any drift in clocks, are shown. A relative clock difference of 100 ppm is used, where the source has the faster clock. The purpose of these simulations is to show the difference in steady-state error in phase between these two types of filters. Using equation [7.16] this should give a steady-state error of ≈ 22 ms with the first filter and the second filter with integral compensation will have a zero steady-state error (because of its infinite gain at zero frequency as described in Section 7.4.1). These simulations are shown in Figures 8.3 and 8.4.

As can be seen in Figure 8.3 the dejittering system has a *settling time* and *rise time* of approximately 450s. As shown in the upper part of Figure 8.3 the system has a start-up frequency error of 100 ppm, which is the difference in frequency between the source and the sink. The *phase error* is zero at the beginning as shown in the lower part of Figure 8.3, and grows to the expected steady-state error ≈ 22 ms. (The effect of non-zero initial phase error is described in 8.4.4.)

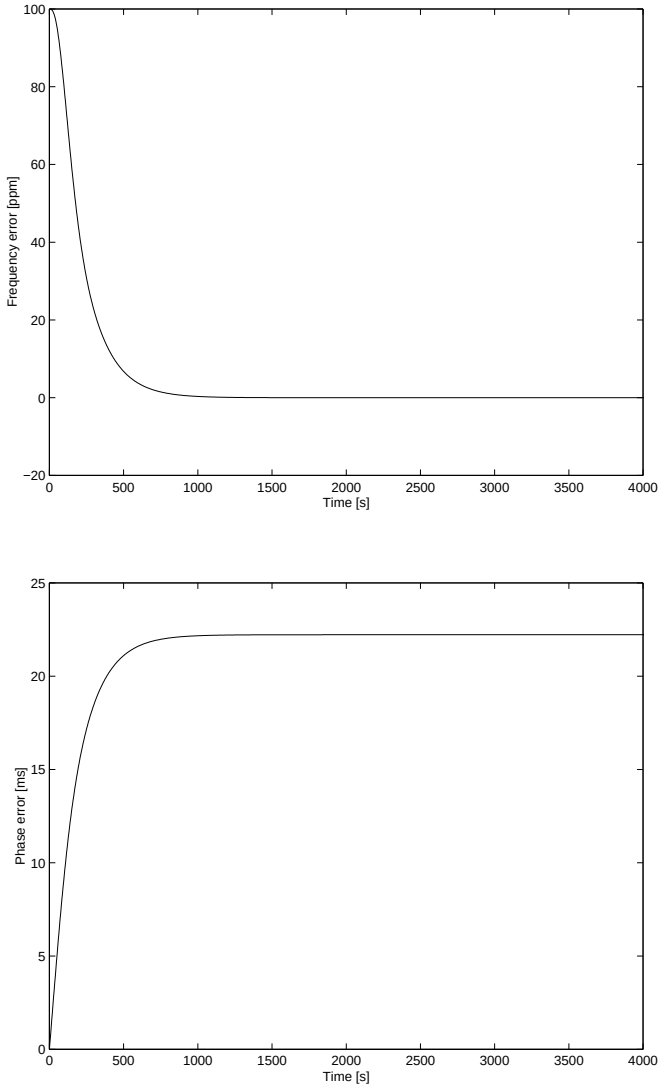

Figure 8.3 Frequency step response of the system using a second order filter without integral compensation, with no delay variations and no initial phase error

The frequency step response of the system using the filter with integral compensation is shown in Figure 8.4. As expected, the dejittering system has no steady-state error of the phase, which can be seen in the lower part of Figure 8.4. Also in this case, the start-up frequency error is 100 ppm. As in the previous case, the system starts up with a zero *phase error*, but will experience an overshot opposed to the previous case. Then it decreases back to a zero *phase error*.

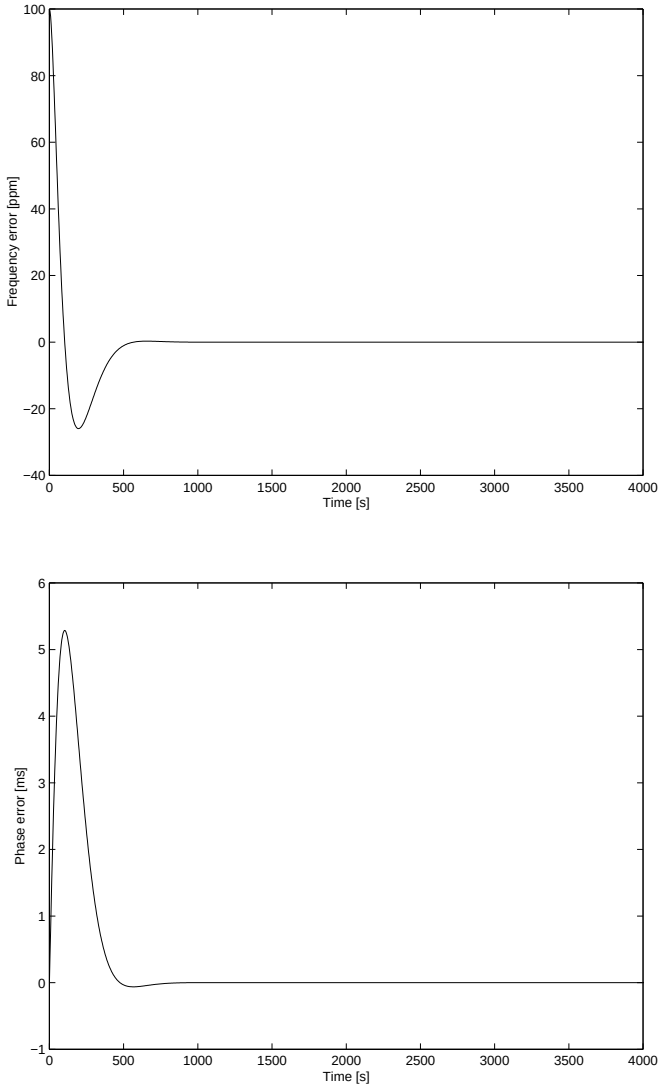

Figure 8.4 Frequency step response of the system using the second order filter with integral compensation, with no delay variations and no initial phase error

The previous plots illustrate the transient behaviour of the system, when it locks to the time base of the source. The following plots will show the "tracking" of changes in the source frequency, starting from a point in time when the system has synchronised to the initial source frequency.

A clock drift of 52 ppm of the clock of the source will be introduced. The drift starts at 2000 seconds and grows linearly in frequency for 3000 seconds. After that the drift decades for 3000 seconds. These simulations are shown in Figures 8.5 and 8.6. The time scale in the figures begins at 1000 seconds not showing the transient phase at start up.

As shown in Figure 8.5, the frequency drift of the source cannot be tracked properly by the system, and the *phase error* grows linearly with the growing frequency error. To conclude, a system without the integral compensation may introduce a significant *phase error*.

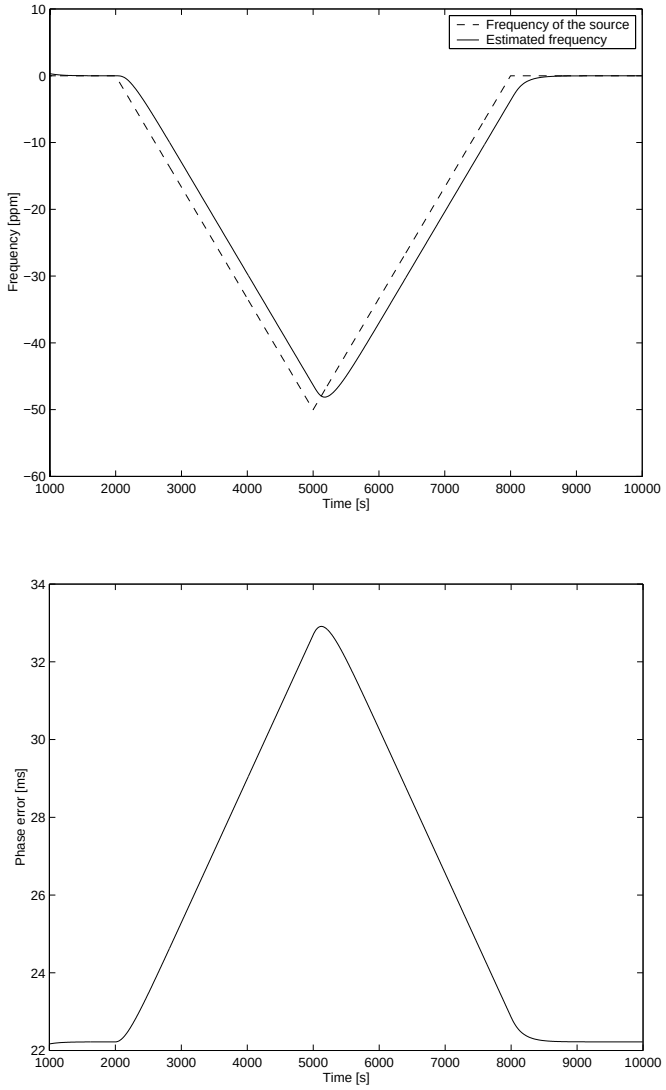

Figure 8.5 Frequency drift of 52 ppm with the system using the second order filter without integral compensation.

In Figure 8.6 the response from the 52 ppm drift of the source with the filter with integral compensation is shown. As opposed to the previous system, the system now follows the drift with constant delay seen in the lower part of Figure 8.6. Note that the phase error introduced is about two orders in magnitude lower than the previous case. The tracking behaviour of this version is obviously preferable compared to the previous version.

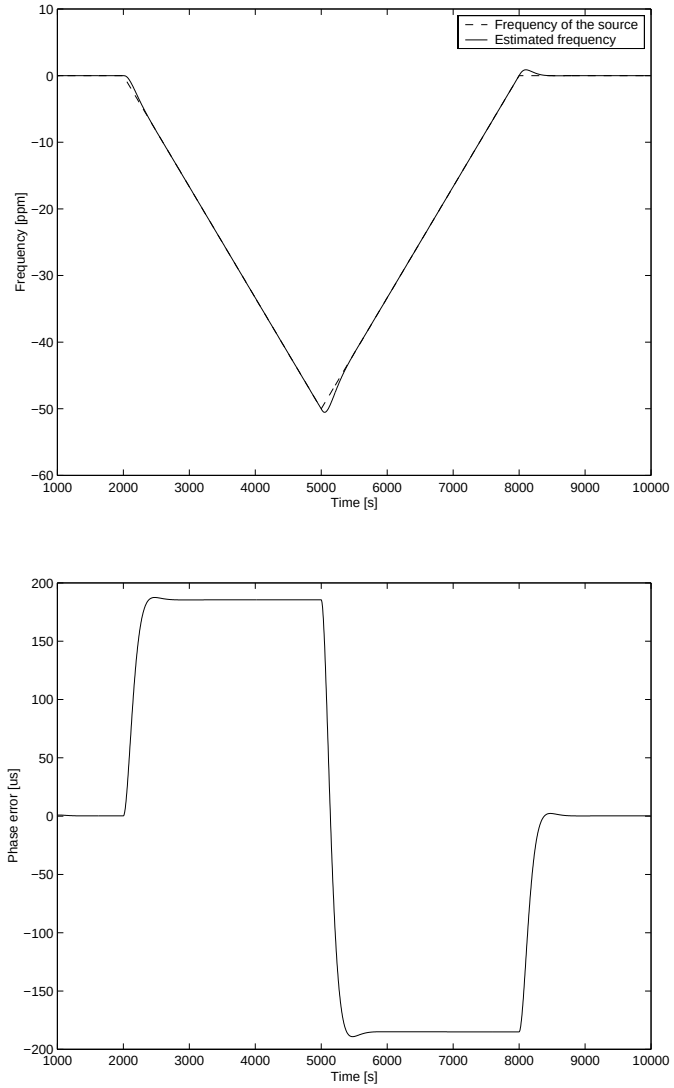

Figure 8.6 Frequency drift of 52 ppm with the system using the second order filter with integral compensation

Comments to Results

In the Figures 8.3-8.6 different behaviours of the system using either filters with or without integral compensation have been shown. When a filter with integral compensation is used, the buffer size of the input buffer is easier to dimension because of the zero steady state error of the phase. In the case where a filter without integral compensation is used, the steady-state error will not be zero and therefore the required buffer size cannot be predicted, because the error depends on the frequency difference between the clock of the source and the sink (which is the parameter we want to

estimate). In the situation shown in Figure 8.3, where the steady state error is ≈ 22 ms the extra buffer to compensate for this has to be at least 22 ms.

8.4.4 Effect of Integral Compensation on Initial Phase Error and Jitter

In the simulations below, the behaviour of the system with different initial *phase errors* is shown. The channel model described in Section 8.2.2 is used in these simulations with a jitter amplitude of 100 ms peak-to-peak. With this model of the channel the maximum initial *phase error* possible is 50 ms (compared to the mean value of the jitter process). Therefore either 0 or -50 ms initial *phase errors* are used in the simulations below. The two filters used in the last simulations are also used here.

In Figure 8.7 the dejittering system, using the filter without integral compensation, starts up with a *phase error* of almost -50 ms. This effects the transient behaviour severely. The *settling time* becomes approximately 750 s, and the frequency will experience an *overshot* of approximately 260 ppm. Such a large frequency error obviously significantly exceeds the 100 ppm region described in Section 2.2.4.

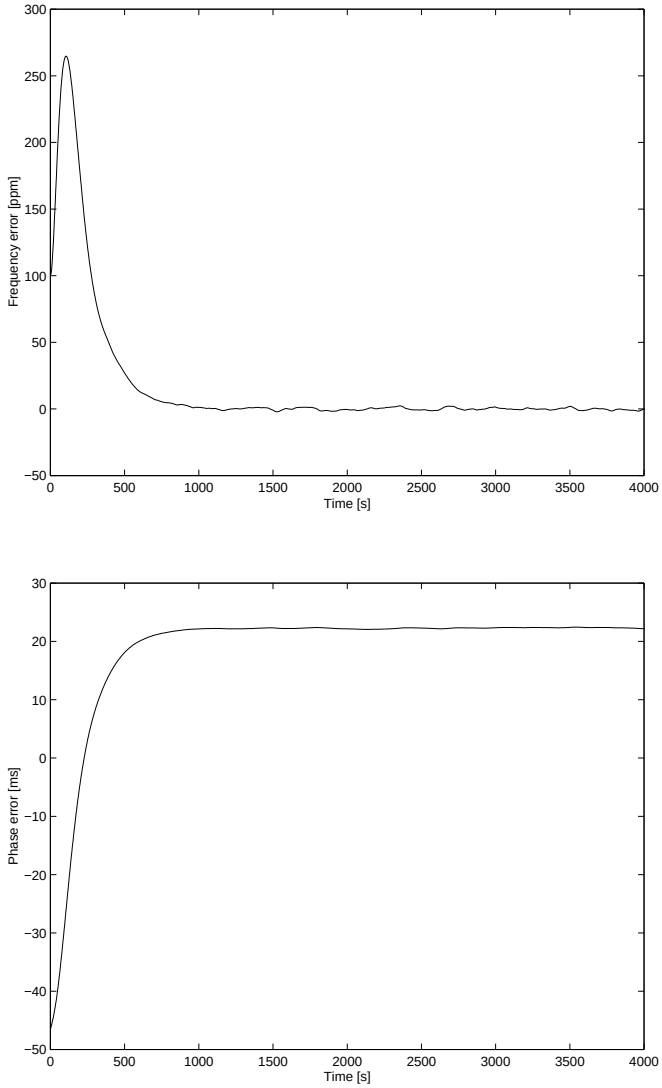

Figure 8.7 Frequency step response with almost -50 ms initial phase error for the system using the second order filter without integral compensation

In Figure 8.8 the system, using the filter without integral compensation, starts up with a zero *phase error*, shown in Figure 8.8. In this case the *frequency error* has no *overshot* and falls within the 100 ppm region.

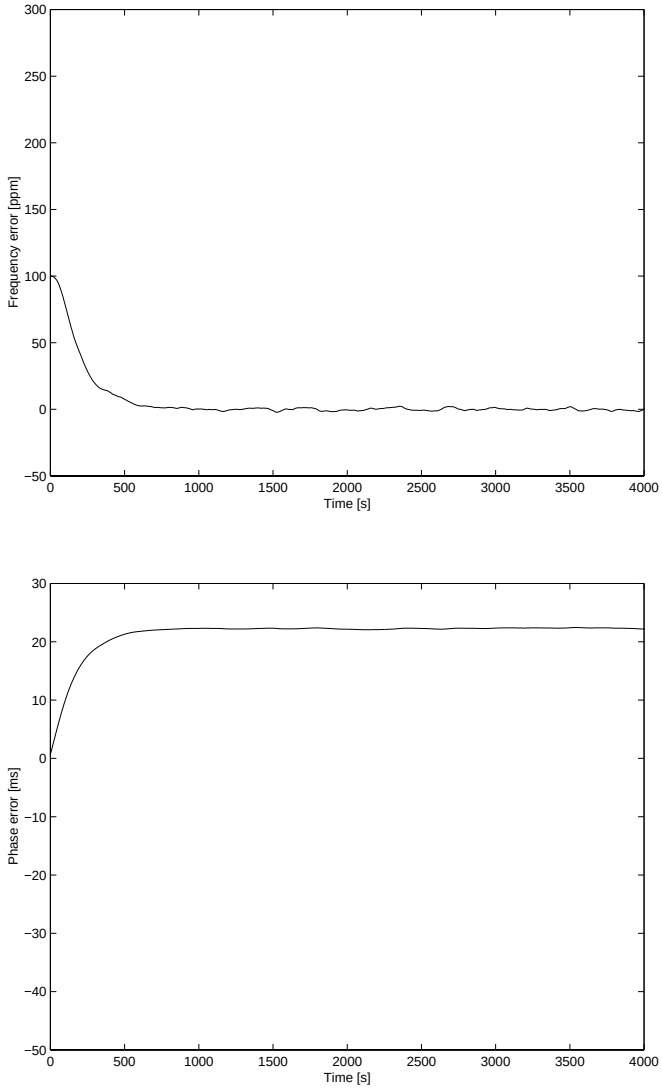

Figure 8.8 Frequency step response with no initial phase error for the system using the second order filter without integral compensation

Figure 8.9 shows the system behaviour, using the filter with integral compensation, and an initial phase error of almost -50 ms. The *settling time* becomes approximately 600 s. As shown in the upper part of Figure 8.9, the *frequency error* has an *overshot* of approximately 650 ppm! The *phase error* has an *overshot* of approximately 16 ms, shown in the lower part of Figure 8.9. Obviously the frequency error is far too high to be acceptable.

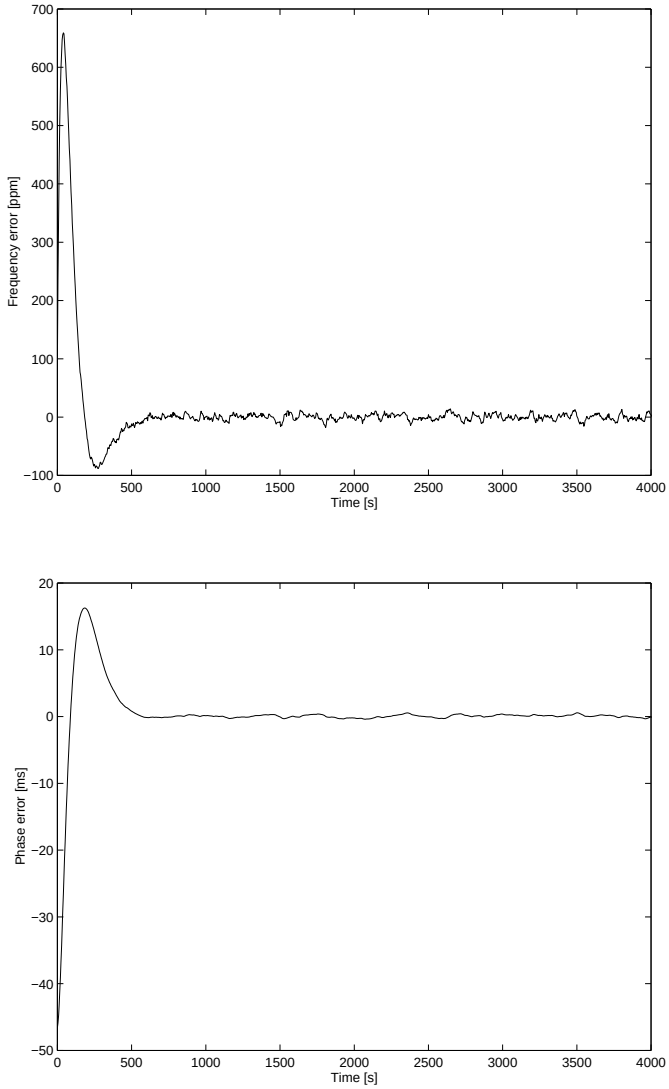

Figure 8.9 Frequency step response with -50 ms initial phase error for the system using second order filter with integral compensation

In Figure 8.10 the system, using the filter with integral compensation, and which starts up with a zero *phase error,* is shown. In this case the *frequency error* has an *undershot* of approximately -30 ppm, shown in the upper part of Figure 8.10. The settling time is now approximately 350 s. As shown in the lower part of the figure the *phase error* has an *overshot* of approximately 5 ms. To conclude, the frequency error is well within the 100 ppm region and the phase error is limited.

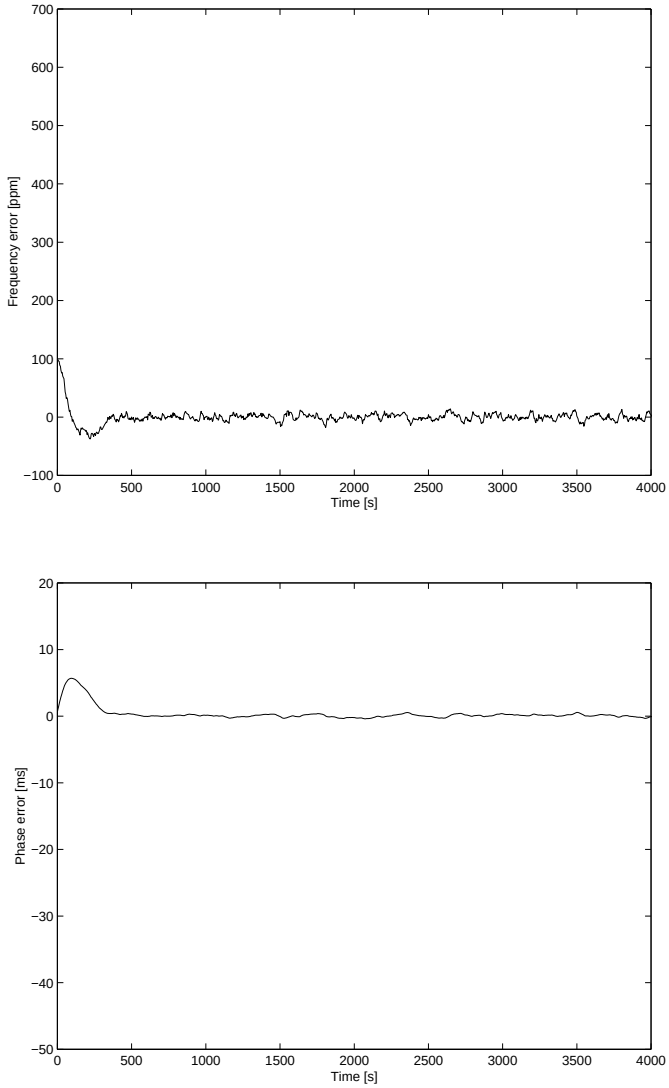

Figure 8.10 Frequency step response with no initial phase error for the system using the second order filter with integral compensation

Comments to results

If the system includes some sort of estimate of the initial phase, one can achieve a much better control of the transient behaviour at start-up. Especially with the filter with integral compensation the frequency step response looks really bad (overshot of 640 ppm), without such an estimation. So, in practise the phase estimation seems to be necessary. It should be noted that with a real jitter distribution, the deterministic bound could be much larger than the maximum initial *phase error* of the simulated jitter process, see Section 4.2.2, so the problem can be worse in reality. An assumption is

made that a correct estimate of the initial phase is made in the rest of the simulations shown below. Therefore a zero initial *phase error* will be used in these simulations.

8.4.5 Results with Improved Filters without Integral Compensation

The last two sections have shown some important difference in behaviour of a system with integral compensation and one without integral compensation. In this subsection and in the following one, we will discuss the achieved system performance vs. the requirements in some more detail.

Below, results from simulations with ordinary Butterworth filter of second order is introduced. Five different Butterworth filters have been chosen with cut-off frequency from 1.8 mHz to 22.5 mHz. Two different gain factors, $K=1\cdot10^{-5}$ and $2\cdot10^{-6}$, are used. An initial relative frequency difference of 100 ppm is chosen in all these simulations. Only simulations with one filter will be introduced in this section, which will be compared to only one filter with integral compensation. The rest of the simulations are presented in Appendix B. The filter used in this section has a gain factor of $K=1\cdot10^{-5}$ and a cut-off frequency of 3.15 mHz. Using formula [7.16] such a filter should give a steady state error of 11 ms.

In Figure 8.11 the frequency response of the dejittering system is shown with this filter. As shown in the figure, the attenuation above 100 Hz is over 270 dB! As mentioned earlier in Section 8.2.2 the jitter process is low pass filtered with a filter with a cut-off frequency of 115 Hz. Therefore that low pass filtering will not affect the results, e.g. regarding jitter filtering performance. It should be noted that, in reality, the system will not have such a good attenuation of high frequencies, because the interpolation done on the incoming timestamps will give some non-linear effects. However these effects are not easy to analyse.

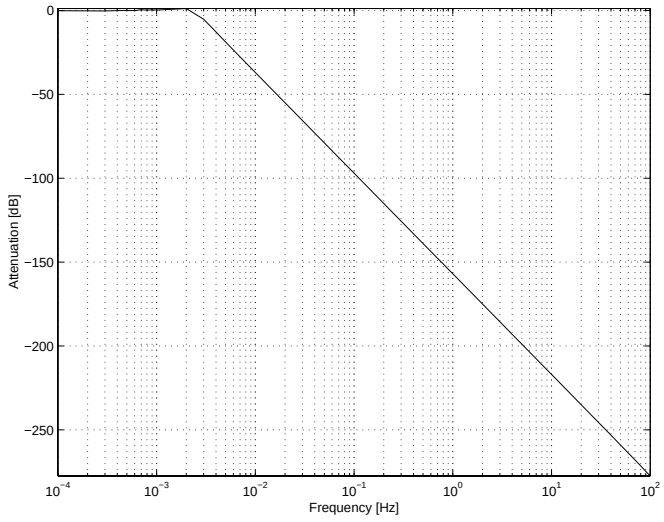

Figure 8.11 Frequency attenuation of the dejittering system using the filter without integral compensation.

As shown in Figure 8.11 the curve falls asymptotically with 60 dB/decade, i.e. it behaves like a third order system, and this can be shown by the equation [7.13].

In Figure 8.12 the frequency step response of the system using the Butterworth filter is shown. The *rise time* is approximately 180 s and the settling time is approximately 360 s as shown in the upper part of Figure 8.12. The phase has an *undershot* of approximately 2.5 ms. The steady state error of the phase is ≈11 ms. Although we have pointed out the problem of predicting the *phase error* of the actual filter type, the results shown in Figure 8.12 would probably be satisfactory in a real decoder. The frequency *undershot* is well within a ±100 ppm region and the frequency variations of the steady state are low, i.e. the jitter is well absorbed. The *phase error* need of course be taken care of by some extra buffer margin.

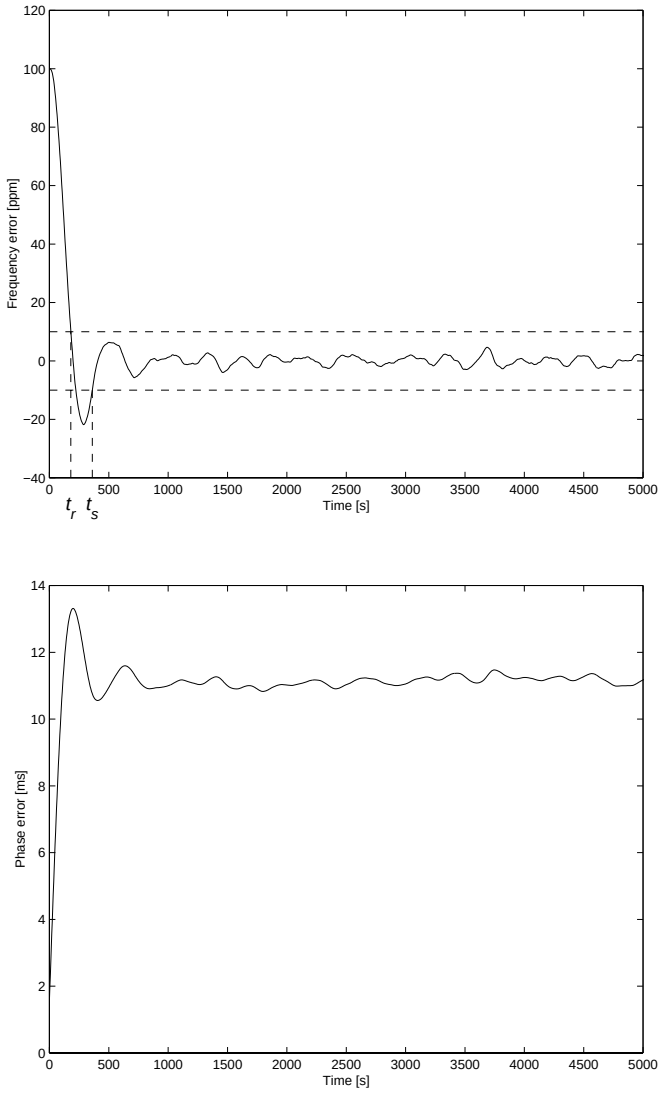

Figure 8.12 Frequency error and phase error of the frequency step response.

In the upper part of Figure 8.13,the *frequency change rate* is shown and in the lower part the residual *jitter* is shown. The *frequency change rate*, shown in the figure, has a maximum of almost –0.7 ppm/s during the transient phase. Thereafter it stabilises to values within ±0.1 ppm/s. The evaluation of these values are somewhat unclear, which will be further discussed in Section 8.4.7.

The residual *jitter* shown in the lower part of Figure 8.13 obviously reached its maximum and minimum values during the transient phase. However, these extreme values are caused by low frequency components, which could be regarded as changes in frequency. The peak-to-peak value of the jitter after the transient phase is approximately 0.09 µs. This is obviously a neglectable jitter amplitude, well within the MPEG-2 RTI which states ±25 µs, see [MPEG2 RTI]. The frequency response shown in Figure 8.11 shows that the attenuation over 0.25 Hz, (which is the cut-off frequency of the high pass filter used to generate the lower part Figure 8.13), is above 120 dB

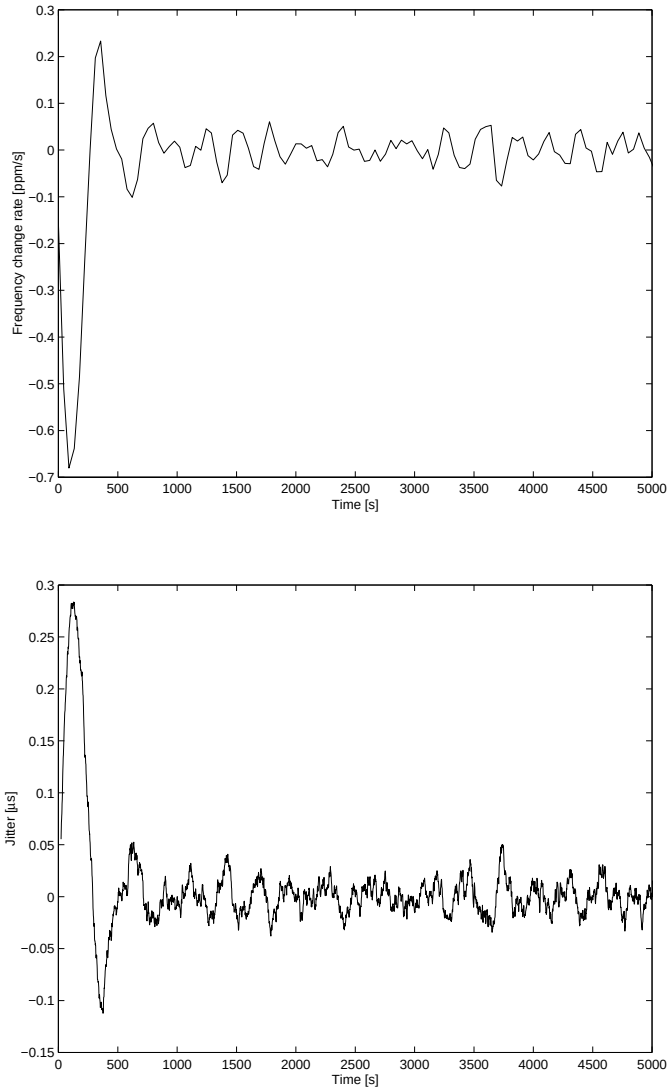

Figure 8.13 Frequency change rate and residual jitter using the same frequency step and jitter process as in the previous cases.

8.4.6 Results with Improved Filters with Integral Compensation

Four different filters with integral compensation have been used in the simulations. They are all of type second order. Only one filter is introduced in this section and is chosen because it shows comparable *rise time* and *settling time* with the filter used in the last section. (The rest of the simulations are introduced in Appendix B.2). The filter used here has a transfer function shown in according to [8.5].

$$H(s) = \frac{K \cdot \left(\frac{s}{\omega_z} + 1 \right)}{s \cdot \left(\frac{s}{\omega_p} + 1 \right)} \quad [8.5]$$

where $K = 5 \cdot 10^{-8}$, $\omega_z = 0.006$ and $\omega_p = 0.03$.

Using this filter, will give a frequency response of the dejittering system shown in Figure 8.14. At 0.25 Hz the attenuation of the dejittering system is above 80 dB. From this curve, one can predict that the system will have a higher residual *jitter* than the system simulated in the last section, (where the attenuation above 0.25 Hz was above 120 dB).

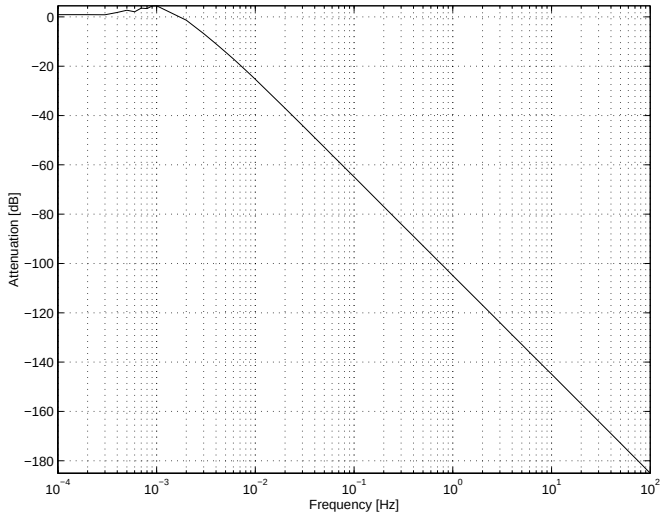

Figure 8.14 Frequency attenuation of the dejittering system using the filter with integral compensation.

As shown in Figure 8.14 the curve falls asymptotically with 40 dB/decade, i.e. it behaves like a second order system. The system is actually a third order system but the zero at $-\omega_z$ in the *S*-domain basically cancels one of the poles.

In Figure 8.15 the frequency step response of the system using the filter with integral compensation is shown. The *rise time* is approximately 140 s and the settling time is approximately 530 s as shown in the upper part of Figure 8.15. The *phase error* has an *overshot* of approximately 9.5 ms. Thereafter the *phase error* stabilises to a zero mean. As can be seen from the upper plot, the jitter causes more noise in the frequency compared to the filter without integral compensation. Still, the frequency variations are small (<±10 ppm).

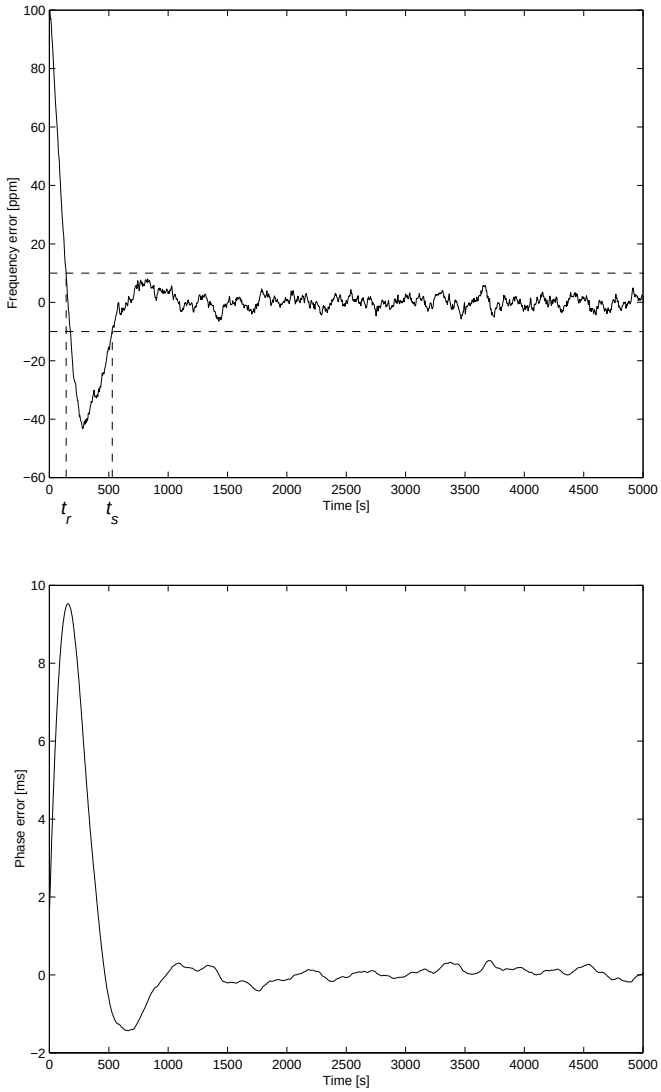

Figure 8.15 Frequency and phase response of a frequency step response.

In the upper part of Figure 8.16, *frequency change rate* is shown and in the lower part the residual *jitter* is shown. The *frequency change rate*, reaches its maximum of almost –0.7 ppm/s under the transient phase. Thereafter it stabilises to values within ±0.15 ppm/s. The peak-to-peak value of the *jitter* after the transient phase is approximately 1 µs. This is obviously an amplitude well below the maximum jitter specified by a MPEG-2 RTI, see [MPEG2 RTI]. The frequency response shown in Figure 8.14 shows that the attenuation over 0.25 Hz (which is the cut-off frequency of the high pass filter used to generate the lower part Figure 8.16), is above 80 dB.

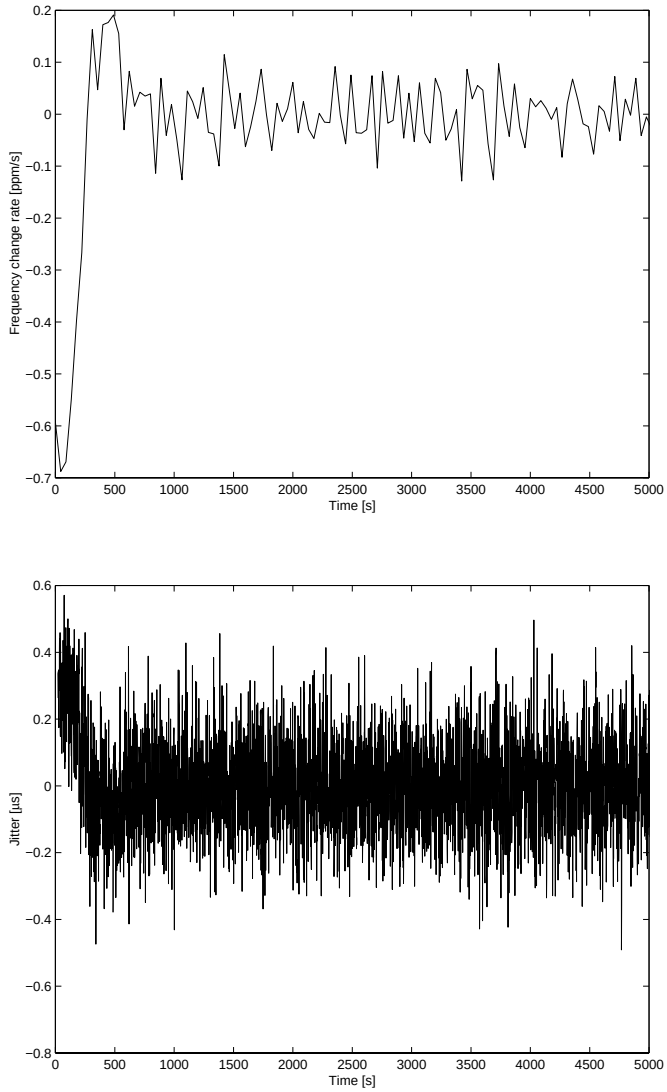

Figure 8.16 Frequency change rate and residual jitter of filter with integral compensation.

8.4.7 Concluding remarks

One problem in analysing the results from the simulations in Sections 8.4.5 and 8.4.6. is that the specifications, regarding the maximum *jitter* and frequency drift an ordinary MPEG-2 decoder and consumer TV set can tolerate, are incomplete. Mainly, the figures that are standardised are the requirements of the transmitting end. These figures are

sometimes confusing. For example, why has the clock drift requirements of the STC in MPEG-2 to be ten times smaller than the clock drift requirements of an analogue PAL signal in [ITU-R 624]?

In RTI of MPEG-2, see [MPEG2 RTI] there is a recommendation that an MPEG-2 decoder should handle 50 µs *jitter*. But it says nothing about the spectral components of the *jitter*. This is highly important because different frequency contents of the *jitter* affect the analogue video signal in different ways. As mentioned earlier, the *jitter* definition used here is the delay variations above 0.25 Hz. Comparing the results from the simulations with the recommendation in RTI of MPEG-2, (50 µs *jitter*) none of the used filters has any problem to cope with this requirement, see Sections 8.4.5-8.4.6 and Appendix B. In the first simulation in Section 8.4.5 the peak-to-peak value of the residual *jitter* is 0.09 µs and the simulation in Section 8.4.6 it is 1 µs. This is about 50-500 times lower than the recommendation in RTI of MPEG-2!

None of these filters can cope with the drift requirements of PAL or the STC in MPEG-2. Both simulations show maximum values of this drift, (*frequency change rate*), about 0.7 ppm/s, which are 30 times larger than the requirements of a PAL signal in [ITU-R 624] and 250 times larger than the requirements of the STC in MPEG-2. As mentioned earlier, these requirements are stated for the transmitting end and say nothing about the requirements of the receiver. It is expected that a consumer TV set will accept a much higher drift rate than those specifications.

The attenuation of the *jitter* is probably much better than what is necessary. As can be seen in Appendix B, there is a strong correlation between relatively high *jitter* attenuation and low maximum *frequency change rate*, so even if the system has a satisfactory attenuation of the *jitter* the maximum *frequency change rate* can be too large. To know how high change rate a real MPEG-2 decoder or/and a consumer TV set can handle, tests have to be made. The problem of high *frequency change rate*s in the transient phase of clock recovery schemes using DPLL was already mentioned in 1993 by Scott Quinn, see [Quinn 93], where the recovered clock had drift peaks 100 times larger than the required 0.75 Hz/s!

Neither are the exact requirements of the accuracy of the frequency really known. As mentioned in Section 2.2.4 the requirement of the accuracy of a PAL signal is about ±1 ppm, but this figure is only standard for studio equipment and is not relevant for the case studied here. The requirement of the STC in MPEG-2 is ±30 ppm. This requirement cannot be satisfied in the cases simulated above.

If the initial relative frequency difference is 100 ppm, the simulated system will always start with this relative *frequency error*. How a complete system will behave during the transient phase, depends on the synchronisation mechanism of the following MPEG-2 decoder, (typically a DPLL as described in Figure 3.4). If the synchronisation circuit has an acquisition range in the region of the initial frequency error there will be no problems and the STC of the MPEG-2 decoder will follow the dejittered clock.

Even if the MPEG-2 decoder has not got a sufficient acquisition range to lock into the initial frequency error in the transient phase, problems may not occur. Say for example that the DPLL of the MPEG-2 decoder has a maximum acquisition range of ±30 ppm and the initial frequency error is 100 ppm. At the start-up, the synchronisation circuit of the MPEG-2 decoder will then enter its maximum deviation point, +30 ppm.

The MPEG-2 decoder will then start up playing out the decoded video with a rate which is 70 ppm too slow, according to the dejittering system. When the dejittered clock from the dejittering system has reached 30 ppm frequency error the MPEG-2 STC will track the frequency changes of this dejittered clock.

If the MPEG-2 decoder has sufficient buffer size this initial difference in clock frequency between the dejittering system and the MPEG-2 decoder will not be a problem. Say for example that the system uses the filter simulated in Section 8.4.6. As shown in Figure 8.15 it takes approximately 105 s for the system to reach 30 ppm frequency error. During this time period the buffer level of the MPEG-2 decoder will be rising because of the too slow play out rate of the decoder. In Figure 8.17 a hypothetical transient phase is shown. The extra required buffer size can be calculated from the shaded area in Figure 8.17. If the shaded area is approximated by a triangle the extra-required buffer size will be ≈3.7 ms, ($70 \cdot 10^{-6} \cdot 105 \cdot 0.5 = 3.7$ ms). If the same calculations are made of the system simulated in Section 8.4.5 the extra buffer size will be ≈5.3 ms ($70 \cdot 10^{-6} \cdot 150 \cdot 0.5 = 5.3$ ms). Those values are small enough to cause neglectable additional delay in a decoder implementation.

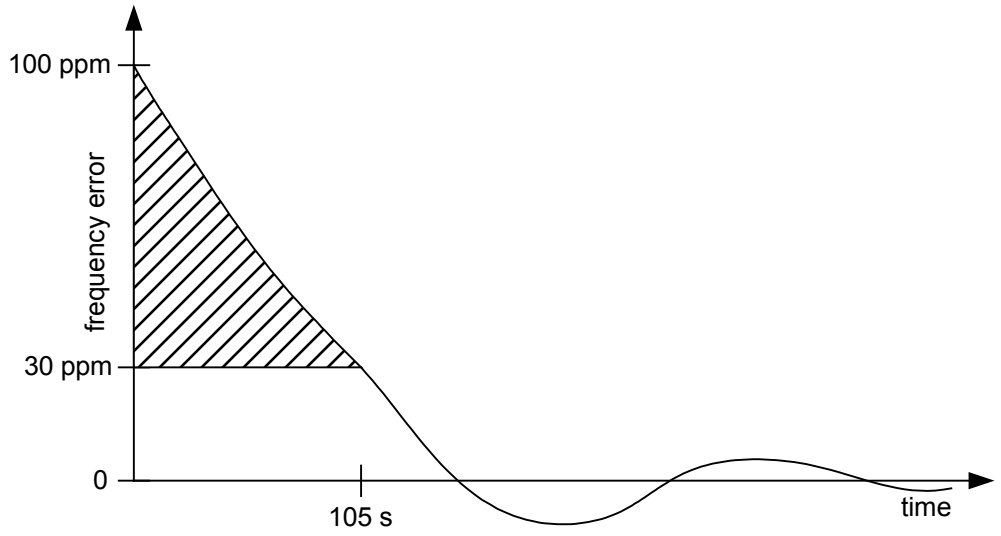

Figure 8.17 Hypothetical transient phase.

The variations of the *phase error* during the transient phase is not a problem for the MPEG-2 decoder or the display device. It is mainly a problem for the input buffer to the dejittering system. The phase error is actually indicating the fullness of the input buffer to the dejittering system and therefore dictates the required input buffer size. The buffer size has to be at least the size of the statistical bound of the delay variation to prevent buffer over- and underflow. To absorb the variations of the *phase error* some additional buffer is needed. In the simulation of Section 8.4.6, the maximum *phase error* is ≈9.5 ms, which is the additional buffer size needed, and in the simulation in 8.4.5 the additional needed buffer size is ≈13.5 ms. So if the dejittering system is designed to handle 100 ms of jitter amplitude, about 15 % of margin is needed using these filters.

There is a difference in the requirements of buffer size between the two systems simulated in Sections 8.4.7 and 8.4.8. The first system, which uses a filter without integral compensation, has a steady-state *phase error* of ≈11 ms and therefore, after the

transient phase, it always requires this additional buffer. However, as described in 8.4.3, the required buffer margin cannot be predicted, since it depends on the input signal. In the second simulation using the filter with integral compensation, which has an almost zero steady-state error of the phase, requires almost no additional buffer after the transient phase. Because of its long-term minimum buffer size requirement, filters with integral compensation are preferable.

Comparing with the results of others'

Now it is time to compare the results from the simulations presented here with other research works done in the area. Two papers mentioned in Section 6.4 are used in this section. The first one [Parekh 97] is an adaptive buffer algorithm, which monitors the fullness of the input buffer to control the play out rate from this buffer. In their simulations they use a relative frequency difference of 40 ppm and a jitter amplitude, peak-to-peak, of 16 ms. In the simulations their scheme manages to reduce this jitter to about 1 ms of residual jitter. The residual jitter above 10 Hz is reduced to 200-350 μs. This is achieved with a settling time, i.e. within the ±10 ppm region, of approximately 100 s. So, the jitter attenuation is not very high. The residual jitter amplitude violates the MPEG-2 RTI specification.

In the other paper [Tryfonas 96] an ordinary DPLL, as shown in Figure 3.4, was used to dejitter MPEG-2 streams over ATM networks. The filter used in the loop is a Butterworth filter of second order with a cut-off frequency of 0.1 Hz (30 times higher than the one used in Section 8.4.5). In all simulations presented, a relative frequency error of 30 ppm between the transmitter and the receiver was used. In one experiment a heavily loaded ATM network was simulated resulting in a jitter amplitude peak-to-peak of 21.6 ms. In this experiment the DPLL had no chance to lock to the clock of the transmitter and the recovered frequency varied with more than ±30 ppm frequency error (more than the initial relative frequency error). The frequency change rate had maximum values of over 2 ppm/s. The results are not strictly comparable with the results presented in this section, because the DPLL used was designed to work in ATM networks and was therefore not designed to handle these high jitter amplitudes. Probably, the DPLLs simulated in [Tryfones 96] could have been designed to handle larger delay variations if the cut-off frequency of the Butterworth filter would have been lowered. There is another reason why this paper is not directly comparable to the results presented in this section, and that is the input rate of timestamps to the DPLL. This is about 20 times lower than the one used in the case simulated in this section. One peculiar thing is that Tryfonas does not even mention loop filters with integral compensation. The DPLL simulated in [Tryfonas 96] probably produces steady-state error, which is small enough, so the addition delay is not an issue.

9 Discussion and Conclusions

9.1 Conclusions drawn from Simulations

Traditionally, it has been debated whether MPEG-2 based high quality audio-visual steaming, suited for TV or HDTV services, requires a low jitter bearer network to work. Examples of such networks are ATM based networks or broadcast type networks like DVB. The QoS requirements of these services have typically been assumed to imply a delay variation of less than a couple of milliseconds or potentially a few tens of milliseconds.

If an IP-based network is to be used for this kind of service it has often been assumed that one need to accept a looser synchronisation, where frame skipping/ frame repetition is unavoidable. Examples of such implementations are commercially available PC-based streaming clients, making use of plug-in MPEG-2 decoder boards without a true PCR clock recovery. (This is the most common type of MPEG-2 decoder boards.)

In this thesis, it has been shown that it is possible to design a dejittering scheme capable of filtering 100 ms of peak-to-peak IP packet delay variation, producing a residual jitter amplitude in the order of a microsecond. Such a low jitter amplitude is obviously well below the MPEG-2 RTI specification of ±25 μs maximum PCR jitter, making the scheme a candidate for implementations of fully synchronised MPEG-2 decoders in IP environments. It also matches the performance requirements that can be expected in consumer TV environments. So, an IP based TV or HDTV service obviously need not be based on looser synchronisation requirements, like frame skipping scheme, but could very well match the performance of the existing digital TV services, based on DVB, from an image and sound quality point of view.

However, it should be noted that an implementation of the dejittering scheme, e.g. in a set-to box or a PC, implies several potential issues, like the real time support of the operating system, see below.

It has been shown in this thesis, that it is possible to combine an extremely low pass filtering with a sufficiently small phase error, which is important since a small phase error is needed to limit the added delay margin. However, a mechanism to minimise the initial phase error is needed, which has been shown in the simulations. A possible solution will be briefly discussed below.

Provided the system clock used in the PC or the set-top box is within a 100 ppm accuracy, the proposed scheme matches what is expected to be the accuracy requirements of the colour sub-carrier of a consumer TV set. However, there is one uncertainty and that is the drift requirements of such a TV set, which have to be further investigated.

In this thesis, it is assumed that the following decoder makes a "true" PCR clock recovery and therefore the performance criteria are determined accordingly. The scheme can of course also be used in a situation where the following decoder uses a looser, class B, synchronisation. In such a case the performance criteria are probably somewhat different from the ones used in this thesis. (Maybe minimising the peak values of the

phase error is more important than a stable frequency error, to prevent buffer under- and overflows.)

Actually, as a secondary effect, the dejittering scheme achieves a recovery of the PCR time base. Probably, some optimisations can be made to use this time-base also in the decoding process in some way, especially if the following decoder uses a class B synchronisation. If the decoding process is made in software, like a software MPEG-2 decoder in a PC, the time base can probably be directly used in the decoding process.

9.2 Implementation into a Real System

In this thesis the proposed dejittering scheme has only been tested in simulations and some aspects have therefore been neglected, which can cause problems in a real implementation.

In the simulations, an assumption is made that the relative frequency difference between the clock of the sender and the system clock used in the dejittering process at the receiver is 100 ppm. As discussed in Section 2.2.4, a consumer TV set can typically handle frequency error in the region of ±100 ppm, so in this case the video signal can probably be displayed from start up of the dejittering system. I.e. the transient phase does not cause any problems regarding frequency deviation. As mentioned in Section 8.2.3, 100 ppm is not an absolute upper bound for the frequency difference but is chosen to reflect the typical performance of a personal computer or workstation. In a case of a much higher frequency error, the video signal cannot be displayed at start up, and one has to wait for the frequency error to settle in the region of 100 ppm frequency error. The scheme can still be usable in applications where the video signal is to be displayed on a computer monitor. Another solution to the problem is if the synchronisation circuit implemented in the MPEG-2 decoder limits this frequency deviation to the required region. The problem then disappears and the video signal can still be displayed at start up in a TV set.

Another aspect not treated in the simulation is the effect of limited resolutions of the floating point numbers used in a possible implementation. A too coarse limitation of this resolution can degrade the jitter attenuation performance of the system. Floating points with a resolution of 8 bytes have been used in all processing and calculations of filter coefficients in the simulations, which is also the resolution of an ordinary Intel Pentium processor, so this will probably not be any problem in a real implementation.

In the simulations an approximately infinite clock tick representation is used in the dejittering process. In a real system this will of course not be the case, and this reduction of the resolution will give an additional contribution to the jitter of the output packet stream from the dejittering system, and therefore limits the total possible reduction of jitter of the system.

In addition to the issues above, the main problem to solve in an implementation is probably around the real time support of the chosen operating system. Since such issues are outside the scope of this thesis, they are not further dealt with here.

9.3 Further Work

If the scheme is to be implemented in a real system some further investigations have to be made, which are briefly pointed out below.

- Some further tuning of the filter coefficients used in the simulations can always be made to get improved locking frequency step responses with the same reduction of jitter.

- More research is required to get knowledge of representative frequency characteristics of the delay variations in typical IP based networks, on which the system can be used. This is important when the filter coefficients are to be designed in a real implementation.

- Further work is required to study implementation aspects when the scheme is to be implemented using a commercially available operating system. Especially issues around real time support of common operating systems, typically used in PCs and set-top boxes, have to be studied.

- Maybe the problem of high initial phase errors can be reduced if the system is also synchronised to NTP (Network Time Protocol). RTCP SR packets can be used to estimate the current phase of the RTP timestamp clock from the NTP timestamps provided in these packets. This has to be further investigated.

Abbreviations

ATM	Asynchronous Transfer Mode
CBR	Constant Bit Rate
DCT	Discrete Cosine Transform
DPLL	Digital Phase Locked Loop
DTS	Decoding Time Stamp
DVB	Digital Video Broadcasting
DVD	Digital Versatile Disc
ES	Elementary Stream
HDTV	High Definition Television
IGMP	Internet Group Management Protocol
IP	Internet Protocol
LAN	Local Area Network
LDU	Logical Data Unit
LPF	Low Pass Filter
MTU	Maximum Transmission Unit
MPEG	Moving Pictures Expert Group
MPEG2	Generic video coding standard
NTP	Network Time Protocol
PCR	Program Clock Reference
PES	Packetised Elementary Streams
PLL	Phase Locked Loop
PS	Program Stream
PTS	Presentation Time Stamp
PU	Presentation Unit
QoS	Quality of Service
RGB	Red Green Blue
RTI	Real Time Interface
RTP	Real Time Protocol
RTCP	Real Time Control Protocol
SCR	System Clock Reference
STC	System Time Clock
TCP	Transmission Control Protocol
TS	Transport Stream
UDP	User Datagram Protocol
VBR	Variable Bit Rate
VCO	Voltage-Controlled Oscillator

References

[Andreotti 95] G.F. Andreotti, G.Michieletto, L. Mori and A. Profumo, *"Clock Recovery and Reconstruction of PAL Pictures for MPEG Coded Streams Transported Over ATM Networks"*, IEEE Transactions on circuits and systems for video technology, Vol. 5, No. 6, December 1995.

[Best 93] R. Best, *"Phase-Locked Loops: theory, design and applications"*, ISBN 0-07-911386-9, McGraw-Hill, 1993.

[Blom 89] G. Blom, *"Sannolikhetsteori och statistikteori med tillämpningar"*, ISBN 91-44-03594-2, Studentlitteratur, 1989.

[Bolot 93] J-C. Bolot, *"Characterizing End-to-End Packet Delay and Loss in the Internet"*, Journal of High-Speed Networks, Vol 2, No. 3, December 1993.

[Cheng 93] C. Chen, *"Analog and Digital Control System Design: Transfer-Functions, State-Space, and Algebraic Methods"*, ISBN 0-03-094070-2, Saunders Collage Publishing 1993.

[Class 97] C. Class, *"Synchronization Issues in Distributed Applications: Definitions, Problems, and Quality of Synchronization"*, TIK-Report No. 31, December 1997.

[Enstedt 88] E. Enstedt, *"TV-teknikens grundläggande principer"*, ISBN 91-44-27931-0, Studentlitteratur 1988.

[Forchheimer 96] R. Forchheimer, *"Image Coding and Data Compression"*, Dept. of Electrical Engineering, Linköping University 1996.

[Glad 97] T. Glad, L. Ljung, *"Reglerteori"*, ISBN 91-44-00472-9 Studentlitteratur, 1997.

[Haskell 96] G. Haskell, A. Puri, A.N. Netravali, *"Digital Video: An Introduction to MPEG-2"*, ISBN 0-41-208411-2, Chapman & Hall, 1996.

[ITU-R 624] ITU-R Report 624-4.

[ITU-T G.114] ITU-T Recommendation G.114, *"One way transmission time"*, 1997.

[Karlsson 96] G. Karlsson, *"Asynchronous Transfer of Video",* IEEE Communications Magazine Aug. 1996.

[Kwok 95] T. Kwok, *"A Vision for Residential Broadband Services: ATM-to-the-Home"*, IEEE Network, October 1995.

[MPEG2 Sys] *"Coding of Moving Pictures and Associated Audio"*, CD 13818-1 (MPEG-2 Systems), ISO/IEC, November 1994.

[MPEG2 RTI] *"Coding of Moving Pictures and Associated Audio"*, CD 13818-9 (Real Time Interface Specification), ISO/IEC, July 1996.

[Noro 99a] R.Noro, M. Hamdi and J.P. Hubaux, *"Circuit Emulation over IP Networks"*, IFIP 6th International Workshop on Protocols for High-Speed Networks, Salem- MA, USA, Aug. 99, pp. 187- 201

[Noro 99b] R. Noro and J.P.Hubaux, *"Clock Synchronization of MPEG-2 Services over Packet Networks"*, Telecommunication Systems Journal, vol. 10, n. 1-2, pp. 3-16, Mar. 1999.

[Parekh 97] S. P. Parekh, *"Jitter and clock recovery with dejitterization for CBR MPEG-2 video over ATM networks"*, Packet Video conferencing proceeding 1997.

[Peters 85] J. Peters, *"Television 50 years"*, European Broadcasting Union 1985. http://www.dvb.org/dvb_articles/dvb_tv-history.pdf

[Ptolemy] The homepage of Ptolemy, http://www.ptolemy.eecs.berkeley.edu/

[RFC2250] RFC 2250, *"RTP Payload Format for MPEG1/MPEG2 Video"*, January 1998.

[Quinn 93] S. Quinn, *"MPEG-2/ATM System Clock Recovery"*, overhead slides, Bellcore, November 93.

[Ramjee 94] R. Ramjee, J. Kurose, D. Towsley, *"Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks"*, Infocom '94, March 1994.

[Rudkin 97] S. Rudkin, A. Grace and M. W. Whybray, *"Real-time applications on the Internet"*, BT Technology Journals Vol. 15 No. 2 April 1997.

[Singh 94] R.P. Singh, *"Jitter and Clock recovery for Periodic Traffic in Broadband Packet Networks"*, IEEE Trans. on Communications, Vol. 42, No 5, May 1994.

[Schulzinne 99] *"RTP: A Transport Protocol for Real-time Applications"*, Internet-draft February 1999.

[Stallings 97] W. Stallings, *"Data and Computer Communications"*, ISBN 0-13-571274-2, Fifth Edition, Prentice Hall 1997.

[Steinmetz 96] R. Steinmetz, *"Human Perception of Jitter and Media Synchronization"*, IEEE Journal on Selected Areas in Communications, Vol 14, No 1, January 1996.

[Stevens 94] R. Stevens, *"TCP/IP Illustrated Volume 1, The protocols"*, ISBN 0-201-63346-9, Addison Wesley, 1994.

[Tryfonas 96] C. Tryfonas, *"MPEG-2 Transport over ATM Networks"*, Master Thesis, University of California Santa Cruz, September 1996.

[Vogel 95] A. Vogel, B. Kerherve, G. Bochmann and J. Gecsei,*"Distributed Multimedia Applications and Quality of Service - A survey"*, IEEE Multimedia, Volume: 2 2 , Summer 1995.

[Wolf 97] C. Wolf, C. Griwodz, R. Steinmetz, *"Multimedia Communication"*, Proceedings of the IEEE, Vol. 85, No. 12, December 1997.

A Appendix: Mathematical Derivations

A.1 Derivation of Transfer Function

The linear part of the dejittering system (without the interpolation) is shown in Figure A.1. An assumption that all signals can be transformed to *Z*-domain is made. The input signals to the system are $T'_{rx}(z)$ and $T'_{l}(z)$. The output signal is $\hat{T}_{tx}(z)$. The extra delay in the loop z^{-1} is needed to make the system implementable.

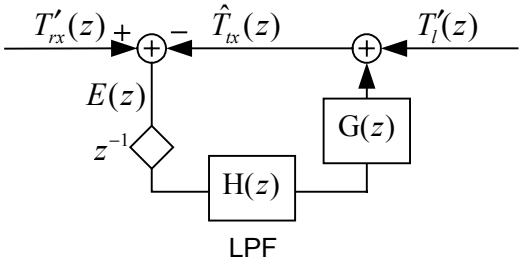

Figure A.1 The dejittering system in the Z-domain

$E(z)$ is defined as the difference between $T'_{rx}(z)$ and $T'_{l}(z)$ see Figure A.1

$$E(z) = T'_{rx}(z) - \hat{T}_{tx}(z) \quad \text{[A.1]}$$

From Figure A.1 it can be seen directly that

$$\hat{T}_{tx}(z) = T'_{l}(z) + z^{-1} \cdot E(z) \cdot H(z) \cdot G(z) \quad \text{[A.2]}$$

Now, if [A.1] is inserted in [A.2] one gets

$$\hat{T}_{tx}(z) = T'_{l}(z) + z^{-1} \cdot \left(T'_{rx}(z) - \hat{T}_{tx}(z)\right) \cdot H(z) \cdot G(z) \quad \text{[A.3]}$$

$$\hat{T}_{tx}(z) \cdot \left(1 + z^{-1} \cdot H(z) \cdot G(z)\right) = z^{-1} \cdot T'_{rx}(z) \cdot H(z) \cdot G(z) + T'_{l}(z) \quad \text{[A.4]}$$

$$\hat{T}_{tx}(z) = \frac{z^{-1} \cdot T'_{rx}(z) \cdot H(z) \cdot G(z) + T'_{l}(z)}{\left(1 + z^{-1} \cdot H(z) \cdot G(z)\right)} \quad \text{[A.5]}$$

$G(z)$ is a discrete integration and its transfer function is given in [A.6].

$$G(z) = \frac{1}{1 - z^{-1}} \quad \text{[A.6]}$$

If one [A.6] is inserted in [A.5] one gets

$$\hat{T}_{tx}(z) = \frac{z^{-1} \cdot T'_{rx}(z) \cdot H(z) + T'_{l}(z) \cdot (1 - z^{-1})}{\left((1 - z^{-1}) + z^{-1} \cdot H(z)\right)} \quad \text{[A.7]}$$

$H(z)$ is a low pass filter and can be written as

$$H(z) = K\frac{h_b(z)}{h_a(z)} \quad [A.8]$$

If now [A.8] is inserted in [A.7] one gets

$$\hat{T}_{tx}(z) = \frac{z^{-1}\cdot T'_{rx}(z)\cdot K\frac{h_b(z)}{h_a(z)} + T'_l(z)\cdot(1-z^{-1})}{\left((1-z^{-1}) + z^{-1}\cdot K\frac{h_b(z)}{h_a(z)}\right)} \quad [A.9]$$

or

$$\hat{T}_{tx}(z) = \frac{z^{-1}\cdot T'_{rx}(z)\cdot K\cdot h_b(z) + T'_l(z)\cdot(1-z^{-1})\cdot h_a(z)}{\left(h_a(z)\cdot(1-z^{-1}) + z^{-1}\cdot K\cdot h_b(z)\right)} \quad [A.10]$$

If [A.10] now is written in matrix notation with $T'_l(z)$ and $T'_{rx}(z)$ as input signals and $\hat{T}_{tx}(z)$ as output signal one gets

$$\hat{T}_{tx}(z) = \frac{1}{z^{-1}\cdot K\cdot h_b(z) + h_a(z)\cdot\left(1-z^{-1}\right)}\cdot\left[z^{-1}\cdot K\cdot h_b(z) \quad h_a(z)\cdot\left(1-z^{-1}\right)\right]\cdot\begin{bmatrix} T'_{rx}(z) \\ T'_l(z)\end{bmatrix} \quad [A.11]$$

A.2 Derivation of Steady State Error Equation

In this section it is assumed that the coming packets to the receiver has experienced zero delay variations and that there is no relative drift between the two clocks at the transmitter and the receiver, i.e. ε is a constant. These assumptions are made, only to simplify the derivation made below.

The frequency of T_{tx} is called f_t and the frequency of T_l is called f_l and the sampling frequency of the dejittering system is called f_s. Then the clock of the receiver T_l makes $\frac{f_l}{f_s}$ clock ticks per sampling instant.

In Figure A.2 the linear part of the dejittering system is shown. The signal $f(n_s)$ is defined as the output signal from the low pass filter $H(z)$. When the system has locked to the clock of the transmitter, the slope of the output signal $\hat{T}_{tx}(n_s)$ has stabilised to the constant value $(1-\varepsilon)\frac{f_l}{f_s}$, see equation [7.2] (in equation [7.2] the slope is described in the time base n but in here it is described in the time base n_s), which is the slope of the clock of the transmitter T_{tx}.

When the discrete integrator $G(z)$ is driven by a constant input signal it will output a ramp signal with a slope equal to the value of the input signal. Therefore the signal

$f(n_s)$ has converged to $-\varepsilon\frac{f_l}{f_s}$ and in turn the error signal $e(n_s)$ has to be converged to the constant value $\frac{-\varepsilon \cdot f_l}{K \cdot f_s}$, where K is the gain of the filter $H(z)$ at zero frequency.

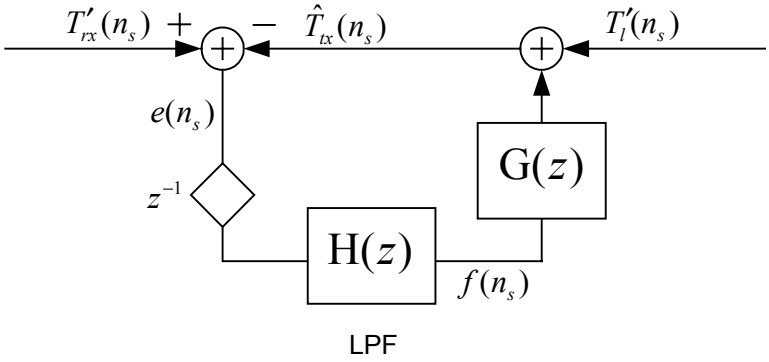

Figure A.2 The dejittering system in the time domain

Then, the steady state can be described with the equation

$$E\{e\} = \frac{-\varepsilon \cdot f_l}{K \cdot f_s} \quad \text{[A.12]}$$

Now, the task is to describe ε with f_t and f_l. The clock of the transmitter T_{tx} makes $\frac{f_t}{f_s}$ clock ticks per sampling instant of the dejittering system and in turn $\frac{f_t}{f_l}$ clock ticks per clock tick of the clock T_l. Then $(1-\varepsilon)$ from equation [7.2] can be described with the equation

$$(1-\varepsilon) = \frac{f_t}{f_l} \quad \text{[A.13]}$$

Then, ε can be described by, using [A.13]

$$\varepsilon = \frac{f_l - f_t}{f_l} \quad \text{[A.14]}$$

if [A.14] is inserted in [A.12] one gets

$$E\{e\} = \frac{(f_t - f_l)}{K \cdot f_s} \quad \text{[A.15]}$$

In [A.15] the steady state error is measured in number of clock ticks of T_l. In [A.16] the steady state error is instead measured in seconds.

$$E\{e\} = \frac{1}{K \cdot f_s} \cdot \left(\frac{f_t - f_l}{f_l}\right) \quad \text{[A.16]}$$

B Appendix: Additional Simulations

B.1 Butterworth filters of second order

B.1.1 Overview

All filters without integral compensation that are used in these simulations can be written in the form shown in equation [B.1].

$$H(z) = K\frac{(1+z^{-1})^2}{1+a_1z^{-1}+a_2z^{-2}} \quad \text{[B.1]}$$

The filters used are all Butterworth filters. In these simulations six different filters are used given in Table B.1.

Filter no.	K	Cut-off frequency [mHz]
1	$5\cdot10^{-6}$	1.8
2	$5\cdot10^{-6}$	4.5
3	$5\cdot10^{-6}$	22.5
4	$1\cdot10^{-5}$	3.15
5	$1\cdot10^{-5}$	4.5
6	$1\cdot10^{-5}$	22.5

Table B.1 Butterworth filters of second order.

B.1.2 Simulations

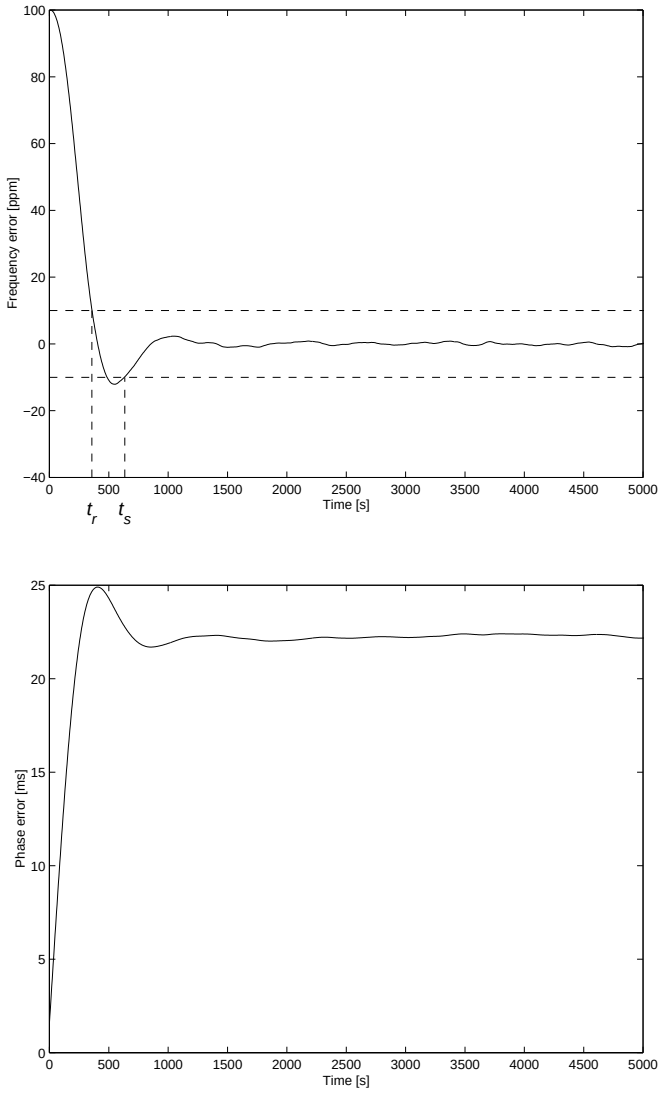

Figure B.1 Frequency and phase error of Filter no. 1.

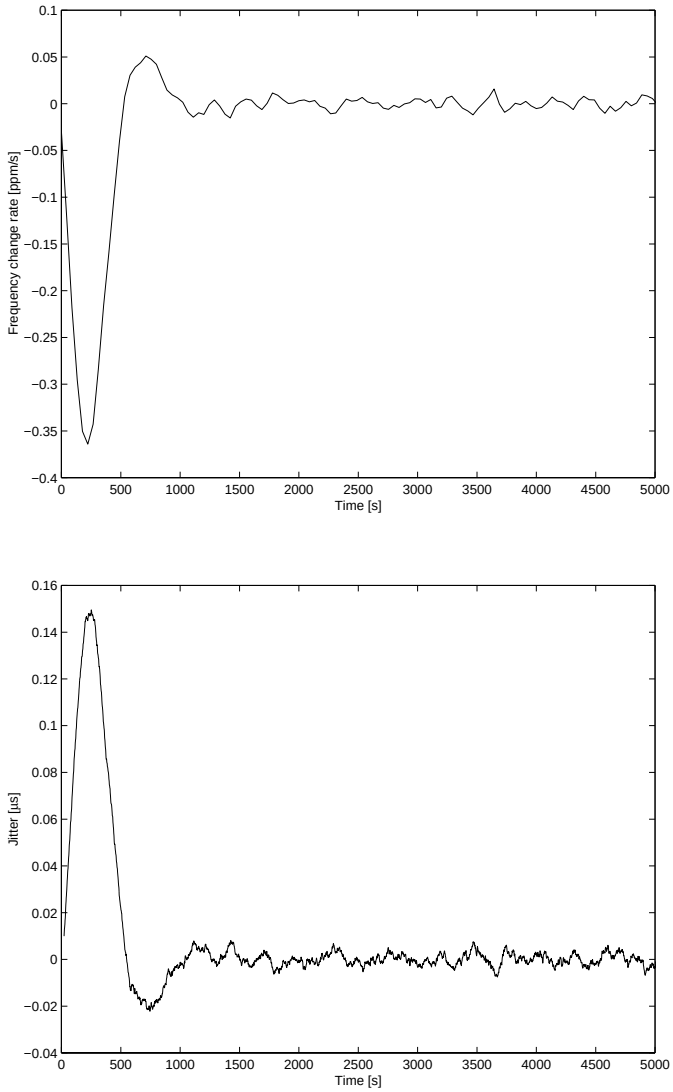

Figure B.2 Frequency change rate and jitter of Filter no. 1.

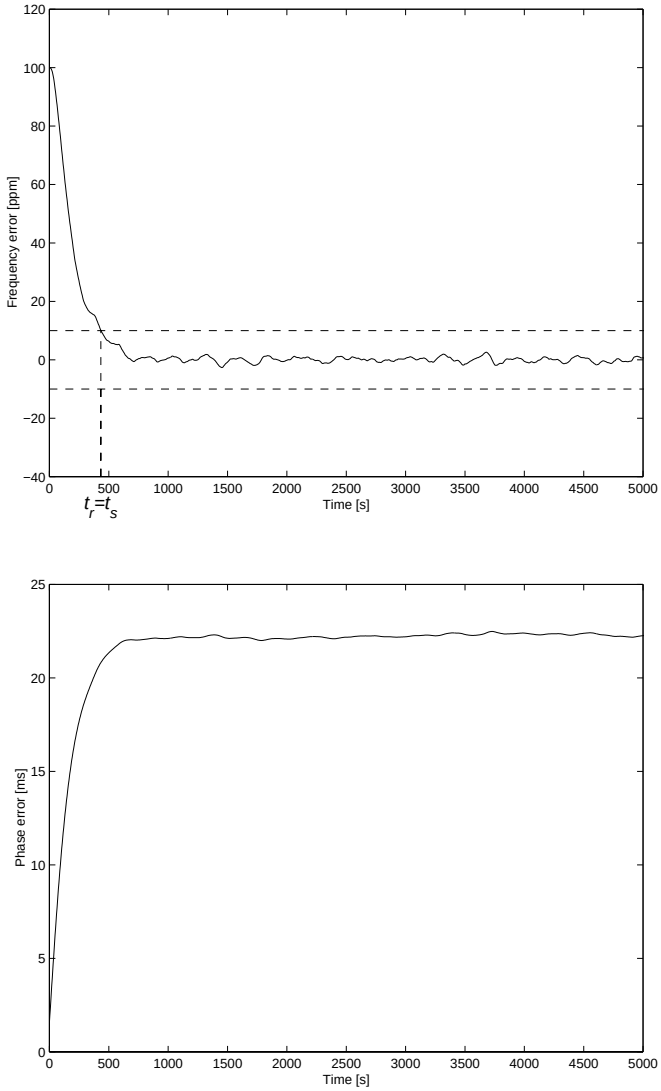

Figure B.3 Frequency and phase response of Filter no. 2.

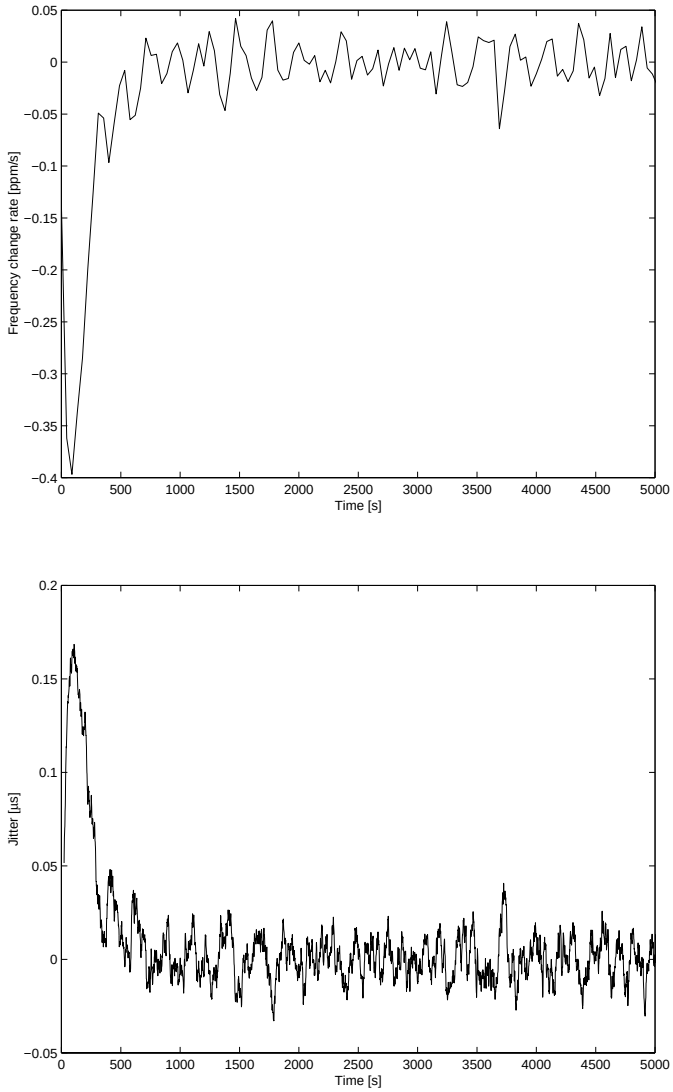

Figure B.4 Frequency change rate and jitter of Filter no. 2.

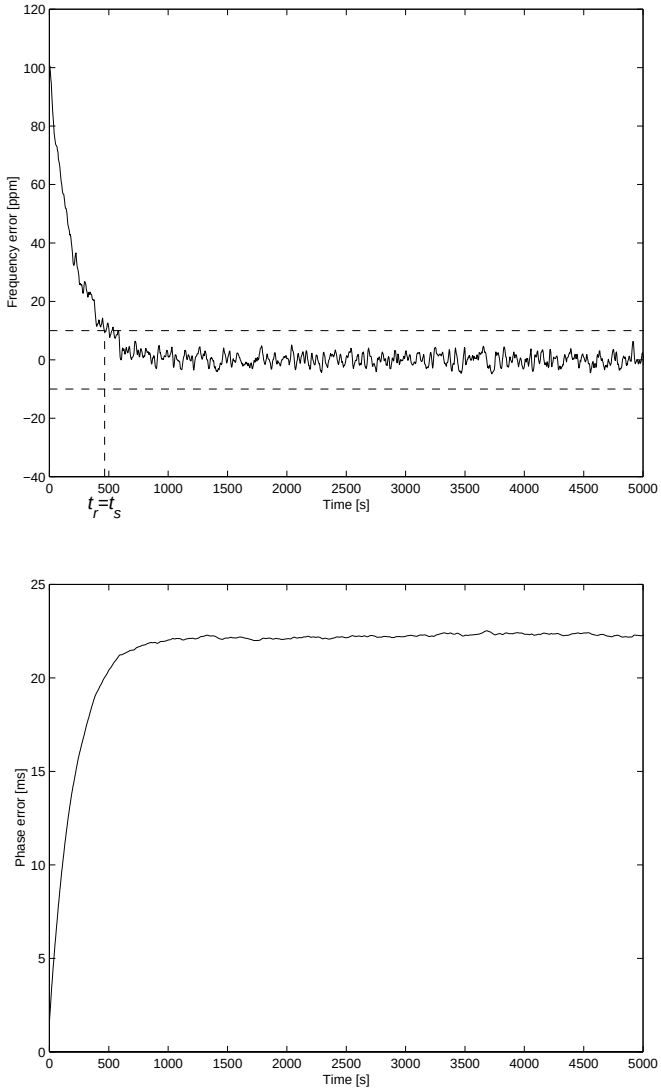

Figure B.5 Frequency and phase response of Filter no. 3.

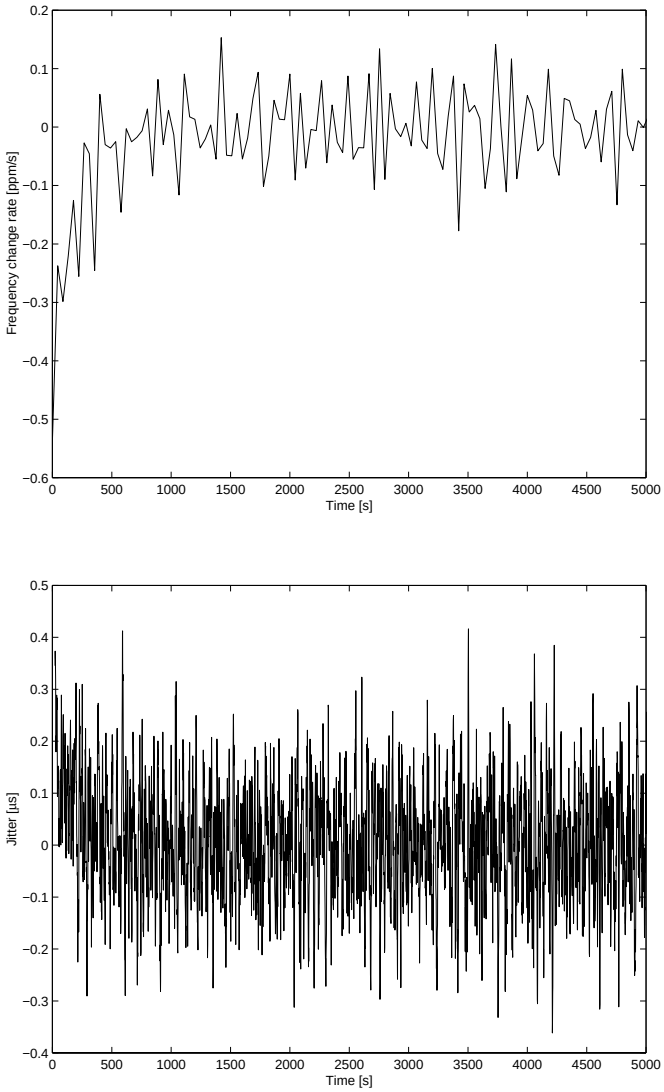

Figure B.6 Frequency change rate and jitter of Filter no. 3.

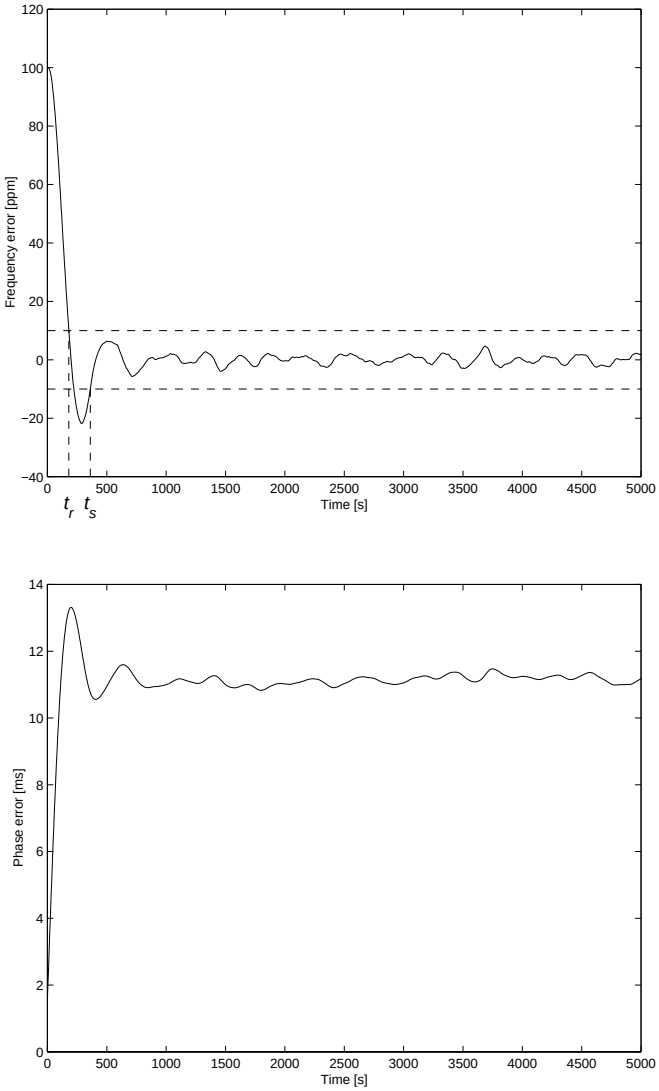

Figure B.7 Frequency and phase response of Filter no. 4.

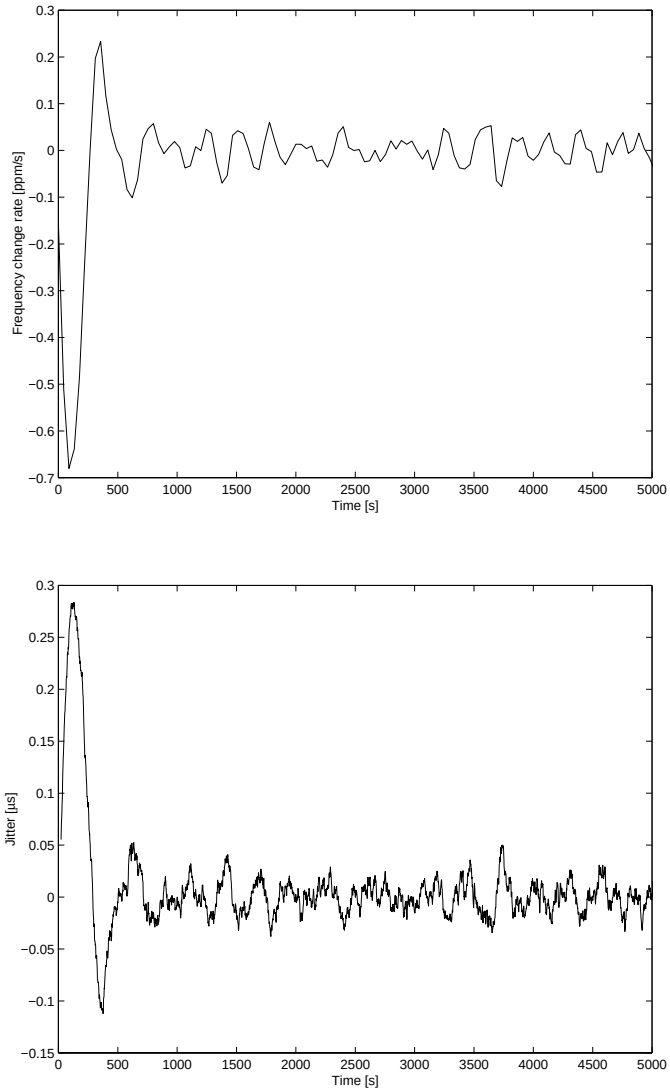

Figure B.8 Frequency change rate and jitter of Filter no. 4.

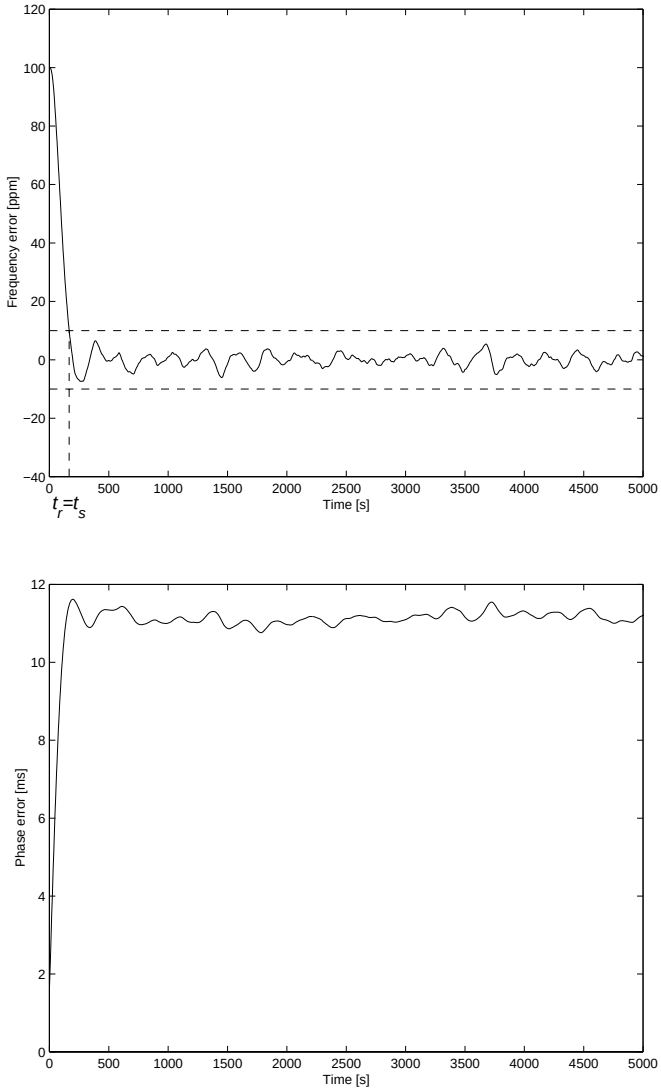

Figure B.9 Frequency and phase response of Filter no. 5.

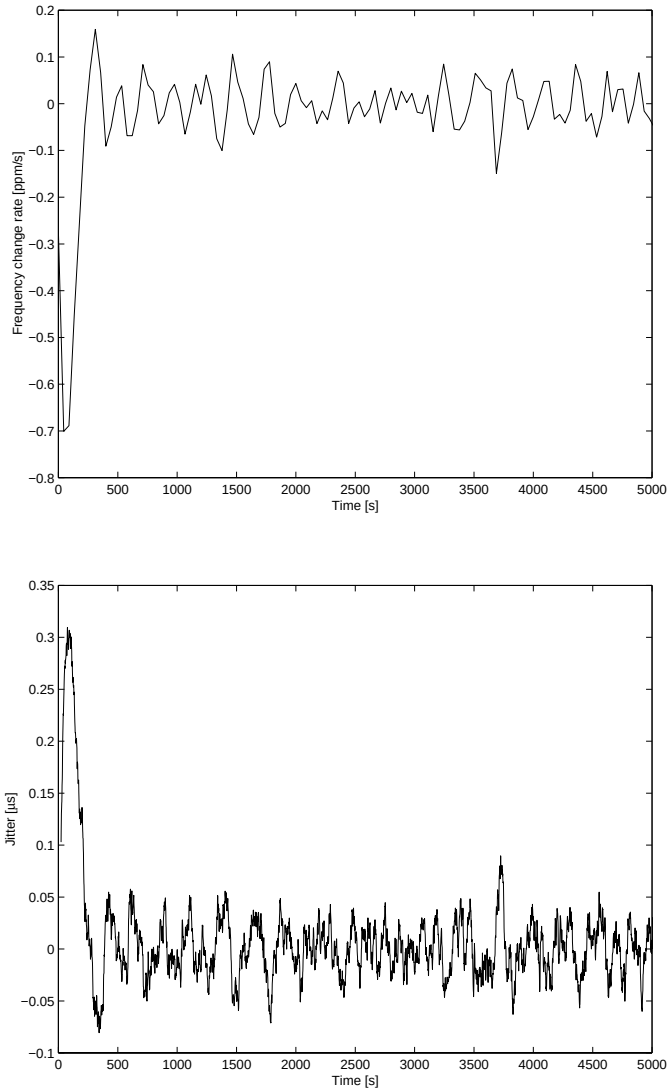

Figure B.10 Frequency change rate and jitter of Filter no. 5.

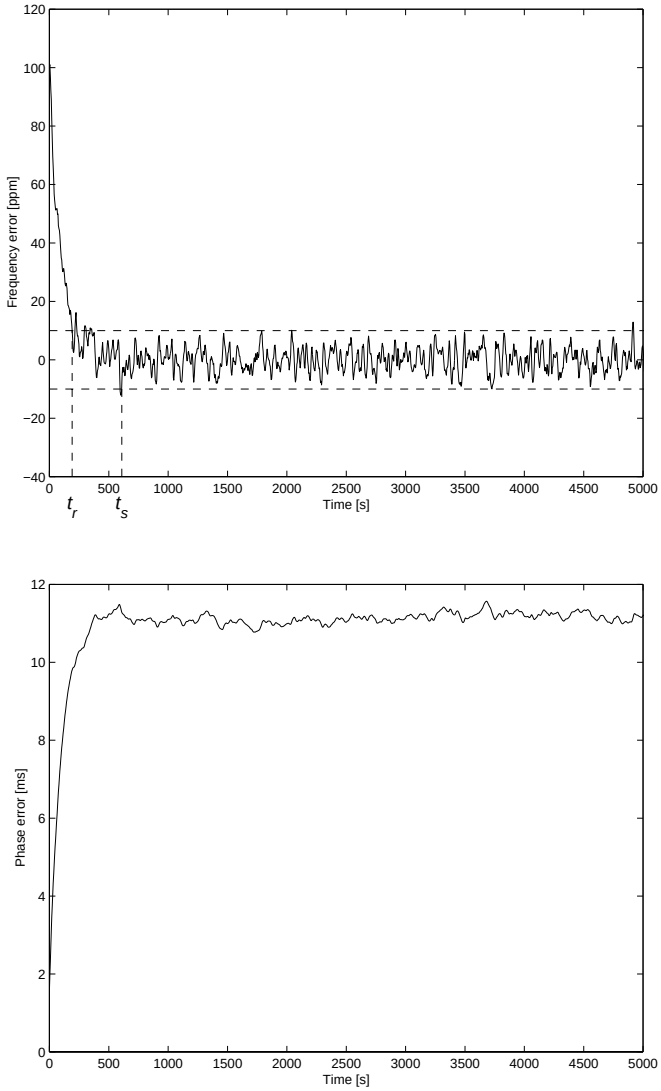

Figure B.11 Frequency and phase error of Filter no. 6.

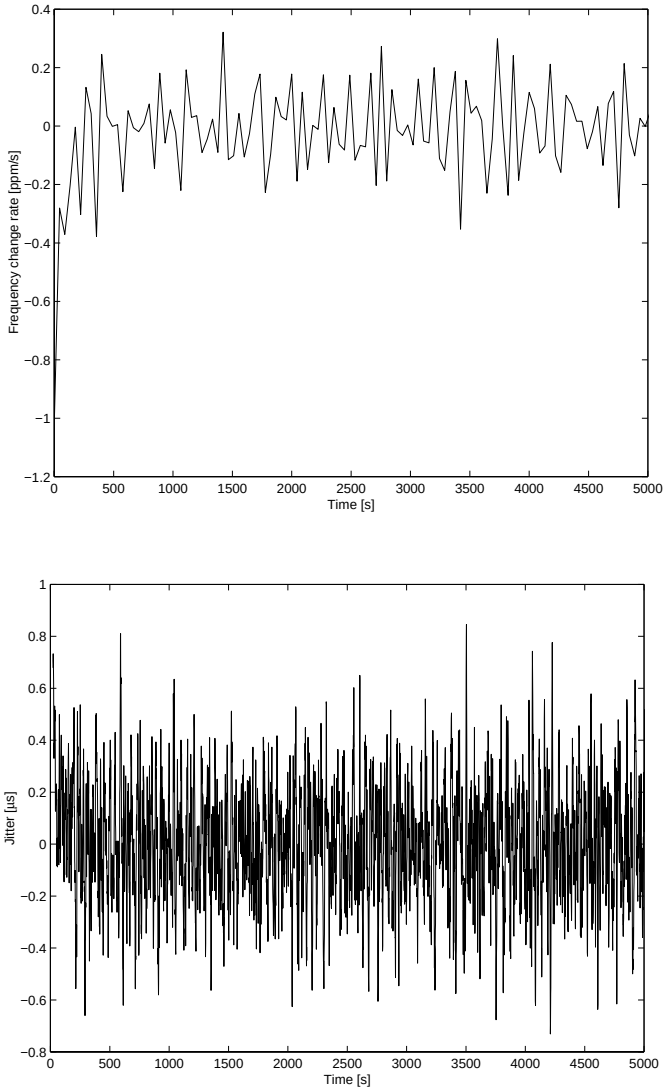

Figure B.12 Frequency change rate and jitter of filter no. 6.

B.1.3 Results

Filter no.	Rise time [s]	Settling time [s]	Jitter [µs]
1	358	635	0.018
2	433	433	0.073
3	465	465	0.777
4	180	361	0.088
5	166	166	0.160
6	192	610	1.77

Table B.2 Results from simulations with Butterworth filters of second order

B.2 Filters with integral compensation

B.2.1 Overview

All filters with integral compensation that are used in these simulations can be written in the form shown in equation [B.2].

$$H(z) = \frac{K \cdot \left(\frac{s}{\omega_z} + 1 \right)}{s \cdot \left(\frac{s}{\omega_p} + 1 \right)} \quad [B.2]$$

where ω_z and ω_p indicate the position of the zero and the pole. To transform these filters to the Z-domain the bilinear transform is used. The bilinear transformation maps the S-domain to the Z-domain by

$$s = 2 f_s \frac{z-1}{z+1} \quad [B.3]$$

where f_s is the sampling frequency. In these simulations four different filters are used given in table B.3.

Filter no.	K	ω_z	ω_p
1	$5 \cdot 10^{-8}$	$\frac{3}{500}$	$\frac{3}{100}$
2	10^{-7}	$\frac{3}{500}$	$\frac{3}{100}$
3	$5 \cdot 10^{-8}$	$\frac{3}{500}$	$\frac{3}{56}$
4	10^{-7}	$\frac{3}{500}$	$\frac{3}{56}$

Table B.3 Filters with Integral compensation

B.2.2 Simulations

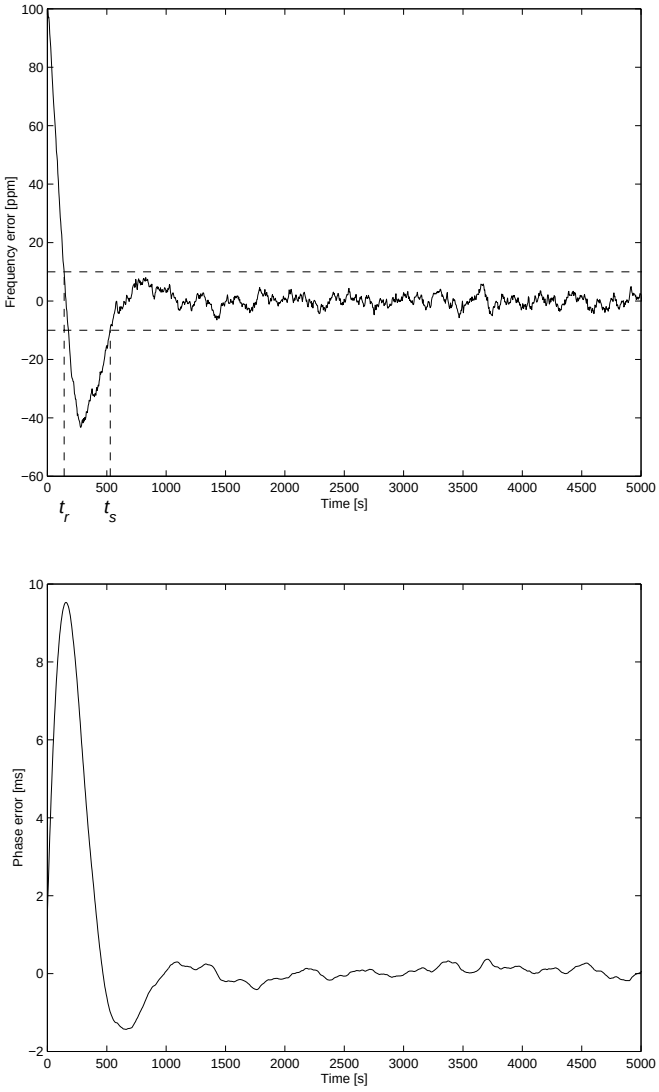

Figure B.13 Frequency and Phase error of Filter no. 1.

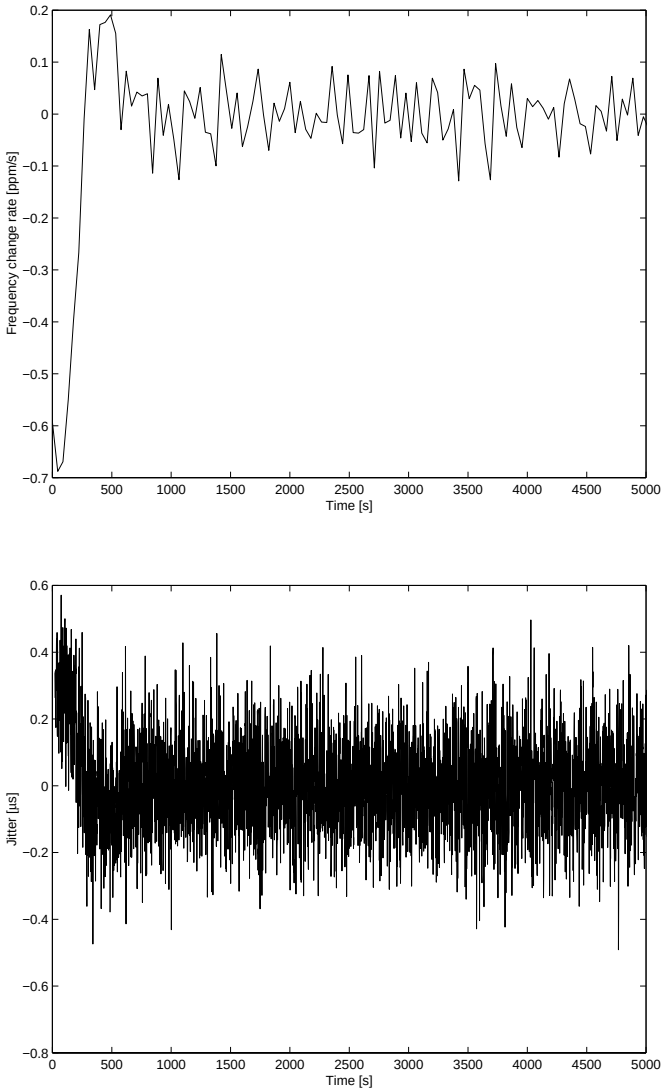

Figure B.14 Frequency change rate and jitter of Filter no. 1.

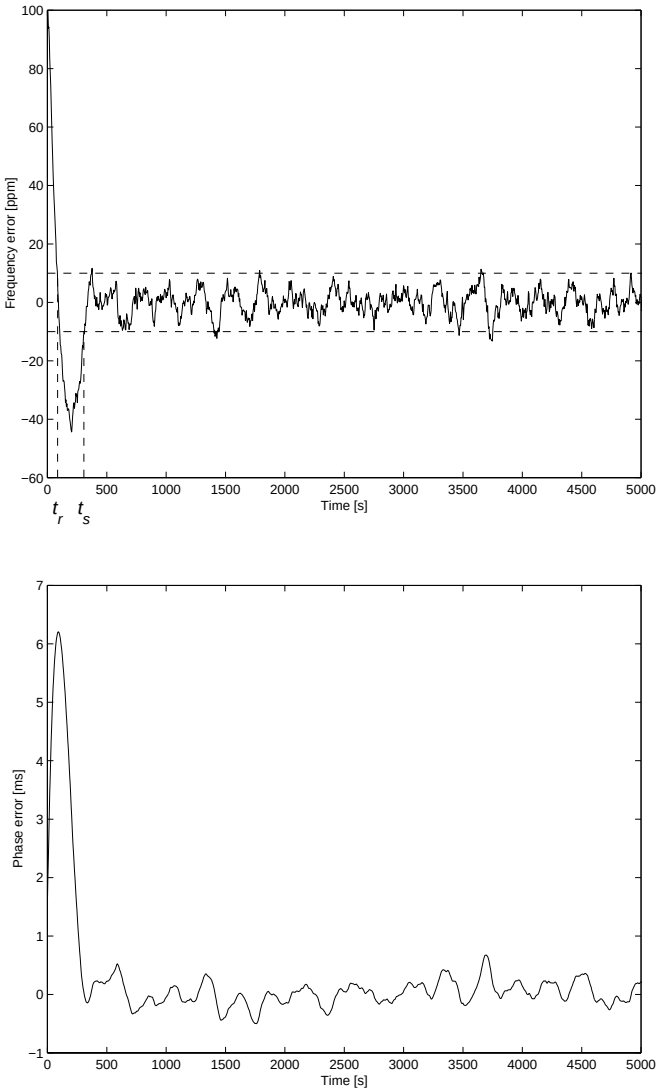

Figure B.15 Frequency and phase error of Filter no. 2.

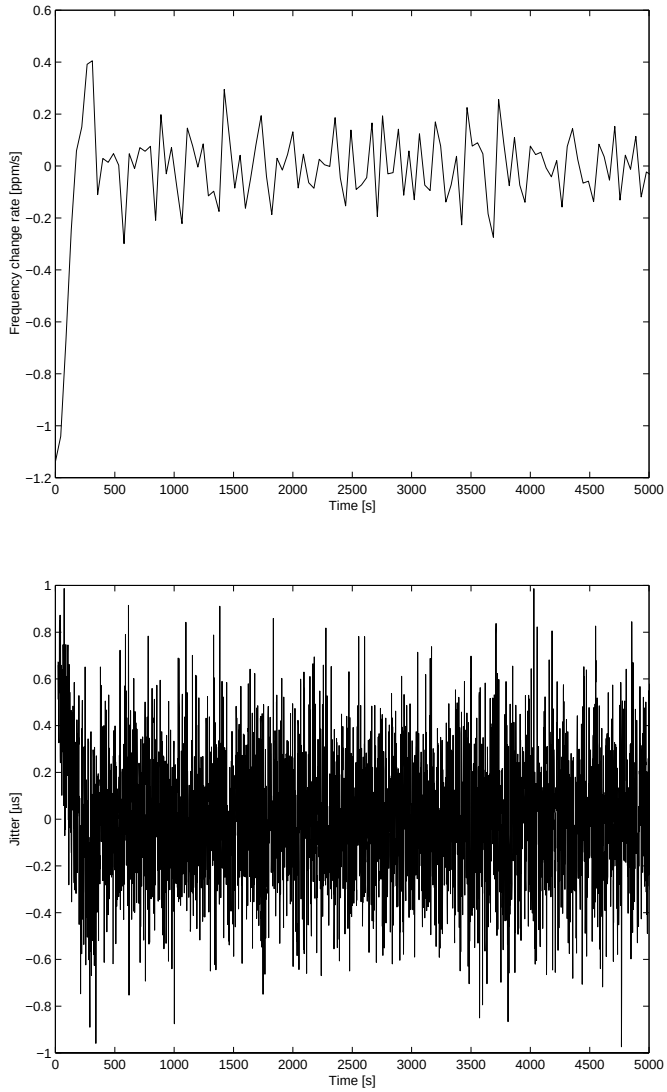

Figure B.16 Frequency change rate and jitter of Filter no. 2.

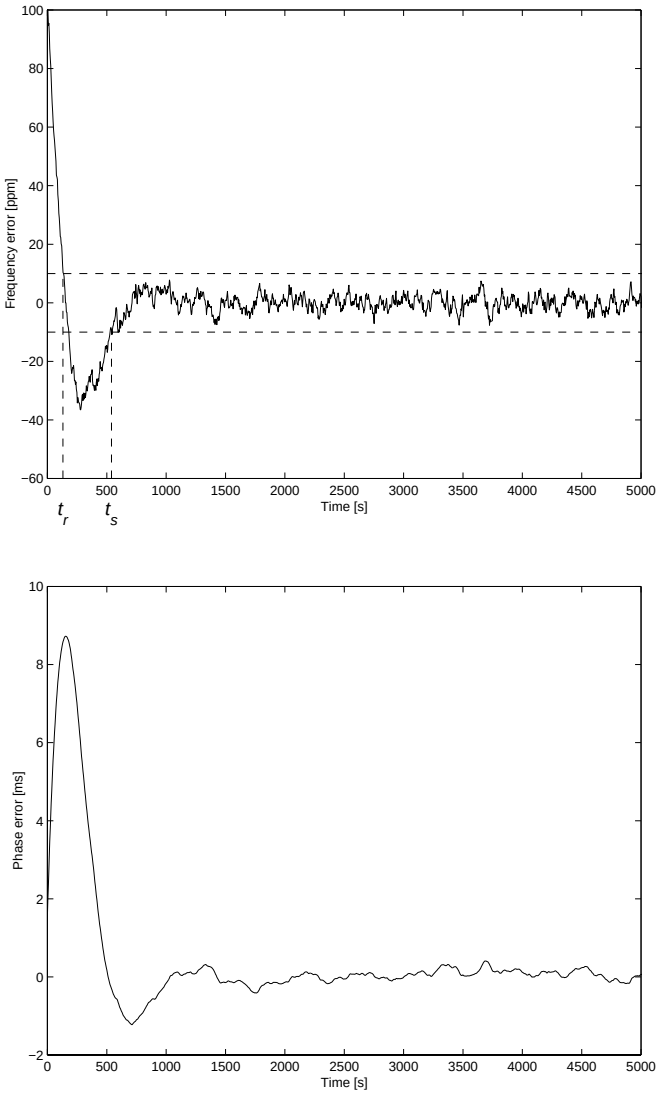

Figure B.17 Frequency and phase error of Filter no. 3.

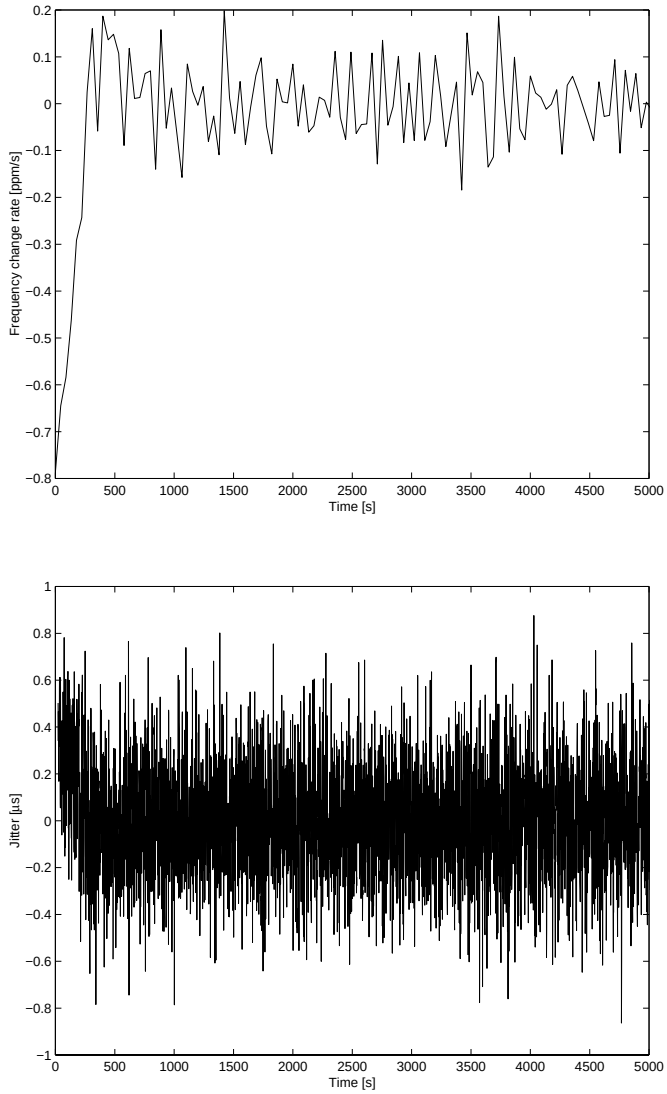

Figure B.18 Frequency change rate and jitter of Filter no. 3.

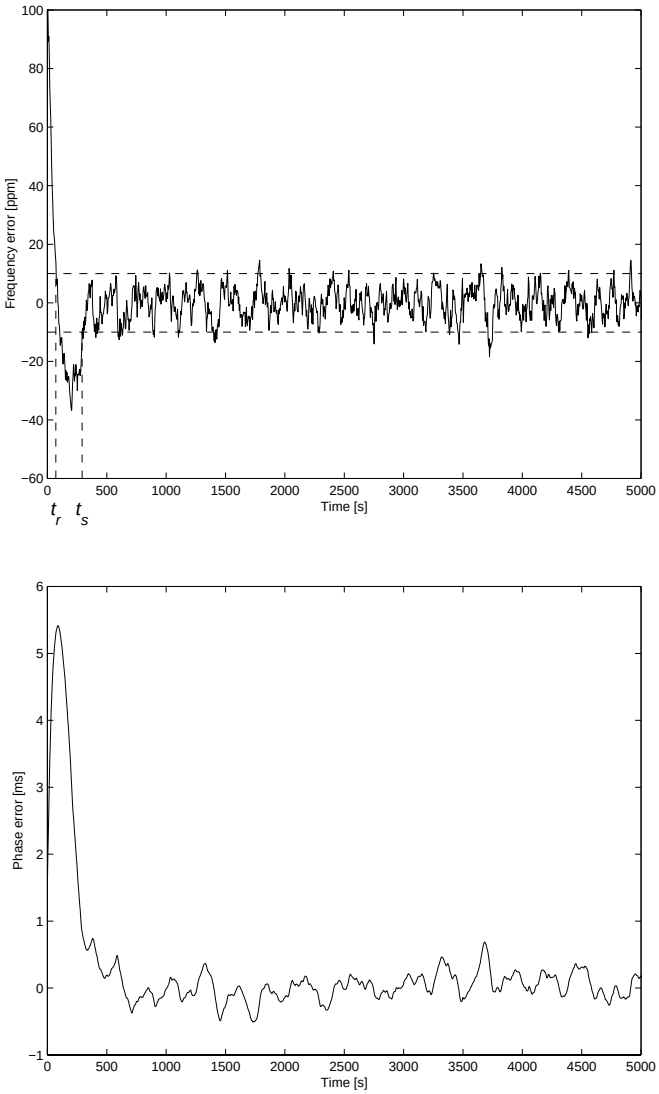

Figure B.19 Frequency and phase error of Filter no. 4.

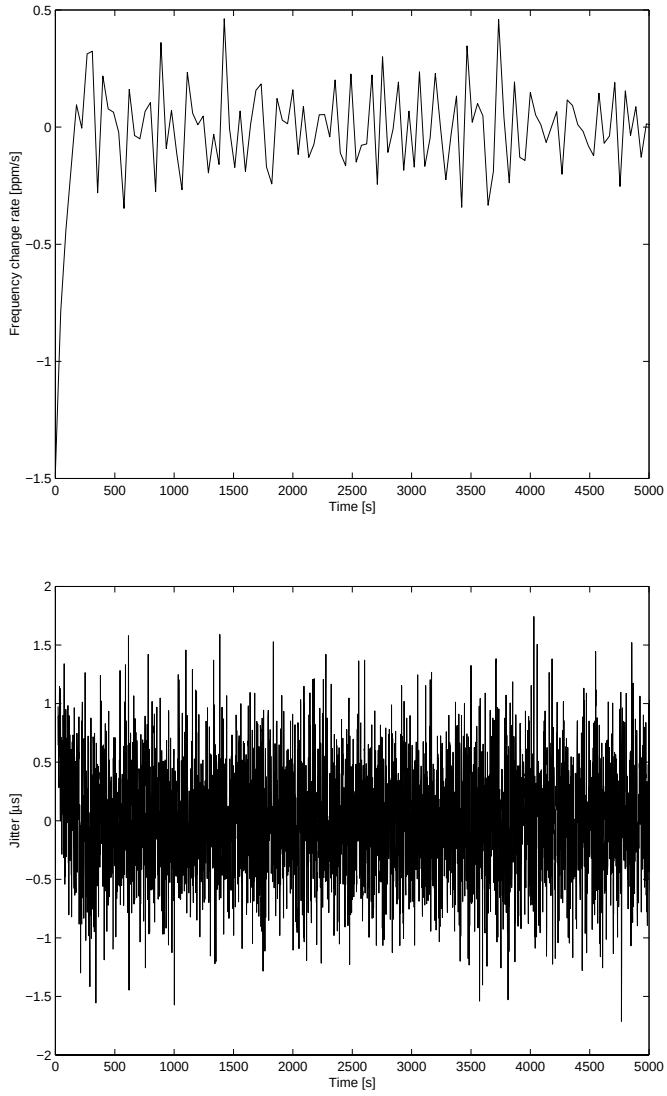

Figure B.20 Frequency change rate and jitter of Filter no. 4.

B.2.3 Results

Filter no.	Rise time [s]	Settling time [s]	Jitter [µs]
1	141	529	0.99
2	86	307	1.96
3	131	539	1.73
4	71	292	3.45

Table B.4 Results from simulations with filters with integral compensation