Synchronization of POTS Systems
Connected over Ethernet

Master’s thesis in Data Transmission
at Linköping Institute of Technology
by

Jonatan Lindblad

Reg nr: LiTH-ISY-EX--05/3666--SE
Linköping 2005
Synchronization of POTS Systems Connected over Ethernet

Master’s thesis in Data Transmission at Linköping Institute of Technology
by

Jonatan Lindblad

Reg nr: LiTH-ISY-EX--05/3666--SE

Supervisor: Tore André
Ericsson AB

Examiner: Mikael Olofsson
LiTH

POTS (Plain Old Telephony Service) systems have traditionally been connected via synchronous connections. When installing new nodes in the telephone network, they may sometimes be connected via packet networks such as Ethernet. Ethernet is an asynchronous network which means that nodes connected to the network don’t have access to the same clock frequency if it is not provided in some other way. If two nodes have different clock frequency, the receiver’s buffer will eventually overflow or starve. While not being a severe problem for telephony, devices used for data transmission, e.g. modems and fax will not be able to function properly. To avoid this it is necessary to synchronize the nodes.

This thesis investigates methods to synchronize nodes connected over Ethernet by simulating them in MATLAB. The simulations show that under certain circumstances it is possible to produce a clock signal conforming to relevant standards.
Abstract

POTS (Plain Old Telephony Service) systems have traditionally been connected via synchronous connections. When installing new nodes in the telephone network, they may sometimes be connected via packet networks such as Ethernet. Ethernet is an asynchronous network which means that nodes connected to the network don’t have access to the same clock frequency if it is not provided in some other way. If two nodes have different clock frequency, the receiver’s buffer will eventually overflow or starve. While not being a severe problem for telephony, devices used for data transmission, e.g. modems and fax will not be able to function properly. To avoid this it is necessary to synchronize the nodes.

This thesis investigates methods to synchronize nodes connected over Ethernet by simulating them in MATLAB. The simulations show that under certain circumstances it is possible to produce a clock signal conforming to relevant standards.

Keywords: Synchronization, phase-locked loops, clock recovery, telephony over IP
Acknowledgment

This thesis could not have been done without the help of a number of people. I would like to express my gratitude to Tore André and Henrik Almedia for the opportunity of doing my thesis at the ASP lab at Ericsson, Älvsjö. I would also like to thank the entire ASP team for a nice time. Great thanks to Mikael Olofsson for valuable comments on the report.

Stockholm, April 2005
Jonatan Lindblad
## Contents

1 Introduction .................................................. 1  
1.1 Background ................................................. 1  
1.2 Purpose ....................................................... 2  
1.3 Limitations ................................................... 2  
1.4 Outline of the Report ....................................... 2  

2 Telecommunication Networks .................................. 3  
2.1 Circuit-switched Networks .................................. 3  
2.1.1 Public Switched Telephone Network ...................... 3  
2.2 Packet-switched Networks .................................. 4  
2.2.1 Ethernet Networks ....................................... 5  
2.2.2 Protocols .................................................. 6  
2.2.3 Delay Jitter .............................................. 7  

3 Telephony over Internet Protocol ............................. 9  
3.1 Overview ..................................................... 9  
3.1.1 Transport ................................................ 10  
3.1.2 Gateway .................................................. 10  
3.1.3 Call Signaling .......................................... 11  
3.2 Timing Distribution over Packet Networks .................. 11  
3.2.1 Sample Frequency Offset ................................. 12  
3.3 Quality of Service (QoS) .................................... 12  
3.3.1 Jitter Buffers ........................................... 12  
3.3.2 Packet Loss .............................................. 13  

4 Synchronization ................................................ 15  
4.1 Phase-locked Loops ......................................... 15  
4.1.1 Analog Phase-locked Loops ............................... 16  
4.1.2 Digital Phase-locked Loops .............................. 16  
4.2 Oscillators .................................................. 18  
4.2.1 Controlled Oscillators .................................. 18  
4.2.2 Temperature Drift ...................................... 18  
4.2.3 Jitter and Wander ...................................... 20
5 Methods Based on Jitter Buffer Level .............................................. 23
  5.1 Proportional and Differential Regulator .................................. 23
  5.1.1 Jitter Transfer ..................................................................... 24
  5.2 Time-averaging Estimation Algorithm ...................................... 25
  5.3 Kalman Filtering ..................................................................... 27

6 Methods Based on Packet Arrival Patterns ........................................ 29
  6.1 Arrival-time of Packets .............................................................. 29
     6.1.1 PLL Structure .................................................................... 29
     6.1.2 Jitter transfer .................................................................... 30
     6.1.3 The Minimum Delay Algorithm ........................................... 32
     6.1.4 Alternatives to the Minimum Delay Algorithm ....................... 33
  6.2 Timestamps method ................................................................... 35

7 Simulation Results ........................................................................ 37
  7.1 Results for the Jitter Buffer-based Methods ................................. 37
     7.1.1 Proportional and Differential Method .................................. 37
     7.1.2 Time-averaging Method ....................................................... 40
     7.1.3 Kalman-based Method ......................................................... 42
  7.2 Results for the Methods Based on Packet Arrival Patterns ............... 44
     7.2.1 Arrival-time of Packets Method .......................................... 44
     7.2.2 Timestamps Method ............................................................ 51
  7.3 Summary ................................................................................... 53

8 Summary ........................................................................................ 55
  8.1 Further Work ........................................................................... 56

Bibliography ..................................................................................... 57

A Notation .......................................................................................... 61
List of Figures

2.1 Example of packet delay distribution ........................................ 8
3.1 Synchronous operation of ToIP ................................................. 11
3.2 Adaptive operation of ToIP ..................................................... 12
4.1 A generic PLL ................................................................. 16
4.2 Frequency multiplier/divider PLL .......................................... 17
4.3 Nonreturn to zero (NRZ) and Manchester binary coding format .... 18
4.4 Block diagram of a digitally controlled oscillator .................... 18
4.5 Frequency error vs. temperature and supply voltage for an oscillator 19
4.6 Clock jitter ................................................................. 20
5.1 Block diagram of the first buffer-based control algorithm .......... 24
6.1 PLL based on arrival time .................................................... 30
6.2 Loop filter used in the PLL based on arrival time .................... 31
6.3 Alternative model of the PLL .............................................. 31
6.4 PLL based on timestamps times ........................................... 35
6.5 The double exponential weighting moving average filter ........... 35
7.1 Frequency error for the PD method ....................................... 38
7.2 Time error results for the PD method .................................... 38
7.3 MRTIE results for the PD method ........................................ 39
7.4 Jitter amplitude for the PD method ....................................... 39
7.5 Frequency error for the time-averaging method ....................... 40
7.6 Time error results for the time-averaging method .................... 40
7.7 MRTIE results for the time-averaging method ......................... 41
7.8 Jitter amplitude for the time-averaging method ....................... 41
7.9 Frequency error for the Kalman method ................................ 42
7.10 Time error results for the Kalman method ............................ 42
7.11 MRTIE results for the Kalman method ................................ 43
7.12 Jitter amplitude for the Kalman method ................................ 43
7.13 Frequency error for the packet arrival method ....................... 44
7.14 Time error results for the packet arrival method .................... 45
7.15 MRTIE results for the packet arrival method ......................... 45
7.16 Jitter amplitude for the packet arrival method ....................... 46
7.17 Frequency error and solution time vs. window size and a .......... 48
7.18 Frequency error and solution time vs. window size and a .......... 49
7.19 Frequency error and solution time vs. window size and a .......... 50
7.20 Frequency error for the timestamps method ......................... 51
7.21 Time error results for the timestamps method ....................... 51
7.22 MRTIE results for the timestamps method ............................ 52
7.23 Jitter amplitude for the timestamps method .......................... 52
List of Tables

4.1 Jitter and wander tolerance requirements ............................................. 21
4.2 2048 kbit/s interface output wander limit ............................................. 22
7.1 Simulation results for the minimum average algorithm ......................... 47
Chapter 1

Introduction

1.1 Background

The telephone network has undergone large changes since it was first introduced at the end of the 19th century. From being a manually operated analog network it has evolved into a computer-controlled digital ditto. The latest development of telephony is to transport voice calls over a packet-switched network as opposed to the standard circuit-switched network. This has become possible with the advances in digital signal processing, high-speed packet networks and cheap hardware. To design a packet-based replacement of the traditional telephone network is however not trivial.

Several potential problems must be handled when developing such a system for Telephony over Internet Protocol (ToIP). To begin with, data transported over a packet network may experience large delays. This will cause conversations to be non-interactive in nature, and also increase the impact of talker echo. Another issue is packet loss where a packet is dropped somewhere on the route to the destination. This causes audible clicks for voice calls and data loss for modem/fax calls. A third issue, the topic of this thesis, is the problem with clock frequency offset between the sender and receiver.

In a traditional telephone system the nodes are using the same reference clock. This means that data is received and transmitted at the same rate. However, when the nodes are connected over a packet network like Ethernet, there is no common clock available.

The receiver incorporates a jitter buffer in which incoming data is placed. Its purpose is to remove the jitter that packets exhibits in the form of packet delay variation. As there is no common clock available at both sender and receiver, sooner or later the buffer will under- or overflow due to the frequency mismatch between sender and receiver.

Using high-accuracy rubidium clocks or Global Positioning System (GPS) clocks at each node could solve this problem. This solution has several drawbacks. Both rubidium and GPS clocks are relatively expensive and the GPS clock is not suitable
for indoor operation. An alternative way of synchronizing the nodes could be to recover the clock from the incoming data rate. However this is not that simple since data transmitted over a packet network might exhibit large variations in transfer time.

1.2 Purpose

The purpose of this thesis is to look closer into the problem of synchronizing the receiver’s clock frequency to that of the sender. A number of possible control algorithms will be described and simulated to see if they are capable of producing a clock signal that is of such quality that it conforms to the applicable standards. The algorithms will be simulated in the MATLAB environment.

1.3 Limitations

The following points limits the scope of this thesis:

- Jitter from the oscillator itself will not be regarded in the simulations.
- Packet loss will not be regarded in the simulations.

These two limitations could be added to the simulations but measurements on actual hardware should be made to fully evaluate the performance of the investigated methods.

1.4 Outline of the Report

Chapter 2 describes the basics of telecommunication networks – circuit-switched as well as packet-switched – with focus on the public switched telephone network (circuit-switched) and on Ethernet (packet-switched).

Chapter 3 describes the basics of using a packet-switched network for transporting telephone services. Voice transport, gateways, and quality issues are some of the treated topics.

Chapter 4 describes the basics of synchronization and synchronization circuits and some of its applications. Some details on measurements of clock quality are also presented.

Chapters 5-6 describe the investigated algorithms used for frequency synchronization. Chapter 5 contains three algorithms that use the buffer fill level to control the receiver’s frequency, while the two methods in chapter 6 use the arrival patterns of packets to control the receiver’s frequency.

Chapter 7 contains the simulation results of the algorithms. Results are presented as figures of for example frequency error, along with explanations of the results.

Chapter 8 concludes the thesis with a summary and comments of the results, and suggestions to further work.
This chapter describes the background and basics of two types of telecommunication networks: circuit-switched, and packet-switched. The circuit-switched networks are suited for voice services, while the packet-switched networks are designed for data traffic.

2.1 Circuit-switched Networks

A circuit-switched (telephone) network is a network where sender and receiver are connected end-to-end by circuit connections. In the beginning of telephony, these connections were all analog, but have gradually evolved into a digital synchronous network.

2.1.1 Public Switched Telephone Network

The Public Switched Telephone Network (PSTN) is a circuit-switched network in which each circuit is a 64 kbit/s digital channel. These channels are then multiplexed by time division multiplexing (TDM) to form a signal of higher bitrate. In Europe, the multiplexed signal is called an E1 signal, and consists of 30 channels for data, plus one channel each for synchronization and signaling, giving a total bit rate of 2048 kbit/s.

The most common use of PSTN is the Plain Old Telephone Service (POTS). POTS provides regular telephony in which voice is limited to frequencies in the 300 to 3400 Hz passband. The customer premises equipment is analog and consists of devices such as telephones, fax machines, and modems. These devices are connected via the local loop or subscriber line to the local exchange in which the signal is digitized at a sample frequency of 8 kHz and quantized to 8 bits, thus yielding a 64 kbit/s signal.
Another possibility is to digitize the speech directly at the customer premises rather than at the local exchange. Two examples of this is the Integrated Services Digital Network (ISDN), which achieves a maximum data rate of 1.920 Mbit/s [1], and Voice over Digital Subscriber Line (VoDSL).

### Signaling

In the beginning of telephony, signaling was dealt with manually. When the subscriber wanted to make a call, he activated his hand-cranked magneto telephone to generate a ringing signal at the telephonist in the exchange. After the telephonist recognized the call, the subscriber and telephonist exchanged the telephone number vocally, after which the telephonist manually connected the subscriber to the other end.

Nowadays the magneto telephones have been replaced with telephones equipped with a hook switch. If the subscriber picks up the telephone the hook switch connects the keypad (modern replacer of the dial) and audio circuit, while disconnecting the bell. This notifies the local exchange that the subscriber is going to dial a number.

Signaling can be separated in two categories:

- *in-band* signaling
- *out-of-band* signaling.

The first category uses signals that are in the frequency band used for voice. The second category uses signals that are in a separate frequency band from voice, but are still carried on the same channel. Out-of-band signaling can be further separated from the voice channel by using a dedicated channel for signaling.

The Signaling System 7 handles the signaling in the PSTN. It is a network which uses the latter form of out-of-band signaling, i.e. it is physically separated from the voice network. This separation minimizes the risk of hacking and fraud which older in-band signaling systems were exposed to [1][2]. The Signaling System 7 is responsible for the set up, tear down and timing of calls for billing. It also provides services such as caller ID and number portability.

### Synchronization

When the telephone network was still analog, synchronization wasn’t needed. However, when circuit-switched data networks were introduced they required more strict synchronization. The method of achieving good synchronization is nowadays to use a synchronization network where timing is distributed.

#### 2.2 Packet-switched Networks

Packet-switched networks were designed to carry data traffic and to make it possible for the user to be connected to the network but only use bandwidth during the
actual transfer. Connections are not end-to-end as for circuit-switched networks. Instead, data can take different routes toward the receiver. Packets can also be dropped.

### 2.2.1 Ethernet Networks

The most commonly used type of packet-switched network is the Ethernet type of network. It was developed during the 70’s by Digital Equipment Corporation, Intel, and Xerox. In 1984 the IEEE 802.3 \[3, 4\] standard for 10 Mbit/s Ethernet was released. Later, the standard has evolved to include speeds of 100 and 1000 Mbit/s.

The various kinds of Ethernet networks use different transmission schemes. For example 10BASE-T uses the Manchester coding described in subsection 4.1.2, 100BASE-T4 uses ternary encoding, while 100BASE-T2 and 1000BASE-T uses quinary encoding \[3\]. Ethernet uses *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD) to retransmit packets whenever a collision occurs. Ethernet can be transported over several types of media including optical fiber, coaxial cable and twisted copper pair.

**The Ethernet Frame**

The basic unit of transport over the network is the Ethernet frame. It consists of the following fields:

- eight-byte preamble
- six-byte destination address field
- six-byte source address field
- two-byte type field
- data field containing a maximum of 1500 bytes
- four-byte checksum.

Ethernet is an asynchronous network which means that there is no common clock available at the nodes. This is the reason that there is a preamble. The preamble contains a sequence of alternating 1s and 0s creating a 5 MHz signal. This signal allows the receiver to adjust its clock frequency so that a correct interpretation of the received data is done. To verify that the received data is valid, a checksum is computed and compared to the received checksum. If the comparison fails, the packet is dropped. The packet is also dropped if its length is not an integer multiple of bytes.

The data field contains data formatted according to a protocol which will be briefly described in the next section.
2.2.2 Protocols

Data transferred over a packet network is formatted in accordance with a protocol. The protocol can specify how long the data is, what type of data is in the packet, and other information of how to handle the data. This information is generally stored in the protocol’s header.

Internet Protocol (IP)

The Internet Protocol is a network layer protocol. The 20 byte header contains fields such as type of service (ToS) which specify how the packet should be handled by the network, and a length field containing the length of the packet.

User Datagram Protocol (UDP)

The User Datagram Protocol (UDP) is a so-called connectionless transport layer protocol. The term connectionless means that delivery is not guaranteed and that the sender will not now if a packet has reached the receiver. It runs on top of IP.

Transmission Control Protocol (TCP)

TCP is the standard protocol used on the Internet. Two endpoints in the network can be virtually connected where each endpoint is defined as an IP address. TCP is the second of two transport layer protocols. As opposed to UDP, TCP is a connection oriented transport layer control. Connection oriented roughly means that packets are retransmitted if they are lost on the way to the receiver. It runs on top of IP and is the standard protocol used for non-realtime data transfers.

Real-time Transport Protocol (RTP)

The real-time transport protocol (RTP) is a protocol for time-sensitive applications such as real-time voice and video. The protocol itself does not provide any guarantees of timely deliverance of packets. RTP is usually run on top of UDP which means that a lost packet is never retransmitted. A retransmitted packet is likely to arrive too late to be of any use in the receiver.

The RTP header contains information which is useful for real-time applications. The sequence number makes it possible to detect out-of-order and lost packets. The timestamp contains a monotonically increasing number which can be used to play out the stream at a correct speed.

The 12 byte RTP header does not contain any information of the length of the actual data so this is handled by the lower protocols.

RTP Control Protocol (RTCP)

The primary function that RTCP provides is monitoring of quality of service (QoS) for RTP traffic. QoS is further discussed in section 3.3 on page 12. Within each RTCP packet a sender and/or receiver report contains statistics such as number of
packets sent and lost, and inter-arrival jitter. This information can for example be
used by the sender to alter its transmission settings.

RTCP packets are sent periodically and are limited to an amount of 5% of
the total traffic. This allows a great number of participants and limits the use of
network resources.

2.2.3 Delay Jitter

An Ethernet network is a best-effort network, i.e. data is transported as fast as
possible, but no guarantees can be made of how long it takes for the data to arrive
at the receiver. Hence, the transmission time of packets may vary. Delay jitter or
packet delay variation (PDV) is the variation in transmission delay that packets
exhibit. It arises when packets travel on paths of different length and in switches
and routers due to queuing.

By setting the type of service flag in the IP header, packets can receive priorit-
ization in the routers and switches in the network. If a prioritized packet arrives
at a switch where a number of other packets are in queue to be transmitted, the
arriving packet can jump ahead in the queue. However, if another packet is being
transmitted from the switch, the arriving packet must wait until this transmission
has ended.

The distribution of packet delays is not a unique one. It depends on the type
of network, e.g. 10 Mbit/s or 100 Mbit/s, packet size, number of switches, and also
on the current congestion. An example of packet delay distribution is shown in
Figure 2.1. As shown, a large number of packets arrive with low delay. The shape
of the delay distribution can be explained as follows [5]: Packets arriving when no
other packets are currently transmitted from the switch are subject to low delay,
thus the large peak. The next level corresponds to prioritized packets having
to wait for low-priority packets currently being transmitted. The second level is
caused by prioritized packets having to wait for other prioritized packets currently
being transmitted.
Figure 2.1. Example of packet delay distribution.
Chapter 3

Telephony over Internet Protocol

Traditionally, telephony has been carried over the Public Switched Telephone Network (PSTN). This circuit-switched network was initially used only for telephony but has evolved to carry data traffic as well.

With the recent advances in high-speed data networks like Ethernet, and digital signal processor technology, the possibility to run voice, video and data in one converged network has appeared. This has the benefits that operators need only a single network for both data and voice, and that they also get simpler and more efficient management. The investments for a packet-based solution are also less than for a circuit-switched equivalent.

However, there are some possible drawbacks. One of them is the transmission delay that packets experience when transported over the network. This can cause a conversation to be non-interactive and echo can disturb the talkers. Another problem is delay jitter which arises in the queues in switches and routers. A third drawback is that an IP telephone needs a separate power supply whereas a regular telephone is powered directly from the telephone line.

Various standards regarding Telephony over Internet Protocol (ToIP) are being developed. The Internet Engineering Task Force (IETF) and the Metro Ethernet Forum are working on standardizing transportation e.g. E1 circuits over Ethernet [6, 7]. The ITU-T Study Group 15 is working on their G.pactiming Recommendation named ”Timing and synchronization aspects of packet networks” [5].

3.1 Overview

Depending on the network architecture, telephones may be connected at different points in the network. IP telephones might be connected directly to the packet-switched network enabling users to talk with each other exclusively over the packet-switched network. Telephones might also be connected to the traditional PSTN,
making it necessary to convert audio and signaling to transport it over a packet-switched network.

The ITU-T Recommendation H.323\(^8\) is a family of protocols for interworking PSTN with packet networks as well as providing interoperability between different ToIP products. H.323 also covers transport of video and data, for example video conferences and modem traffic. An H.323 network has four types of devices:

- **Terminal**: An H.323 terminal is an endpoint in the network, for example an IP telephone.
- **Gateway, or Interworking Function (IWF)**: An H.323 gateway is an endpoint in the network that connects the packet-based network with the PSTN.
- **Gatekeeper**: The gatekeeper is an optional part which, among other things, controls access to the network and routes calls when the network has multiple gateways.
- **Multipoint control unit**: Provides for multiple terminals to participate in conferences and other sessions where multiple simultaneous terminals are involved.

### 3.1.1 Transport

When transporting voice over a packet network, it is broken down into packets containing a small amount of data. The packets usually contain 5 to 20 ms of speech. Larger packets will increase the packetization and transmission delay. Another consequence of large packets is that the jitter buffer must be larger to accommodate the larger packets, which in turn increases the total delay. In case of packet loss, large packets will cause long gaps in the received audio stream, reducing the perceived quality.

A small packet size is thus desirable. However, choosing a smaller packet size increases the overhead. The combined header size of IP, UDP and RTP packets is 40 bytes. Therefore, the overhead is 20% for a 20 ms packet. Obviously, there is a tradeoff between delay and bandwidth usage.

### 3.1.2 Gateway

The gateway is responsible for the conversion of TDM signals to/from packets, and also handles signaling used for e.g. call setup and tear-down. For calls from the PSTN, the signal is digitized, coded and placed into packets. The coding supported by the gateway must at least include the 64 kbit/s G.711 PCM codec but can include more advanced codecs like G.729 for an 8 kbit/s stream\(^9,10\). In the opposite direction the gateway reads packets from the jitter buffer, decodes the data and performs a digital to analog conversion.
3.1.3 Call Signaling

As opposed to the PSTN, the H.323 protocol does not use a separate network for signaling. Call signaling is instead transported over the same network as used for call transport. The call signaling is handled with the ITU-T Recommendation H.225 [11] protocol which provides a number of messages which are used to setup calls. If the packet network is connected to the PSTN, then a signaling conversion takes place in the gateway.

3.2 Timing Distribution over Packet Networks

In a regular PSTN network, a timing distribution network provides synchronization that is traceable to a primary reference clock (PRC). This clock is a high-accuracy clock with an error smaller than $10^{-5}$ ppm [12].

As discussed in [5] there are two classes of methods to provide synchronization. These are

- Network synchronous methods
- Packet based methods.

Timing distribution methods belonging to the first class are used in PSTN. These methods are PRC distributed method and master-slave method. The PRC distributed method uses a clock that is available at all the nodes, and could for example be based on GPS.

In the master-slave method, timing is distributed to synchronization nodes through a synchronous physical layer. As Ethernet does not provide a synchronous layer, the master-slave method is not an option.

The scenario of a distributed PRC is shown in Figure 3.1.

![Figure 3.1. Synchronous operation of ToIP.](image)

Timing can also be achieved by using an adaptive timing recovery in the receiver. Adaptive timing recovery can – as described in later chapters – be based on either fill level of the jitter buffer or the inter- or arrival-time of packets. The adaptive timing scenario is shown in Figure 3.2.
3.2.1 Sample Frequency Offset

In the case of adaptive timing recovery it is likely that the source and receiving sample clocks doesn’t match. This error in frequency can be described in a relative frequency offset. Denoting source and receiver sample clocks with $f_S$ and $f_R$ respectively, the frequency offset relative to the source is specified as

$$\frac{\Delta f}{f_S} = \frac{f_R - f_S}{f_S}$$  \hspace{1cm} (3.1)$$

and is usually measured in parts per million, ppm.

The frequency offset determines the *slip rate*, which is how many samples per time unit are deleted or inserted by the receiver to accommodate the frequency offset. An offset of 100 ppm corresponds to a slip rate of 0.8 slips per second or one slip every 1.25 seconds. The frequency offset also determines how long it takes until the jitter buffer is emptied or filled. With an offset of 1 ppm and a 20 ms jitter buffer it takes about 333 minutes until the buffer empties.

3.3 Quality of Service (QoS)

QoS is the ability of the network to provide reliable services in terms of e.g. packet delivery and sound quality. To achieve QoS various methods can be applied. Prioritization of packets flagged with a certain type of service allows voice packets to jump ahead packets already in queue.

3.3.1 Jitter Buffers

As described earlier, the purpose of the jitter buffer is to remove the delay variation caused by the asynchronous nature of Ethernet networks. An additional delay is introduced as the packets are stored in the buffer. For voice traffic, this delay should be kept as small as possible. This is because conversations turn to be more non-interactive with greater delay. If proper echo control is accommodated the upper limit of permissible delay is 150 ms, according to ITU-T Rec. G.114 [13].
3.3 Quality of Service (QoS)

Other contributors to delay are coder delay, packetization delay and transmission delay.

Jitter buffers can be designed in several ways. For example, the size of the buffer may be either fixed or adaptive, and of integer or fractional number of packets. Other parameters are given in ITU-T Rec. G.1020 [14].

Packets with delay greater than a certain maximum level are treated as lost packets. A larger jitter buffer will be able to handle more delay variation and thus reduce the amount of lost packets at the cost of increased delay. On the other hand, smaller buffers gives lower delay, but a higher number of packets are lost.

Using integer packet sized buffers allow packets to be reordered in case packets takes different routes and arrive out of order. This is not possible with buffers of fractional packet size, where out of order packets are discarded [14].

3.3.2 Packet Loss

Except those factors described in the previous section, the use of RTP to carry data over the network is another.

As opposed to the TCP, RTP doesn’t include re-transmission of lost packets. This is because it is run on top of UDP, which is a connectionless protocol. If instead RTP were to be run on top of TCP, which is a connection-oriented protocol, the receiver is forced to wait during re-transmission of lost packets. Several other drawbacks of using TCP for real-time data are listed in [15].

Packet Loss Concealment

To attenuate the effect of packet loss, packet loss concealment (PLC) algorithms can be employed. PLC is mostly useful only when a small number of consecutive packets are lost. Several model-based codecs, such as ITU-T Rec. G.729 [10], includes PLC in their implementation but for waveform codecs like ITU-T Rec. G.711 [9], it is optional.

For model-based coders like G.729, sound quality gradually decreases with increasing amount of packet loss. At the cease of packet loss, sound quality begins to return to normal as packets arrive. This is different from waveform codecs where quality is restored immediately after packet loss.

Due to speech often being locally stationary, common techniques for PLC are based on using previously received speech to generate a synthesized signal during the loss. Simple techniques like silence insertion and packet repetition are easily implemented and have low computational complexity, but have low sound quality. While more advanced techniques like pitch period repetition and time-scaling gives better quality [16], they also require more processing power.

As PLC is aimed toward voice traffic, for services like fax and modem traffic it might not be beneficial.

Depending on the PLC algorithm, an additional delay might be added. The algorithm described in [17] for G.711 adds a delay of 3.75 ms for providing a smooth transition between the synthetic signal and the next correctly received signal.
Chapter 4

Synchronization

Synchronization, the act of aligning two or more events, is a necessary thing in many cases. For example, the frequency of a radio has to be tuned to be able to receive a radio broadcast; the same goes for television. Another area where synchronization is necessary is in data transmission. Without synchronization, data transmission would be error-prone if at all possible. This chapter gives a brief overview of synchronization circuits, oscillators, and measurements of clock quality.

4.1 Phase-locked Loops

The phase-locked loop (PLL) is the central unit in most synchronizers. The PLL is responsible for driving the local oscillator’s output to be in-phase with the input. This kind of synchronization is called carrier phase synchronization, and is essential for data transmission systems that use the signal’s phase as a means of transferring information. Examples of this type of modulation are binary phase shift keying and quadrature phase shift keying, whose details will not be treated in this report.

The generic PLL consists of a phase detector (PD), a loop filter, and a voltage-controlled oscillator (VCO). The phase detector compares the phase of the input signal, the reference signal, to that of the signal from the VCO. This difference, the error signal, is fed to the loop filter, whose purpose is to attenuate noise that is present on the input signal. The loop filter outputs a signal that is proportional to the phase error and drives the VCO output in a direction to minimize the phase error.

Figure 4.1(a) shows the generic structure of the PLL. In Figure 4.1(b), a PLL that locks to a continuous sinusoid, where the phase detector is a multiplier/low-pass filter (LPF). The output from the multiplier will contain one sinusoid with a frequency that is the sum of the two input signals’ frequencies, and one sinusoid with a frequency that is the difference between the two signals’ frequencies. By filtering out the first component, the only part left will be a function of the
frequency difference, and will thus drive VCO’s output to be in phase with the reference signal.

All PLLs are exposed to noise – both noise in the input signal and also intrinsic noise from the oscillator. This requires that the bandwidth of the PLL is narrow in order to reject as much noise as possible. However, a narrower bandwidth causes the PLL to act slower, and thus the time to drive the phase error to zero will increase. Hence, there is a trade-off between pull-in time and noise rejection.

Figure 4.1. In (a) a generic PLL, and in (b) a PLL with sinusoids and multiplier/LPF phase detector (notation taken from [18]).

### 4.1.1 Analog Phase-locked Loops

Analog PLLs were the first type of PLLs. In these, the loop filter is built from analog components such as capacitors and coils. Typical applications involve high-speed communication systems such as satellite communication [18, 19], but it was with television the PLL became popular [19].

### 4.1.2 Digital Phase-locked Loops

A digital PLL often has the same structure as an analog PLL. However, the digital PLL is a discrete-time system while the analog is a continuous-time system. The digital PLL also differs from the analog PLL in that the loop filter is built from multiplication, delay and adder elements.

**Frequency Synthesis**

An important property of digital PLLs is their ability to generate, or synthesize, a frequency that is a multiple of the reference frequency. This is achieved by inserting a divide-by-N counter between the VCO and the phase detector. The result is that the VCO produces a frequency which is the reference frequency multiplied by \( N \). In a similar way, the PLL can generate a frequency that is the reference
frequency divided by \( N \), if the counter is inserted between the reference signal and the phase detector. These two PLL structures are shown in Figure 4.2(a) and 4.2(b) respectively.

\[\text{Input} \xrightarrow{\text{PD}} \div N \xrightarrow{\text{Loop filter}} \text{VCO} \xrightarrow{\text{Output}} (a)\]

\[\text{Input} \xrightarrow{\div N} \xrightarrow{\text{PD}} \xrightarrow{\text{Loop filter}} \text{VCO} \xrightarrow{\text{Output}} (b)\]

Figure 4.2. In (a) a frequency multiplier PLL, and in (b) a frequency divider PLL.

Clock Recovery

When digital data is transmitted over a channel, it is sent in synchronism with a clock signal. The clock signal is however usually not transmitted. Hence, the receiver must recover the clock signal in some way to be able to correctly decode the received data.

The description of clock recovery in this section deals with data transmitted as a baseband signal. The term baseband means that the digital signal is sent directly over the channel, i.e. the signal’s frequency band is not shifted to a different frequency as in carrier-based transmission.

Baseband signals can be coded in different ways, that is, the pulses that represent 1 or 0 can be sent in various ways to either make synchronization easy or limit the bandwidth usage. Two common codes are the nonreturn to zero (NRZ) code and the Manchester code. The former is the simplest form of code since a 1 is sent as a positive pulse and a 0 is not sent at all. The Manchester code on the other hand sends a 1 as ”high-low” and a 0 as ”low-high”. For an example of these two codes, see Figure 4.3 where the sequence 100101 is coded. With Manchester or biphase [19] coding, clock recovery is particularly easy since there is a level transition in the middle of each interval. Compared to NRZ though, the Manchester coding has the disadvantage that it requires twice the amount of bandwidth for the same bit rate. The NRZ format on the other hand suffers from synchronization problems when the transmitted data consists of a long sequence of 1s or 0s. For example, if a sequence of 500 consecutive 0s were transmitted, and the receiving end’s clock drift a small amount, then the receiver will be unable to determine if 499, 500, or 501 0s were transmitted.
4.2 Oscillators

There are several types of oscillators and with various accuracies. The most accurate is the cesium beam atomic clock which has a short-term stability ranging from less than $10^{-13}$ to $3 \cdot 10^{-12}$. This type of clock is often used as a primary reference clock (PRC) in the synchronization networks used in public switched telephone networks (PSTN).

An oscillator with less stability is the rubidium atomic oscillator, which has a short-term stability that ranges from $5 \cdot 10^{-12}$ to $5 \cdot 10^{-11}$. Further down the line of oscillators comes the quartz crystal oscillators (XO), which have a stability in the order of $10^{-6}$.

4.2.1 Controlled Oscillators

In contrast to oscillators with fixed frequency, controllable oscillators output a frequency that is proportional to an applied input signal. The input can for example be a voltage, as used by voltage-controlled oscillators (VCO), or a digital input, as used by digitally controlled oscillators (DCO). The DCO is used in all-digital PLLs and may for example be implemented as the $\div N$ counter DCO shown in Figure 4.4.

4.2.2 Temperature Drift

Because of temperature changes, the frequency of the local oscillator will drift. One way of overcoming this problem is to use an oven-controlled crystal oscillator where the oscillator is placed in an oven with a temperature well above the surrounding.
An alternative method is to use an ordinary voltage-controlled crystal oscillator and measure the temperature (e.g. $25^\circ C$) and compensate for this.

Measurements were made to see how the frequency of an XO changed with temperature. This XO was rated to have an overall stability of $\pm 28$ ppm and temperature stability better than $\pm 2$ ppm/$^\circ C$. The results are shown in Figure 4.5(a). The XO’s sensitivity of supply voltage was also measured and is shown in Figure 4.5(b).

**Figure 4.5.** Frequency error as a function of (a) temperature measured at a supply voltage of 3.3 V, and (b) supply voltage measured at a temperature of 36$^\circ C$. 
4.2.3 Jitter and Wander

Clock jitter is the variation in the arrival of clock edges from their ideal position in time. The term jitter is used whenever the frequency of the jitter is higher than or equal to 10 Hz. For variations with frequencies lower than 10 Hz, the term wander is used. An illustration of the effect of clock jitter is shown in Figure 4.6.

![Figure 4.6. Clock jitter. The dashed line represents the correct clock.](image)

The jitter and wander of the output clock used in a PSTN network based on the 2048 kbit/s hierarchy must conform to the ITU-T Rec. G.823 [20]. This recommendation specifies the limits of output jitter and jitter tolerance. These requirements ensure the interoperability of equipment from different manufacturers provided that the equipment fulfills the requirements. The limits of jitter are specified as amplitude as a function of frequency and the limits of wander are specified with the maximum relative time interval error (MRTIE).

The measurement of jitter can be divided in the following categories:

- **Output jitter** – The jitter present on an output. Usually expressed in peak-to-peak amplitude.
- **Jitter tolerance** – The amount of jitter a device can accommodate without producing bit errors.
- **Jitter transfer** – The amount of jitter transferred from the input to the output.

Jitter is usually measured in ns or µs, or *Unit Intervals* (UI). The unit interval is defined as the length of a bit clock period. Therefore, the unit interval is unrelated to the bit rate but instead a relative measurement. For the limits of jitter tolerance given in G.823, see Table 4.1.

**Maximum Relative Time Interval Error**

The MRTIE is a peak measure of the time difference of a clock compared to a reference clock during an observed interval. If the reference clock is a primary reference clock, then the word relative is dropped and the term maximum time interval error (MTIE) is used. In ITU-T Rec. G.823 [20] MTIE is used for measuring the wander of synchronization interfaces, while MRTIE is used for measuring the wander of traffic interfaces.

In order to define the MRTIE, it is first necessary to define the time function, \( T(t) \), of a clock. The time function of a clock is defined in terms of its total...
Table 4.1. Minimum requirements for 2048 kbit/s interface input jitter and wander tolerance.

<table>
<thead>
<tr>
<th>Frequency $f$ [Hz]</th>
<th>Requirement [pk-pk phase amplitude]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$12 , \mu &lt; f \leq 4.88 , m$</td>
<td>$18.0 , \mu s$</td>
</tr>
<tr>
<td>$4.88 , m &lt; f \leq 10 , m$</td>
<td>$0.088 , f^{-1} , \mu s$</td>
</tr>
<tr>
<td>$10 , m &lt; f \leq 1.67$</td>
<td>$8.8 , \mu s$</td>
</tr>
<tr>
<td>$1.67 &lt; f \leq 20$</td>
<td>$15.0 , f^{-1} , \mu s$</td>
</tr>
<tr>
<td>$20 &lt; f \leq 2.4 , k$</td>
<td>$1.5 , UI^a$</td>
</tr>
<tr>
<td>$2.4 , k &lt; f \leq 18 , k$</td>
<td>$3.6 \cdot 10^3 , f^{-1} , UI^a$</td>
</tr>
<tr>
<td>$18 , k &lt; f \leq 100 , k$</td>
<td>$0.2 , UI^a$</td>
</tr>
</tbody>
</table>

*1 UI = 488 ns

The phase of an actual clock signal can be modeled as a function of initial phase offset, frequency offset, frequency drift rate and random phase deviation [21]. For a clock with higher frequency than the reference, $(d/dt)T(t) > 1$ holds, and vice versa. Taking the difference between the time function of a clock and that of a reference clock, one gets the time error function, $x(t)$:

$$ x(t) = T(t) - T_{ref}(t) $$

where $T_{ref}(t)$ is the time function of the reference clock.

From the time error function it is possible to define the time interval error (TIE). This value is simply the change in the time error during an observation interval $\tau$ and is defined by

$$ TIE(t, \tau) = x(t + \tau) - x(t). $$

If the time error is measured from time $t_0$ to $t_1$ the MTIE is calculated by

$$ MTIE(\tau) = \max_{t_0 \leq t \leq t_1-\tau} \left\{ \max_{t \leq \xi \leq t+\tau} \{x(\xi)\} - \min_{t \leq \xi \leq t+\tau} \{x(\xi)\} \right\}. $$

The value $MTIE(\tau)$ is a maximum peak-to-peak change of $TIE(t)$ for all possible intervals of length $\tau$ inside the interval $t \in [t_0, t_1]$. However, it is not feasible to measure the continuous time error function, but instead a sampled version. Using a sampling interval $\tau_0$ the MTIE is estimated with the following equation:

$$ MTIE(n\tau_0) \approx \max_{1 \leq k \leq N-n} \left\{ \max_{k \leq i \leq k+n} \{x_i\} - \min_{k \leq i \leq k+n} \{x_i\} \right\}. $$
where $N$ is the number of samples, \( \{x_i = x(i\tau_0), i = 1, \ldots, N\} \) is $x(t)$ sampled, and $n\tau_0 = \tau$ is the observation interval with $n = 1, 2, \ldots, N - 1$.

The G.823 MRTIE requirements of the wander of a 2048 kbit/s traffic interface are given in Table 4.2.

**Table 4.2.** 2048 kbit/s interface output wander limit.

<table>
<thead>
<tr>
<th>Observation Interval $\tau$ [s]</th>
<th>MRTIE Requirement [$\mu$s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.05 &lt; \tau \leq 0.2$</td>
<td>$46\tau$</td>
</tr>
<tr>
<td>$0.2 &lt; \tau \leq 32$</td>
<td>$9$</td>
</tr>
<tr>
<td>$32 &lt; \tau \leq 64$</td>
<td>$0.28\tau$</td>
</tr>
<tr>
<td>$64 &lt; \tau \leq 1000$</td>
<td>$18$</td>
</tr>
</tbody>
</table>
Chapter 5

Methods Based on Jitter Buffer Level

The first three methods are all based on using the jitter buffer level as an indication of whether the clock frequency is higher or lower than the nominal value.

It is here assumed that data is inserted to the buffer in packets and read out in samples. This causes the fill level to look like a sawtooth when plotted as a function of time.

The three methods described in this chapter all make use of the change in buffer level to control the clock frequency. While all three methods perform filtering of the buffer level, it is only the first that performs a selection of which packets that should be used for frequency control.

5.1 Proportional and Differential Regulator

Although this control algorithm has been described for use in an ATM network, it is applicable to similar packet-switched networks [22].

The fill level, denoted \( \phi[i] \), is sampled at every packet arrival. For each \( M \)th packet the maximum value, \( \phi_{max}[j] = \max \{ \phi[(j-1)M-1], \ldots, \phi[jM] \} \), of the previous \( M \) samples is determined. The change in this value indicates how wrong the local frequency is. Thus, the difference \( \phi_{max}[j] - \phi_{max}[j-1] \) is used to control the frequency.

However, if the frequency is adjusted based only on the difference in maximum fill level, the fill level might settle down at a position near the bottom or the top of the buffer, making it vulnerable to over- or underflow. The optimal position, regarding likeliness of over- and underflow, is in the middle of the buffer, here denoted \( \phi_{opt} \). Thus, the frequency is adjusted based on \( \phi_{max}[j] - \phi_{opt} \) as well as \( \phi_{max}[j] - \phi_{max}[j-1] \). The resulting control equation is

\[
 f_R[j] = f_R[j-1] + \alpha(\phi_{max}[j] - \phi_{max}[j-1]) + \beta(\phi_{max}[j] - \phi_{opt}), \quad (5.1)
\]
where \( \alpha, \beta > 0 \) are constants affecting the rate of change in the receiver’s sample frequency \( f_R \).

The change in \( \phi_{\text{max}}[j] \) between two consecutive blocks primarily depends on the delay jitter, and to a smaller degree on the frequency error. Over a longer period of time, the change in \( \phi_{\text{max}}[j] \) primarily depends on the frequency. This means that the change in \( \phi_{\text{max}}[j] \) between blocks separated more than one block would give a better estimation of the frequency error. Using this approach the modified control equation then becomes

\[
f_R[j] = f_R[j - 1] + \alpha (\phi_{\text{max}}[j] - \phi_{\text{max}}[j - J]) + \beta (\phi_{\text{max}}[j] - \phi_{\text{opt}}),
\]

(5.2)

where \( J \geq 1 \) denotes the block separation. The corresponding block diagram is shown in Figure 5.1.

![Figure 5.1. Block diagram of the first buffer-based control algorithm.](image)

### 5.1.1 Jitter Transfer

In order to derive the jitter transfer function of the system, it is first necessary to observe that in absence of jitter, the maximum buffer fill level satisfies the following equation

\[
\phi_{\text{max}}[j] = \phi_{\text{max}}[j - 1] + T(f_S[j - 1] - f_R[j - 1])
\]

(5.3)
5.2 Time-averaging Estimation Algorithm

where $T$ is the time between sampling instants $j$ and $j-1$, and $f_{S}[j-1]$ is the source clock frequency. To see this, first assume that that the sender and transmitter are operating at the same clock frequency. Then $\phi_{max}$ will be constant as the number of samples played out in the time between instants $j$ and $j-1$ is equal to the number of samples in the arriving packet at instant $j$. Now if the sender’s clock frequency is lower (higher) than the receiver’s, $\phi_{max}$ will decrease (increase).

By taking the z-transform of (5.3) and combining it with the z-transform of (5.2), it is found that the transfer function from $f_{S}$ to $f_{R}$ is

$$H(z) = \frac{T(\alpha + \beta)z^{-1} - T\alpha z^{-1-j}}{1 - (2 - T(\alpha + \beta))z^{-1} + z^{-2} - T\alpha z^{-1-j}}.$$  (5.4)

5.2 Time-averaging Estimation Algorithm

This buffer-based algorithm [23] uses the time-average of the buffer fill level as an indication of whether the local clock frequency is higher or lower than the clock at the sending node.

The algorithm doesn’t ensure that the fill level approaches an optimal point. Instead, the algorithm is reset when the fill level reaches a point too close to the bottom or the top of the buffer. A possible effect of this could be that the output frequency exhibits sudden changes when the algorithm is reset. This means that the requirements from G.823 might not be fulfilled.

By computing an estimated frequency error $\hat{\Delta}f$, the frequency can be controlled by adding a weighted value of $\hat{\Delta}f$ to $f_R$, i.e.

$$f_{R}[m] = f_{R}[m - 1] + \alpha \hat{\Delta}f[m - 1],$$  (5.5)

where $0 < \alpha < 1$.

Two indices will be used in the following. First, as in the above equation, the index $m = 0, 1, \ldots$ refers to the instants where the frequency is updated. Second, the index $i = 0, 1, \ldots$ refers to the packet arrival instants.

The basis for estimating the frequency error is the state equation for the buffer fill level (similar to (5.3)):

$$\phi[i + 1] = \phi[i] + T[i](f_{S} - f_{R}[i]) + d[i]$$  (5.6)

where $T[i]$ is the time between sampling instants $i + 1$ and $i$, and $d[i]$ is the random fluctuation in buffer fill level due to the packet jitter. Subtracting $\phi[i]$ from both sides gives

$$\Delta \phi[i] = \phi[i + 1] - \phi[i]$$  (5.7a)

$$= T[i] \Delta f[i] + d[i].$$  (5.7b)

By taking the time-average of (5.7) it is possible to get an estimate $\hat{\Delta}f$ of the frequency error where the jitter is averaged out. The average is computed of the
methods based on jitter buffer level. this corresponds to the \(j[m] + 1\) previous sampling instants of \(\phi\). here, \(j[m]\) is a value which increases after each computed average. by choosing a larger \(j[m]\), more jitter fill be filtered out. however, a large \(j[m]\) will increase the time until \(f_R\) has converged to \(f_S\). thus, \(j[m]\) is proposed to be smaller in the beginning to shorten the pull-in time, and then gradually increased for each \(m \geq 1\). the following equation is proposed to calculate \(j[m]\):

\[
j[m] = \frac{1}{(1 - \alpha)^m} j[0]. \tag{5.8}
\]

the parameter \(j[0]\) must be chosen such that the buffer will not over- or underflow during the the first \(j[0] + 1\) packets, given the maximum frequency error which is known from the accuracy of the oscillator.

the starting points of the average calculation will now be discussed. at the first iteration the indices \(m\) and \(i\) are both zero. as described in the previous sections, there are \(j[m] + 1\) samples of \(\phi\) in the \(m\)th interval. therefore the first interval will start at \(i = 0\) and end at \(i = j[0]\), which constitutes to \(j[0] + 1\) samples. the second interval where \(m = 1\) will start at \(i = j[0] + 1\) and end at \(i = j[0] + j[1] + 1\) since the number of samples in the second interval is \(j[1] + 1\). by continuing in this fashion, a general expression for the starting points can be obtained. denoting the starting point of each interval with \(i_m\), the recursive expression for the starting points is

\[
i_{m+1} = i_m + j[m] + 1, \quad i_0 = 0. \tag{5.9}
\]

an alternative form of (5.7) is

\[
\Delta \phi[i] f_P[i] = \Delta f[i] + d[i] f_P[i] \tag{5.10}
\]

where \(f_P[i] = 1/T[i]\) is the packet arrival frequency. assuming that the packet frequency doesn’t change during the intervals \([m, m + 1]\), \(f_P[i] = f_P[i_m]\) will hold, and taking the average of (5.10) now gives

\[
\hat{\Delta} f[m] = \frac{1}{j[m]} \sum_{i = i_m}^{i_m + j[m]-1} \Delta \phi[i] f_P[i]
\]

\[
= \frac{f_P[i_m]}{j[m]} \sum_{i = i_m}^{i_m + j[m]-1} (\phi[i + 1] - \phi[i])
\]

\[
= \frac{f_P[i_m]}{j[m]} \left( \phi[i_m + j[m]] - \phi[i_m] \right) \tag{5.11a}
\]

or equivalently

\[
\hat{\Delta} f[m] = \frac{f_P[i_m]}{j[m]} (\phi[i_{m+1}] - \phi[i_m]). \tag{5.11b}
\]
Thus, by using (5.5), (5.8), (5.9) and (5.11) the local clock frequency can be adjusted so that it approaches that of the sender.

At some point, $\hat{\Delta}f[m]$ will become small. At this point it is proposed that a constant value is used for $j[m]$. When $|\hat{\Delta}f[m]| < \varepsilon$ for some $m$, $j[m]$ will not be increased, i.e. $j[m + 1] = j[m]$. Here, $\varepsilon$ is a design parameter.

## 5.3 Kalman Filtering

The Kalman filter [24] is a linear filter which makes use of the error variance. In the case where the disturbing noise is gaussian, then the Kalman filter is the optimal filter; a nonlinear filter will not give better results.

In this case the Kalman filtering [23] of the jitter buffer level uses the same formula as the time-averaging method for updating the frequency, i.e. (5.5). As with the time-averaging algorithm, the algorithm is reset whenever the fill level reaches a point too close to the top or the bottom of the buffer.

The estimated frequency error by the Kalman filter is

$$
\hat{\Delta}f[m] = \frac{p[m]}{1 + j[m]p[m]}(\phi[i_{m+1} - 1] - \phi[i_m])f_R[m] + 
\frac{1}{1 + j[m]p[m]}E\{\Delta f[m]\}
$$

(5.12)

where

$$p[m] = \frac{\pi[m]}{E\{f_R^2[m]\\hat{v}_m\}}, \quad \text{ (5.13)}$$

$\pi[m]$ is the variance of $f_R[m]$, $E\{\cdot\}$ denotes expectation value, and $\hat{v}_m$ is the estimated variance in the buffer fill level and computed by

$$
\hat{v}_m = \hat{v}[i_{m+1} - 1]
$$

(5.14a)

$$
\hat{v}[i + 1] = \frac{i}{i + 1} \hat{v}[i] + \frac{1}{i + 1} \Delta \phi^2[i].
$$

(5.14b)

For further details, see [23].
Chapter 6

Methods Based on Packet Arrival Patterns

This and the next section contain descriptions of two methods that directly or indirectly make use of timestamps contained in the RTP header. Both methods use the timing information of regular data packets and does not need separate synchronization packets.

In the first section a method based on arrival times of packets is described. This method do not make use of the timestamp to directly adjust the clock, but rather checks the timestamp to determine if the packet is useful for synchronization.

In the second section a method based on timestamp difference between packets is described. This method measures the difference in arrival times of packets and compares it to the timestamp difference of the arriving packets.

6.1 Arrival-time of Packets

With a fixed packet size, packets are transmitted at equal distance. Thus, if the transmission media doesn’t add any random delays, the packets will arrive at equal distance at the receiver. However, this is not the case with a packet network like Ethernet. Instead, packets arrive at expected time instants plus a positive random delay. If the arrival times are measured, the clock skew can be estimated [25, 26].

6.1.1 PLL Structure

The arrival time is measured with a 16-bit counter. This counter wraps around in the time between two packet arrivals. Using a system clock of 32.768 MHz, the counter will wrap around every \( \frac{2^{16}}{32.768 \cdot 10^6} = 2 \cdot 10^{-3} \) seconds. However, the packet size is usually 5, 10, or 20 ms, so in order to make the counter wrap around a scaling is required. For example with a packet size of 10 ms, a scaling of 5 makes the counter wrap around every 10 ms.
In order to generate a zero output when in steady state, a constant value is subtracted so that the expected arrival time is zero.

The PLL consists of the following blocks: a register that stores the output from the counter at each packet arrival, minimum delay calculation block, loop filter, digital-to-analog converter (DAC), voltage-controlled crystal oscillator (VCXO). The PLL structure is shown in Figure 6.1.

![Figure 6.1. PLL based on arrival time.](image)

The loop filter, shown in Figure 6.2, consists of an exponential weighting moving average (EWMA) followed by an integrator.

The moving average is a filter defined by the equation

\[ y[n] = ax[n] + (1 - a)y[n - 1] \]  

(6.1)

where \( a \in [0, 1] \) is the weighting parameter, \( x[n] \) is the input signal, and \( y[n] \) is the output signal. Choosing a value of \( a \) that is close to zero gives the filter a low-pass character, while a value close to 1 gives the filter an all-pass character. As the purpose of the filter is to attenuate the packet delay variation, a small value of \( a \) is desired.

As shown in Figure 6.2, the input signal is multiplied with \( a^2 \). This is equivalent to multiplying the input signal with \( a \) before entering the EWMA filter. The purpose of this is that for different values on \( a \), the closed loop (see Figure 6.3) step response will look approximately the same, but scaled in time.

The integrating term allows the parameter \( a \) to be changed during the start-up phase, without causing the PLL to restart the acquisition mode. By adjusting \( a \), the bandwidth of the PLL can be made wider during the acquisition mode in order to decrease the pull-in time. When the PLL has entered the tracking mode, \( a \) can be adjusted to narrow the bandwidth in order to reject as much noise as possible.

### 6.1.2 Jitter transfer

To derive the transfer function of this PLL, the DAC, VCXO and counter are replaced with an integrator. The corresponding block diagram is shown in Figure 6.3.

By analyzing this setup, and noting that the transfer function of the loop filter is

\[ H_L(z) = \left(1 + \frac{ac}{1 - z^{-1}}\right) \frac{a^2}{1 - (1 - a)z^{-1}}, \]  

(6.2)
6.1 Arrival-time of Packets

\[ a^2 + a \cdot c + z^{-1} b + z^{-1} \]

Figure 6.2. Loop filter used in the PLL based on arrival time. The left part of the filter is the EWMA filter, where \( b = 1 - a \). The rightmost part is the integrator. Here, the factor \( a \cdot c \) determines how much weight to put on the integrated signal. The factor \( a \) is used to adjust this weight during the start-up, and \( c \) is a constant.

\[ \text{Input} \rightarrow + \rightarrow \text{Loop filter} \rightarrow d \rightarrow + \rightarrow z^{-1} \rightarrow \text{Output} \]

Figure 6.3. Alternative model of the PLL.

The transfer function of the PLL can be seen to be

\[ H(z) = \frac{a^2 dz((1 + ac)z - 1)}{(z - 1)^2(z - 1 + a) + a^2 dz((1 + ac)z - 1)}. \]  \hspace{1cm} (6.3)

For the special case of \( c = 0 \), the system is reduced to a second order system. In this case \( H(z) \) becomes

\[ H(z) = \frac{a^2 dz}{(z - 1)(z - 1 + a) + a^2 dz}. \]  \hspace{1cm} (6.4)

which has poles at

\[ z = \frac{2 - (da + 1)a \pm a \sqrt{(da + 1)^2 - 4d}}{2}. \]  \hspace{1cm} (6.5)

For other values of \( c \) the pole locations are easier analyzed by using the root locus method.

As described in subsection 4.2.2 on page 18 the oscillator has a temperature drift. During the startup of the board, the oscillator temperature will rise from the ambient temperature to, let’s say 55°C. In the case of the ambient temperature being close to 25°C, this would mean a frequency change of about 4 ppm. The PLL must be able to compensate for this ramp.
The error transfer function of the PLL is

\[ H_e(z) = 1 - H(z) = \frac{1}{1 + \frac{a^2 dz((1 + ac)z - 1)}{(z-1)^2(z-1+a)}}. \]  

(6.6)

Then by using the final value theorem for the z-transform, the steady state error for a ramp can be determined.

The final value theorem [27] states that

\[ \lim_{n \to \infty} x[n] = \lim_{z \to 1} (z - 1)X(z) \]

if the region of convergence of \((z - 1)X(z)\) includes all points \(|z| > 1\).

The z-transform of a ramp \(x[n] = An\) with slope \(A\) is \(zA/(z - 1)^2\). Then

\[ \lim_{z \to 1} (z - 1)X(z)H_e(z) = \lim_{z \to 1} \frac{zA}{(z - 1)^2} \frac{1}{1 + \frac{a^2 dz((1 + ac)z - 1)}{(z-1)^2(z-1+a)}} \]

\[ = \lim_{z \to 1} \frac{(z - 1)zA}{(z - 1)^2 + \frac{a^2 dz((1 + ac)z - 1)}{z-1+a}} \]

\[ = \frac{0 \cdot A}{0 + a^2 dc} \]

\[ = 0. \]  

(6.8)

From this it is clear that the integrating part multiplied with \(ac\) causes the ramp error to approach zero. This behaviour of an integrator is however known from control theory [28]. In the same way of analysis, the step error will also approach zero.

A possible drawback of the PLL is that if a packet has an exceptionally long delay, the counter might wrap around more than one time, and thus a false delay will cause the PLL to drive the frequency in the wrong direction. This can also happen if the filter parameters are such that the PLL is very slow.

### 6.1.3 The Minimum Delay Algorithm

As described in subsection 2.2.3 on page 7, the packets exhibits a random delay variation when they travel across the network. With increasing traffic and congestion, the variation increases. However the minimum transfer delay will remain approximately constant as many packets arrive with delay near the minimum [14]. Therefore if the PLL only uses packets with low delay, the clock skew can be estimated from the trend of measured delays.

The minimum delay algorithm uses a window of size \(N\) to calculate the minimum packet delay of the last \(N\) received packets. This value is then the input to the PLL.

A disadvantage of the minimum delay algorithm is that it is useful only if there are many packets arriving with low delay. As the amount of packets with low delay
gets smaller, the input variance to the PLL increases, and thus the output becomes more oscillative.

**Window Size vs. Filter Parameter \( a \)**

The window size and the filter parameter \( a \) have a great influence of the characteristics of the PLL. If the window size is too large compared to the time constant of the PLL, an old value might remain as input for a long time. This causes the solution time to increase and the PLL might even become unstable. It is therefore interesting to see if there exists a certain optimum ratio between the window size and \( a \). A general rule of thumb is to keep the window size much smaller than the time constant.

**6.1.4 Alternatives to the Minimum Delay Algorithm**

**Minimum Average Delay**

Instead of using the minimum delays as input to the PLL, the average delay of packets that has a delay such that they are the \( \gamma \) percent fastest could be used as input.

**Optimal Detector**

One other possible alternative is to use an optimal time-detector. A very common detector used in communication systems is the *maximum-likelihood* (ML) detector \[18, 25\]. Traditionally the ML detector is used to determine the most probable symbol received from a prior known symbol set. In this context the idea of using the ML detector is to give an input signal were the jitter is reduced.

The optimal detector will be derived for two distributions, the normal distribution and the exponential distribution.

1. First, consider the case where the packet delays are truncated normally distributed \( \mathcal{N}(\mu, \sigma) \), that is, the probability density function is

   \[
   f(t) = \frac{g(\mu, \sigma)}{\sqrt{2\pi}\sigma} e^{-(\sigma(t-\mu)^2/2\sigma^2)} u(t)
   \]

   where \( u(t) \) is the unit step, given by

   \[
   u(t) = \begin{cases} 
   0, & t < 0 \\
   1, & t \geq 0 
   \end{cases}
   \]

   and \( g(\mu, \sigma) \) is a normalization constant to ensure that the total probability is 1, i.e. \( \int_0^{\infty} f(t) \, dt = 1 \). \( g(\mu, \sigma) \) is given by

   \[
   g(\mu, \sigma) = \left[ \int_0^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-(\sigma(t-\mu)^2/2\sigma^2)} \, dt \right]^{-1}.
   \]
As the probability density function is continuous, the probability that a packet has a certain delay \( t_k \) is zero. Thus it is necessary to study the probability in a differential interval \( dt \) around \( t_k \). Assuming that the arrival-times are independent, the joint probability of \( N \) arrival-times \( t_k \) is

\[
Pr(t) = \prod_{k=1}^{N} f(t_k) \, dt \\
= \prod_{k=1}^{N} \frac{1}{\sqrt{2\pi\sigma}} e^{-(t_k-\mu)^2/2\sigma^2} u(t_k) \, dt \\
= \exp \left\{ -\frac{1}{2\sigma^2} \sum_{k=1}^{N} (t_k - \mu)^2 \right\} \prod_{k=1}^{N} \frac{1}{\sqrt{2\pi\sigma}} u(t_k) \, dt 
\]  

(6.12)

where \( t = \{t_1, \ldots, t_N\} \). Finding the maximum of (6.12) is equivalent to finding the minimum of

\[
\sum_{k=1}^{N} (t_k - \mu)^2. 
\]

(6.13)

By derivating (6.13) and setting it equal to zero, it is readily found that the optimal estimation of \( \mu \) is

\[
\mu = \frac{1}{N} \sum_{k=1}^{N} t_k. 
\]

(6.14)

2. Now consider an exponential distribution of packet delays

\[
f(t) = \lambda e^{-\lambda(t-\theta)} u(t - \theta). 
\]

(6.15)

Still assuming independence, the joint probability of \( N \) arrival-times \( t_k \) is

\[
Pr(t) = \prod_{k=1}^{N} f(t_k) \, dt \\
= \prod_{k=1}^{N} \lambda e^{-\lambda(t_k-\theta)} u(t_k - \theta) \, dt \\
= \exp \left\{ -\lambda \sum_{k=1}^{N} (t_k - \theta) \right\} \prod_{k=1}^{N} \lambda u(t_k - \theta) \, dt 
\]  

(6.16)

The first factor in the above expression is maximized when \( \theta \) is maximized. However, due to the fact that the second factor is zero for \( \theta > \min\{t_k\} \), maximum is achieved for \( \theta = \min\{t_k\} \). Thus the minimum delay algorithm is actually optimal in the case of exponentially distributed delays.
6.2 Timestamps method

A different approach to control the clock frequency is to use the timestamps of RTP packets as an input signal [29]. The transmitter sends out packets with timestamps $T[n] = T_0 + n\Delta T$, where $T_0$ is some random starting value, and $\Delta T$ is the number of samples in each packet which is constant.

The receiver’s clock drives a local timestamp counter $R[n]$. When the clocks are synchronized, this counter will be incremented by $\Delta T$ during the time between two packets. When the receiver’s clock is faster than the transmitter’s clock, the timestamp counter will be incremented by more than $\Delta T$. Conversely, when the receiver’s clock is slower than the transmitter’s clock, the timestamp counter will be incremented by less than $\Delta T$. Thus, an estimate of $\Delta T$ disturbed by jitter and the frequency offset between the transmitter and receiving clock can be computed by taking the difference $\Delta R[n] = R[n] - R[n-1]$. The signal $e[n] = \Delta T - \Delta R[n]$ forms the error signal of the PLL and is the input to the loop filter. The corresponding PLL structure is shown in Figure 6.4, where the proposed loop filter is the double exponential weighting moving average (DEWMA) filter shown in Figure 6.5.

![Figure 6.4. PLL based on timestamps times.](image)

![Figure 6.5. The double exponential weighting moving average filter. Note that $\beta_k = 1 - \alpha_k$ for $k \in \{1, 2\}$.](image)
Chapter 7

Simulation Results

This chapter provides an overview of the results obtained when simulating the different methods. The results are divided into two sections, one for the methods based on buffer fill level and one for the methods based on packet arrival patterns.

The jitter measurements were performed with the MATLAB command \texttt{fft}. The MRTIE function had to be written and was first implemented directly as \texttt{Equation 4.5} on page 21. The direct computation of MRTIE is however computationally intensive so analyzing the data was quite slow. An optimized version was later implemented which uses the extreme fix method described in [30].

The methods were all simulated in MATLAB with the following parameters:

- packet delay distribution approximately as shown in \texttt{Figure 2.1} on page 8
- packet size of 160 samples or 20 ms
- initial frequency offset of 25 ppm
- simulation time of 1500 seconds
- jitter and MRTIE measurements started 200 seconds into the simulations.

Longer simulations than 1500 seconds were also made to see if there were any long-term issues with any of the methods. The only method that could be seen to have such issues was the PD method.

7.1 Results for the Jitter Buffer-based Methods

7.1.1 Proportional and Differential Method

The PD method was simulated with the following parameters: proportional factor $\alpha = 20$, differential factor $\beta = 0$, block separation $J = 7$, window size $M = 80$.

The PD method performed well as shown in \texttt{Figure 7.1} to \texttt{7.4} on pages 38–39. Jitter is low due to the fact that it uses the maximum fill level to adjust the
frequency. However, when the deviation from the optimal position is taken into account, i.e. $\beta > 0$, and the simulation lasted longer than 1500 seconds, then large changes in the time error occur when the fill level approaches the optimal position.

**Figure 7.1.** Frequency error for the PD method. The dashed lines show the ±25 ppm levels, where 25 ppm is the initial frequency error.

**Figure 7.2.** Time error results for the PD method.
7.1 Results for the Jitter Buffer-based Methods

Figure 7.3. MRTIE results for the PD method. The dashed line is the upper limit of MRTIE from the G.823 recommendation.

Figure 7.4. Jitter amplitude for the PD method. The dashed line is the upper limit of peak-to-peak jitter amplitude from the G.823 recommendation.
7.1.2 Time-averaging Method

The time-average method was simulated with the following parameters: $\alpha = 0.5$, $j[0] = 500$. As shown in Figure 7.5 to 7.8 on pages 40–41, the limits for jitter and wander are exceeded. The cause of this is that the frequency is constant for long periods of time while being "far" from the transmitter’s frequency.

![Graph showing frequency error for the time-averaging method.](image)

**Figure 7.5.** Frequency error for the time-averaging method. The dashed lines show the $\pm25$ ppm levels, where 25 ppm is the initial frequency error.

![Graph showing time error results for the time-averaging method.](image)

**Figure 7.6.** Time error results for the time-averaging method.
7.1 Results for the Jitter Buffer-based Methods

Figure 7.7. MRTIE results for the time-averaging method. The dashed line is the upper limit of MRTIE from the G.823 recommendation.

Figure 7.8. Jitter amplitude for the time-averaging method. The dashed line is the upper limit of peak-to-peak jitter amplitude from the G.823 recommendation.
7.1.3 Kalman-based Method

The Kalman-based method was simulated with the following parameters: $\alpha = 0.5$, $j[0] = 50$. The pull-in time was short and jitter amplitude was very low as well. The MRTIE values were the lowest of the methods. However, the Kalman filter exhibits a decreasing trend in the time error as shown in Figure 7.10. This did however not cause the jitter limits to be exceeded for low frequencies. The reason behind the decreasing trend in the time error is that the frequency error was approximately constant and somewhat smaller than zero. For plots of the results, see Figure 7.9–7.12 on pages 42–43.

![Figure 7.9](image)

**Figure 7.9.** Frequency error for the Kalman method. The dashed lines show the $\pm 25$ ppm levels, where 25 ppm is the initial frequency error.

![Figure 7.10](image)

**Figure 7.10.** Time error results for the Kalman method.
7.1 Results for the Jitter Buffer-based Methods

![Figure 7.11](image1.png)

**Figure 7.11.** MRTIE results for the Kalman method. The dashed line is the upper limit of MRTIE from the G.823 recommendation.

![Figure 7.12](image2.png)

**Figure 7.12.** Jitter amplitude for the Kalman method. The dashed line is the upper limit of peak-to-peak jitter amplitude from the G.823 recommendation.
7.2 Results for the Methods Based on Packet Arrival Patterns

7.2.1 Arrival-time of Packets Method

The simulations were done as follows: Initial value on $a$ was 0.04 which was reduced to 0.0025 in three steps. Initial window size was 8 packets which was increased to 64 in two steps. The values on $c$ and $d$ were 0.1 and 0.005 respectively.

The results, see Figure 7.13–7.16 on pages 44-46, show that the packet arrival method performs well. The reason behind this is that it only uses packets with low delay.

![Figure 7.13. Frequency error for the packet arrival method. The dashed lines show the ±25 ppm levels, where 25 ppm is the initial frequency error.](image-url)
7.2 Results for the Methods Based on Packet Arrival Patterns

Figure 7.14. Time error results for the packet arrival method.

Figure 7.15. MRTIE results for the packet arrival method. The dashed line is the upper limit of MRTIE from the G.823 recommendation.
Figure 7.16. Jitter amplitude for the packet arrival method. The dashed line is the upper limit of peak-to-peak jitter amplitude from the G.823 recommendation.
Window Size vs. Filter Parameter $a$

As described on page 33, the window size and filter parameter $a$ affect the characteristics of the PLL. This section shows some results on how the filter parameters influence the PLL in terms of solution time and frequency error.

Measurements on solution time and frequency error were made for different values on window size, $a$ and $d$. The parameters $a$ and window size were kept constant during the simulations as opposed to gradually decrease $a$ and increase the window size. A low value is desirable for solution time as well as frequency error. These values are indicated with a dark color in the plots. Hence, good values on $a$ and window size are achieved where two dark areas are overlapping each other.

Three sets of simulations were done for $d \in \{0.001, 0.005, 0.01\}$. The parameter $a$ was in the range of $0.001 - 0.02$, while the window size was in the range of $1 - 3500$ packets. Packet size and $c$ were as stated previously, i.e. 160 samples and 0.1 respectively. The results are shown in Figure 7.17–7.19.

As shown, the solution time and frequency error rapidly increases outside the grey region. The white values were actually larger than shown and have been clamped in order to show more details in the grey regions.

Minimum Average Algorithm

Some simulation results are presented in Table 7.1. With 40% and 50% as the amount of packets used in the algorithm, the MRTIE limits are exceeded.

For comparison, the minimum delay algorithm gave frequency errors in the interval $-0.26$ ppm to $0.24$ ppm. Thus the minimum average algorithm could give better performance than the minimum delay algorithm.

**Table 7.1.** Simulation results for the minimum average algorithm with initial values $d = 0.005$, $a = 0.05$, $c = 0.1$, $N = 2$. Minimum and maximum frequency error in ppm measured starting from 200 seconds.

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Min ppm error</th>
<th>Max ppm error</th>
<th>G.823 compliant</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>$-0.58$</td>
<td>$0.66$</td>
<td>No</td>
</tr>
<tr>
<td>40</td>
<td>$-0.44$</td>
<td>$0.48$</td>
<td>No</td>
</tr>
<tr>
<td>30</td>
<td>$-0.23$</td>
<td>$0.22$</td>
<td>Yes</td>
</tr>
<tr>
<td>20</td>
<td>$-0.17$</td>
<td>$0.18$</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>$-0.16$</td>
<td>$0.18$</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>$-0.18$</td>
<td>$0.19$</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>$-0.23$</td>
<td>$0.26$</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Figure 7.17. Frequency error [a] and solution time [b] for different values of \(a\) and window size when \(d = 0.001\). Low values on solution time and frequency error are indicated with a dark color. Hence, good values for \(a\) and window size are achieved where two dark areas are overlapping each other.
7.2 Results for the Methods Based on Packet Arrival Patterns

Figure 7.18. Frequency error (a) and solution time (b) for different values of $a$ and window size when $d = 0.005$. Low values on solution time and frequency error are indicated with a dark color. Hence, good values for $a$ and window size are achieved where two dark areas are overlapping each other.
Figure 7.19. Frequency error [a] and solution time [b] for different values of $a$ and window size when $a = 0.01$. Low values on solution time and frequency error are indicated with a dark color. Hence, good values for $a$ and window size are achieved where two dark areas are overlapping each other.
7.2 Results for the Methods Based on Packet Arrival Patterns

7.2.2 Timestamps Method

The method was simulated with the values $\alpha_1 = \alpha_2 = 0.005$. The frequency error is rather noisy which is because all packets are used in calculating the frequency estimate. Because it is the interarrival-time that is being used as a means of controlling the frequency, no information can be obtained of which packets are received with minimum delay.

The pull-in time was short but the MRTIE limits are exceeded. See Figure 7.20–7.23 on pages 51–52.

![Figure 7.20](image1.png)

**Figure 7.20.** Frequency error for the timestamps method. The dashed lines show the ±25 ppm levels, where 25 ppm is the initial frequency error.

![Figure 7.21](image2.png)

**Figure 7.21.** Time error results for the timestamps method.
Figure 7.22. MRTIE results for the timestamps method. The dashed line is the upper limit of MRTIE from the G.823 recommendation.

Figure 7.23. Jitter amplitude for the timestamps method. The dashed line is the upper limit of peak-to-peak jitter amplitude from the G.823 recommendation.
7.3 Summary

The PD method as well as the Kalman filter had the best performance of the methods based on buffer fill level. The time-average method cannot be used to recover a clock signal that complies with G.823. The assumption that the fill level can be read with a resolution of samples may not hold. If re-ordering of out-of-sequence packets, then an integer sized buffer has to be used \[14\], which lowers the resolution to packets. A resolution of packets is likely to be too coarse to be able to achieve a clock signal in compliance G.823.

The method based on arrival-times of packets is better in all respects than the timestamps method. This is likely because only packets with low delay are used.

When packet loss is concerned, the buffer based methods are likely to be more vulnerable. If a packet is lost, then the buffer fill level decreases which will be interpreted as the local clock frequency higher than the sender’s clock frequency. The PD method should however be more resistant to packet loss than the others as it uses the maximum fill level of the buffer. However, if packet loss can be detected, for example with the use of sequence numbers, then the change in buffer level caused by packet loss could be compensated for when adjusting the frequency. The methods based on packet arrival patterns should be more insensitive to packet loss. Take the timestamps method for instance. If a packet is lost, then the counter will count longer than normally, but the timestamp difference will also be larger, hence the lost packet will not disturb the frequency much.
Chapter 8

Summary

Five methods for frequency synchronization of POTS systems connected over Ethernet have been described and simulated in this thesis. The first three methods are based on observing the fill level of the jitter buffer. If the buffer fill level is decreasing, the local frequency is likely to be higher than the sender’s frequency. Hence, frequency can be adjusted by filtering the buffer fill level. The first method is a proportional-differential (PD) regulator that filters the maximum fill level of the buffer. The second and third method uses the time-average of buffer fill level. The third method differ from the second in that it uses a Kalman filter instead of just using the time-average of the buffer fill level.

The fourth and fifth methods use the arrival pattern of packets as a means of adjusting the local frequency. The fourth method measures the delay that each packet experience. If the measured delay increases, the local frequency is likely to be higher than the sender’s frequency. The fourth method only uses packets with minimum delay, which is similar to the PD regulator that uses the maximum fill level of the buffer. The fifth method measures the time between packet arrivals and compares it to the difference between the timestamps contained in the RTP packets. If the measured time is greater than the timestamp difference, then the local frequency is higher than the sender’s frequency.

The simulation results showed that the PD, Kalman-filtering, and packet-arrival methods are able to regulate the receiver’s sample clock frequency in such a way that it complies with the G.823 recommendation ratified by ITU-T. The methods that use the buffer fill level for the adjustment of the local frequency have been simulated to be able to measure the buffer fill level with a resolution of samples. It is unlikely that the PD and Kalman-filtering methods will be able to comply with G.823 if instead a resolution of packets were to be used.
8.1 Further Work

In order to fully verify that a clock signal of adequate quality can be generated, measurements on actual hardware must be made. This will show if packet loss and intrinsic jitter from the local oscillator will be a problem.

The influence of the shape of the packet delay distribution could also be studied more in detail. For example, what happens if the packet delay is normally distributed? If so, the minimum average delay algorithm might be more suitable than the minimum delay algorithm used in the method based on arrival-time of packets.
Bibliography


Appendix A

Notation

Symbols

\( f_P \)  Packet arrival frequency
\( f_R \)  Receiver clock frequency
\( f_S \)  Source clock frequency
\( \phi[i] \)  Jitter-buffer level.

Operators and Functions

\( E\{\cdot\} \)  Expectation value.

Acronyms and Abbreviations

ATM  Asynchronous Transfer Mode
CODEC  Coder and Decoder
CSMA/CD  Carrier Sense Multiple Access with Collision Detection
DAC  Digital-to-analog Converter
DCO  Digitally Controlled Oscillator
DEWMA  Double Exponential Weighting Moving Average
DSL  Digital Subscriber Line
EWMA  Exponential Weighting Moving Average
GPS  Global Positioning System
ID  Identity Display
IEEE  Institute of Electrical and Electronics Engineers
IETF  Internet Engineering Task Force
IP  Internet Protocol
ISDN  Integrated Services Digital Network
ITU  International Telecommunication Union
ITU-T  Telecommunication standardization sector of ITU
IWF  Interworking Function
LPF  Low-pass Filter
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML</td>
<td>Maximum-Likelihood</td>
</tr>
<tr>
<td>MTIE</td>
<td>Maximum Time Interval Error</td>
</tr>
<tr>
<td>MRTIE</td>
<td>Maximum Relative Time Interval Error</td>
</tr>
<tr>
<td>NRZ</td>
<td>Nonreturn to Zero</td>
</tr>
<tr>
<td>PCM</td>
<td>Pulse Code Modulation</td>
</tr>
<tr>
<td>PD</td>
<td>Phase Detector, or Proportional and Differential</td>
</tr>
<tr>
<td>PDV</td>
<td>Packet Delay Variation</td>
</tr>
<tr>
<td>PLC</td>
<td>Packet Loss Concealment</td>
</tr>
<tr>
<td>PLL</td>
<td>Phase-locked Loop</td>
</tr>
<tr>
<td>pk-pk</td>
<td>peak-to-peak</td>
</tr>
<tr>
<td>POTS</td>
<td>Plain Old Telephone Service</td>
</tr>
<tr>
<td>ppm</td>
<td>Parts per million</td>
</tr>
<tr>
<td>PRC</td>
<td>Primary Reference Clock</td>
</tr>
<tr>
<td>PSTN</td>
<td>Public Switched Telephone Network</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RTCP</td>
<td>RTP Control Protocol</td>
</tr>
<tr>
<td>RTP</td>
<td>Real-time Transport Protocol</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TIE</td>
<td>Time Interval Error</td>
</tr>
<tr>
<td>TDM</td>
<td>Time Division Multiplexing</td>
</tr>
<tr>
<td>ToIP</td>
<td>Telephony over Internet Protocol</td>
</tr>
<tr>
<td>ToS</td>
<td>Type of Service</td>
</tr>
<tr>
<td>UI</td>
<td>Unit Interval</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>VCO</td>
<td>Voltage-controlled Oscillator</td>
</tr>
<tr>
<td>VCXO</td>
<td>Voltage-controlled Crystal Oscillator</td>
</tr>
<tr>
<td>VoDSL</td>
<td>Voice over Digital Subscriber Line</td>
</tr>
<tr>
<td>XO</td>
<td>Crystal Oscillator</td>
</tr>
</tbody>
</table>
Copyright

The publishers will keep this document online on the Internet – or its possible replacement – for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

Upphovsrätt

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämlig som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

© Jonatan Lindblad 2005