UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUÇÃO EM ENGENHARIA ELÉTRICA

FPGA Implementation and Evaluation of Synchronization Architectures for Ethernet-based Cloud-RAN Fronthaul

Igor Antonio Auad Freire

DM: 01/2016

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2016

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUÇÃO EM ENGENHARIA ELÉTRICA

Igor Antonio Auad Freire

FPGA Implementation and Evaluation of Synchronization Architectures for Ethernet-based Cloud-RAN Fronthaul

DM: 01/2016

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2016

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUÇÃO EM ENGENHARIA ELÉTRICA

Igor Antonio Auad Freire

FPGA Implementation and Evaluation of Synchronization Architectures for Ethernet-based Cloud-RAN Fronthaul

A dissertation submitted to the examination committee in the graduate department of Electrical Engineering at the Federal University of Pará in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering with emphasis in Telecommunications.

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2016

Dados Internacionais de Catalogação-na-Publicação (CIP)
Sistema de Bibliotecas da UFPA

Freire, Igor Antonio Auad, 1991-
    Fpga implementation and evaluation of synchronization architectures for ethernet-based cloud-ran fronthaul / Igor Antonio Auad Freire. - 2016.

    Orientador: Aldebaro Barreto da Rocha Klautau Júnior.
    Dissertação (Mestrado) - Universidade Federal do Pará, Instituto de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Belém, 2016.

    1. Telefonia celular. 2. Arranjos de lógica programável em campo. 3. Computação em nuvem. 4. Serviços da Web. I. Título.

CDD 22. ed. 621.38456

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

"IMPLEMENTAÇÃO E AVALIAÇÃO EM FPGA DE ARQUITETURAS PARA SINCRONIZAÇÃO VIA FRONTHAUL BASEADO EM PACOTES EM CLOUD-RAN"

AUTOR: **IGOR ANTÔNIO AUAD FREIRE**

DISSERTAÇÃO DE MESTRADO SUBMETIDA À BANCA EXAMINADORA APROVADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, SENDO JULGADA ADEQUADA PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA NA ÁREA DE TELECOMUNICAÇÕES.

APROVADA EM: 18/01/2016

BANCA EXAMINADORA:

Prof. Dr. Aldebaro Barreto da Rocha Klautau Junior
(Orientador – PPGEE/UFPA)

Prof. Dr. Eduardo Coelho Cerqueira
(Avaliador Interno – PPGEE/UFPA)

Prof. Dr. Adalbery Rodrigues Castro
(Avaliador Externo ao Programa – FCT/UFPA)

VISTO:

Prof. Dr. Evaldo Gonçalves Pelaes
(Coordenador do PPGEE/ITEC/UFPA)

Acknowledgments

First, I would like to thank my advisor, Prof. Aldebaro Klautau, for the intellectual guidance on this work, for the motivational support during my masters degree program and for the knowledge provided during this fortunate interaction over the past two years.

Secondly, I wish to thank Prof. Adalbery Castro and Prof. Eduardo Cerqueira for serving as readers and examiners of this dissertation.

Third, I would like to thank the colleagues that collaborated to this work. Special thanks to Ilan Souza and Joary Fortuna, who in many occasions collaborated with solutions to the issues that arose during the FPGA hardware development presented in this work. I was privileged to have received insights from their expertise in the field. Thanks also to Leonardo Ramalho, who collaborated with the use of the measurement equipments adopted in this material.

I also wish to express my gratitude to the colleague Igor Almeida, who collaborated with many thoughtful discussions that motivated the ideas investigated in this work and participated from the very beginning of the research in the process of gaining perception on the topic.

Furthermore, I am grateful to the colleagues from the Signal Processing Laboratory at the Federal University of Pará who made the past years of work enjoyable.

Last and most importantly, I wish to thank to my father Antonio Freire, my mother Rosangela Auad, my brother Gustavo Freire and my girlfriend Camila Matni for all their love and support that motivates me to pursue excellence as a professional and a human being.

Igor Antonio Auad Freire

January 2016

List of Symbols

ADC Analog to Digital Conversion

ASIC Application Specific Integrated Circuit

AVB Audio-Video Bridge

AxC Antenna-carrier

BBU Baseband Unit

BER Bit Error Rate

BF Basic Frame

BS Base Station

BW Bandwidth

CBR Constant Bit Rate

CDR Clock and Data Recovery

CPRI Common Public Radio Interface

DAC Digital to Analog Conversion

DAS Distributed Antenna System

DL Downlink

FDD Frequency Division Duplexing

FEC Forward Error Correction

FIFO First-in First-out

FPGA Field Programmable Gate Array

HARQ Hybrid Automatic Repeat Request

HDL Hardware Description Language

HFN Hyperframe Number

ICIC Inter-cell interference coordination

IEEE Institute of Electrical and Electronics Engineers

IP Internet Protocol

IQ In-phase and Quadrature

ISR Interrupt Service Routine

ITU International Telecommunications Union

LTE Long Term Evolution

MAC Medium Access Control

MIMO Multiple Input Multiple Output

NTP Network Time Protocol

OBSAI Open Base Station Architecture Initiative

OC Ordinary Clock

OFDM Orthogonal Frequency Division Multiplexing

ORI Open Radio Interface

OTN Optical Transport Networks

PDH Plesiochronous Digital Hierarchy

PDV Packet Delay Variation

PHY Physical Layer

PLL Phase Locked Loop

PRC Primary Reference Clock

PTP Precision Time Protocol

RAN Radio Access Network

RF Radio Frequency

RRH Remote Radio Head

RRU Remote Radio Unit

RTC Real Time Clock

SDH Synchronous Digital Hierarchy

TAE Time Alignment Error

TC Transparent Clock

TDD Time Division Duplexing

TDM Time Division Multiplexing

TIE Time Interval Error

UE User Equipment

UL Uplink

VCO Voltage-controlled Oscillator

VHDL Very High Speed Integrated Circuits Hardware Description Language

WDM Wavelength Division Multiplexing

List of Figures

List of Tables

Contents

Abstract

The use of Ethernet infrastructure available in most commercial buildings can alleviate the costs involved with fronthaul provisioning in cloud radio access networks. However, current fronthaul specifications rely on dedicated synchronous links, which natively support features such as accurate synchronization across radio devices. In this context, a cost-effective and backwards-compatible solution is to adapt standard fronthaul interfaces (e.g. CPRI) to asynchronous networks by using endpoint equipments capable of meeting synchronization requirements over legacy Ethernet. This dissertation proposes design considerations for synchronization of radio over Ethernet and evaluates the feasibility of synchronization architectures by developing an FPGA-based hardware testbed. It contrasts two solutions, a simplistic approach that employs elastic buffering for recovering frequency and an end-to-end IEEE 1588 Precision Time Protocol solution for time and frequency alignment. Results suggest that the scheme with PTP solely at the endpoints can comply with time and frequency alignment requirements of current mobile standards if packet delay variation is treated with sound algorithms.

Index Terms— Synchronization, Fronthaul, Precision Time Protocol, FPGA, Cloud-RAN

Resumo

A utilização da infra-estrutura Ethernet disponível na maioria dos edifícios comerciais pode aliviar os custos envolvidos com provisionamento de fronthaul em redes de acesso de rádio em nuvem. No entanto, as especificações atuais de interfaces fronthaul contam com links síncronos e dedicados, os quais suportam nativamente recursos como a distribuição da sincronização através da camada física dos dispositivos. Neste contexto, uma solução de bom custo benefício e compatível com protocolos e equipamentos em atual uso é adaptar as interfaces fronthaul padrão (e.g. CPRI) para redes assíncronas usando equipamentos capazes de atingir requisitos de sincronização através de redes Ethernet legadas. Esta dissertação propõe considereções de projeto e avalia a viabilidade de arquiteturas de sincronização em tal cenário, através do desenvolvimento de um banco de testes com hardware baseado em FPGA. São contrastadas duas soluções: uma abordagem simples que emprega fila elástica de recepção para a recuperação de frequencia e uma solução fim-a-fim utilizando o protocolo de precisão de tempo IEEE 1588 para alinhamento em tempo e frequencia. Resultados sugerem que o esquema utilizando PTP somente nos pontos finais pode atingir os requisitos de sincronização de tempo e frequencia estabelecidos por padrões de telefonia móvel atuais se a variação de atraso de pacote for tratada com razoáveis estratégias implementadas a nível de algoritmo.

Palavras-chave — Sincronismo, Fronthaul, Protocolo de Preciso de Tempo, FPGA, Cloud-RAN

Chapter 1

Introduction

The evolution of both radio access technologies and application layer services offered to mobile subscribers are constantly driving enormous changes in mobile systems. A case of interest is the rise of the cloud computing paradigm, which is driving the introduction of its technologies (e.g. *virtualization*) into the infrastructure of mobile systems. The concept has became a buzzword in the telecommunication industry, known as cloud radio access networks, often referred to as Cloud-RAN or simply C-RAN. The rising popularity of the technology comes from the strong potential of cloudification for improving many aspects of traditional RANs, such as efficiency in the allocation of resources, manageability and power consumption.

Naturally, the emergence of Cloud-RAN turns questionable many technologies that were consolidated in traditional networks. One of them, which is of interest in this work, encompasses the communication stack adopted in the *fronthaul*, namely in the networks that carry baseband radio data between baseband and radio-frequency equipments. In order to foster flexible and efficient Cloud-RANs, the fronthaul must undergo modifications. This chapter will introduce the reason why packet-based networks and especially the ubiquitous Ethernet are strong candidates to complement or replace current synchronous *fronthaul* networks.

Although the replacement of current synchronous *fronthaul* networks by packet-based networks has an upside on flexibility improvement, it introduces new challenges. An obstacle is the fact that synchronous networks inherently enable the delivery of synchronization via the physical layer, while in packet-based networks alternative methods are necessary. This is in fact a classical issue in packet-based networks, reason why the industries that rely on the delivery of synchronization have standardized solutions to accomplish this task. The problem, however, is that the achievable accuracy has to be made compatible with the needs of current and future

mobile systems. Such investigation is one of the objectives of this work.

More specifically, the goal is to analyze and prove the concept of packet-based synchronization strategies compatible with current 4th-generation (4G) and future RAN requirements. In light of this scope, this chapter will initiate the discussion by introducing the evolution of the radio infrastructure, motivating the demand for increased flexibility in fronthaul networks and clarifying the need for network delivery of frequency and time synchronization.

1.1 From Traditional to Cloud-based RAN

Traditional standalone base stations have been adopted since the beginning of mobile systems. In this architecture, the entire baseband and radio-frequency (RF) processing is performed in a single standalone unit, whose output is an RF signal typically fed via a coaxial cable towards a passive antenna placed at the top of towers or rooftops, as illustrated in Fig. 1.1 [4]. This architecture, however, suffers from severe radio transmission losses along the analog path from base station to antenna. For this reason, vendors and operators realized that one solution was to place the RF processing closer to antennas (outdoors) in order to achieve better performances.

The evolution lead to the BBU-RRH separated architecture, where baseband and RF front-end are separated among units known as baseband unit (BBU) and remote radio head (RRH), respectively. The former, just like the previous generation standalone unit, is placed in a particular specially conditioned room at the site where the served antennas are placed (e.g. at the bottom of a mast). The latter (RRH) is the portion that is placed closer to antenna, as illustrated in Fig. 1.2. This architecture is also often named as distributed RAN (D-RAN)[1].

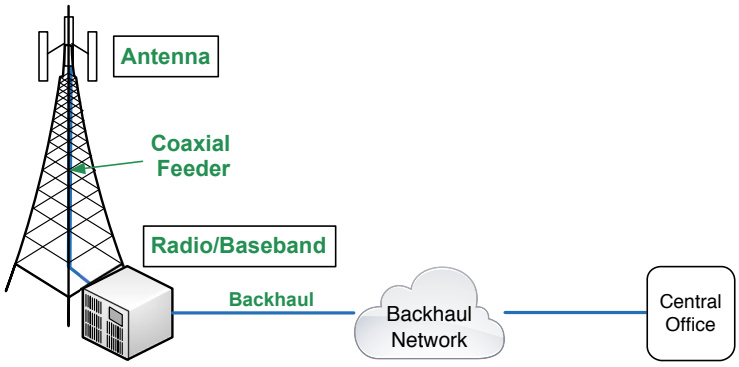


Figure 1.1: Traditional radio base stations.

[1]It is common to find the term D-RAN referring to both the BBU-RRH architecture or the centralized RAN.

In the BBU-RRH separated architecture, the RRH is solely responsible for operations (mostly analog) on the modulated signals received from baseband units, such as filtering, up or downconversion, power amplification and cross-factor reduction. Meanwhile, the baseband unit is responsible for significantly more processing, including physical layer (PHY) functionality, medium access control (MAC), and layer 3 (network) functionalities to allow transportation of data received via radio interface to the core of the network over the *backhaul*. Then, BBUs send the baseband processed signals via digital communication links to the RRHs, which are responsible for simple decoding and RF processing necessary prior to feeding the antenna.

The communication between BBUs and RRHs is typically over digital optical fiber links. Such strategy overcomes the transmission losses that affected previous standalone RANs, since RRHs are wired to nearby or integrated antennas. When not integrated, antennas are typically connected via coaxial cable connections of 2 to 3 m [4]. In contrast, when RRHs are integrated with antennas (what is known as active antennas), even lower transmission losses between RRH and antenna are attained. Furthermore, the referred integration can provide other advantages such as simpler installation, reduced footprint and avoided wiring expenses.

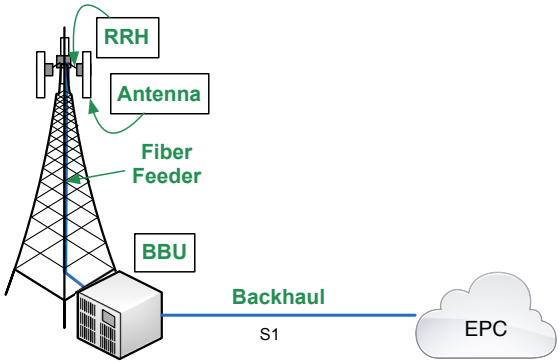


Figure 1.2: BBU-RRH separated radio base stations.

The BBU-RRH separated radio access networks continued to evolve. The telecommunications industry at some point recognized that a further physical separation between RRHs and BBUs could be even more cost-effective, which lead to the architecture known as *centralized RAN*. In this RAN generation, baseband units are moved far-apart to centralized locations, often known as central rooms, while radio modules remain close or integrated with antennas at positions desirable for coverage (e.g. at the top of masts), as illustrated in Fig. 1.3.

The motivation for extending the separation is that, by centralization of base stations and baseband signal processing, operators can achieve improved efficiency and reduced costs, both

in terms of capital expenditures (CapEX) and operational expenditures (OpEx). For example, when base stations are centralized, power supply can be shared, power consumptions is reduced (e.g. shared air conditioning), maintenance expenditures can be more efficiently allocated, physical upgrades are facilitated and the space required at cell sites is reduced (reduced rental expenses). As an illustrative case, [5] mentions an experiment that observed 41% power consumption reduction due to shared air-conditioning of 21 base stations that were aggregated to support 101 remote radio units. In addition to these savings, the own infrastructure for deployment of interconnection between base stations is minimized and, if the baseband units are co-located with the core of the network, backhauling expenses can be minimized too.

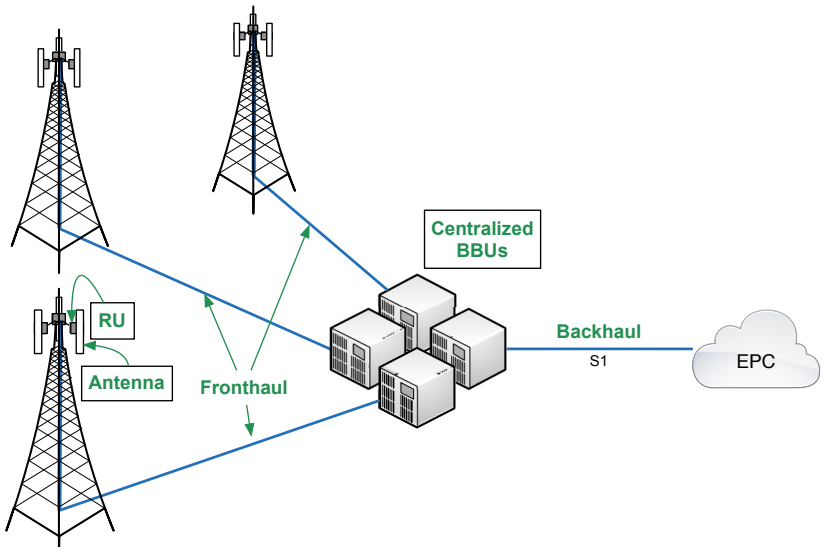


Figure 1.3: Centralized radio access networks.

Another factor that drove the centralization of access networks was the growth in indoor mobile traffic and the upsurge of small cell deployments in heterogeneous networks. According to [6], nearly 70% of mobile usage happens indoors. Historically, indoor coverage has been provided via distributed antenna systems (DAS), which redistribute macro cell signals to and from an indoor antenna grid, typically using passive coaxial RF networks or active fiber-coaxial hybrid network. Today, in order to offload macro cells and improve user experience, operators are often more interested on the deployment small cells (residential, commercial and public) than DAS. This is because DAS can be economically infeasible if not used with multiple RF bands and not shared by multiple-operators [7]. Besides, in contrast to DAS, small cells provide frequency reuse, namely its deployment creates new sectors and enhances the network capacity (provided backhauling does not impose a bottleneck). Additionally, small cells can introduce

interference management capabilities, so that even higher data-rates can be achieved.

In the context of the rising deployment of small cell (e.g. femto, enterprise, pico and metro), centralization of baseband units emerged as an opportunity for operators to reduce the complexity, power consumption and maintenance needs of radio units deployed within indoor environments. Additionally, centralization allows simplification of backhauling from small cell radio units to central baseband processing locations. Since small cells are meant to serve a small number of users, centralization avoids the requirements and costs that would otherwise be incurred with complex backhauling, which could turn the investment infeasible. Therefore, an overall reduction in the total cost of ownership (TCO) of the networks can be achieved.

Yet another motivation for centralization is the capability of implementing joint processing with lower latencies. Since densification of networks made inter-cell interference more severe than in previous mobile generations (2G or 3G), collaborative radio techniques became essential for improving capacity. Many techniques such as *joint-transmission* (JT) or *coordinated scheduling and beamforming* (CS/BF) from LTE release 11 [8] emerged for coordination of BBUs. These techniques allow the coordinated baseband units to make better decisions in terms of interference management, by leveraging on information received from BBUs serving adjacent cells or sectors. However, coordination requires low latencies between base stations [5], so centralization of BBUs becomes appealing to reduce the distances and latencies.

In the centralized RAN, similar to the BBU-RRH separated architecture, radio and baseband units communicate through high-speed optical links. However, the two RAN generations differ in the distance between these units, which is significantly higher in the centralized architecture. The latter was conceived with the objective of reducing the number of equipment rooms at sites [9] through the aggregation of several baseband units on a shared location, so that distances to RF units are increased. Another difference relates to the remote radio equipments, which generally encompass more digital signal processing in the centralized RAN relative to the previous generation. This is why these equipments are often named distinctly as radio units (RUs) or remote radio units (RRUs). Fig. 1.3 highlights the higher distances between baseband and radio units and emphasizes the *fronthaul*, defined as the portion of the RAN containing the aforementioned high-speed links between BBUs and RUs, which will be the focus of this work.

The centralized RAN infrastructure gained (and continues to gain) widespread adoption, but mobile traffic growth continuously change the requirements posed to RANs. First, rising smartphone subscriptions and increasing data consumption per subscriber are leading to fore-

casts of nine-fold increase in global mobile traffic by the end of 2020 [10], while the increasing number of connected devices leads to forecasts of 26 billion connected devices by the same year. Secondly, not only unprecedented traffic volume numbers will be experienced, but also a large quantity of new applications are expected to be served by future generation RANs, such as machine-to-machine communications and driverless cars. These applications will introduce different tiers of requirements in terms of latency, data-rate and energy consumption [11]. Finally, mobile traffic growth enhances the traffic load variations throughout the day and according to location, known as *traffic imbalance*. For example, traffic loads tend to show increased peaks near business areas during working hours and near residence areas at night [9]. Therefore, in the context of these predictions, it is natural that access networks must be subject to adaptation.

Historically, the adaptation with respect to traffic growth has been mainly provisioned through further network densification and increased spatial reuse. Studies show that these alone provided 1600-fold gains since 1957, in contrast to minor improvements due to spectrum widening (25-fold gain) and PHY layer improvements (5-fold gain) [12]. This is because by shrinking cell sizes, user density per cell is decreased and competition for spectrum resources is correspondingly reduced, so as long as interference is adequately controlled the limit becomes the point in which a base station serves a single user. Currently, it is believed that this limit is very far from being explored by operators, so network densification is set to continue as one of the most effective solutions for increased throughput [13]. More specifically, together with two other important solutions[2], forthcoming fifth-generation (5G) mobile systems are expected to exploit very high base station densities in the so-called *ultra-densification* [11].

However, extreme densification comes at several prices. For example, ultra-dense small deployments increase the infrastructure installation and maintenance costs for backhauling, increase the signaling in the network due to e.g. more frequent handover procedures and require higher degrees of radio coordination and interference management (e.g. the aforementioned inter-cell interference mitigation techniques). Furthermore, denser heterogeneous deployments increase the probability for a particular base station node to be idle when serving a small number of users. In this scenario, the centralized infrastructure is rapidly becoming obsolete due its limited flexibility and fixed usage of radio and baseband resources.

In small cell deployments, the user density per base-station decreases, which implies more

[2]Improvements in capacity for 5G are also expected to come from very high carrier frequencies (in e.g. millimeter wavelength) with massive bandwidths (likely up to tens of gigahertz for subterahertz bands) and unprecedented numbers of antennas coupled to higher order multi-antenna processing.

diverse traffic characteristics as well as the opportunity of more efficient load balancing among small-cells. Today, imbalance is such that about 15% of all sites carry about 50% of the total traffic [14]. Moreover, ever-increasing mobile traffic accentuates the range of traffic imbalance, specially in events that physically or remotely attract a large number of people. For example, data reports in [10] show that mobile traffic was five times higher than average busy hour traffic during the soccer world championship final in 2014. These facts together call for dynamical allocation of resources (baseband or radio processing) on macro and small cells.

The evolution in terms flexibility of resource allocation lead to the concept of cloud-RAN (C-RAN), which was first proposed in 2009. In contrast to centralized RAN, Cloud-RAN intends to explore cloud computing technologies to concentrate baseband processing instances of multiple base stations (BSs) into one cloud server [15]. The goal is to improve management and allocation of resources, by enabling scalable and load-driven pooling of resources on demand.

In the C-RAN, the resources pertaining to a particular base station are not exclusively used by it anymore. Instead, these resources can be shared between coordinated base stations, through an implementation of what is known in clouding computing as *virtualization* or *cloudification*. For example, in this architecture, signals from a geographically far-apart RRH can be processed by any baseband card in the pool of base stations allocated for the task. This feature helps to balance the load, switch off baseband units during periods of low activity and, in general, helps with maintenance. From a similar perspective, through the dynamic allocation of resources, it can be stated that C-RAN leverages on the cloud computing concept known as *elasticity*, which is the autonomous adaptation to non-uniform workloads posed by traffic imbalance. All of these features are also leading to the term *radio access network as a service* (RANaaS) [14], similar to the widely used cloud computing terminologies adopted for e.g information technology infrastructure as a service (IaaS) or software as a service (SaaS).

A typical C-RAN network is composed by a the baseband processing cloud (BBU pool), the RRUs and the fronthaul, as in Fig. 1.4. The BBU pool is deployed at a centralized location and is composed by a dynamic array of BBU instances. These instances, in turn, contain processing resources and capabilities that are allocated and reconfigured based on real-time conditions. In fact, this nature of BBU instances motivated the nomenclature of *"soft" BBU nodes* in [5]. The RRUs are composed by slightly more complex digital systems capable of interfacing with BBU pools. Finally, the *fronthaul* shall provide flexible, power-efficiently managed and yet cost-effective transport between distributed antennas, i.e. RRUs and BBU instances. It is

similar to the one in the centralized RAN with respect to infrastructure (often assumed to be composed by optical fiber cables), but different with respect to functionality and flexibility.

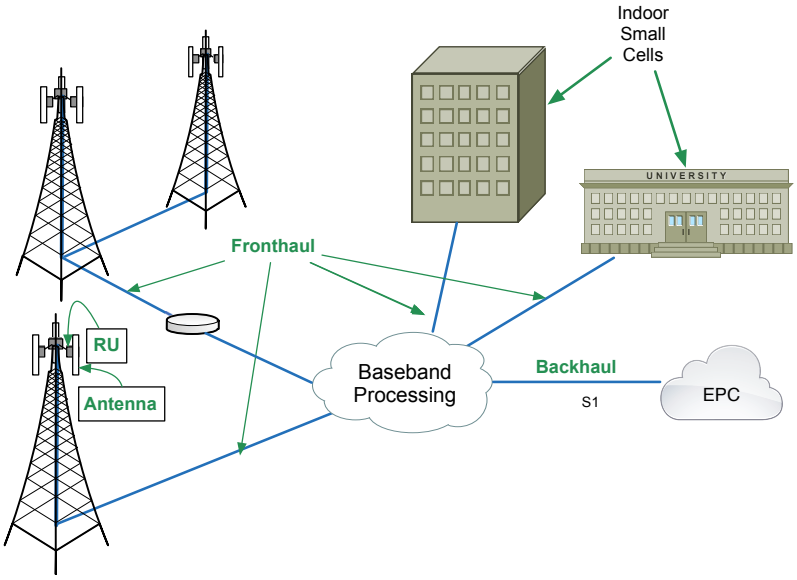


Figure 1.4: Cloud radio access networks (C-RAN).

Aside from increased flexibility in resource allocation, the C-RAN architecture facilitates system upgrade, reduces the footprint of base stations, allows improved energy efficiency and allows better performances on interference management. Due to centralization and elimination of wasteful idle operation of baseband resources, significant power consumption savings are obtained. Regarding interference, the cloud of BBU resources employed in the C-RAN architecture facilitates the implementation of coordinated multi-point (CoMP) techniques. This is because the co-located BBU instances are inter-connected with low latency links (as in data centers), facilitating the exchange of information that is necessary for joint-processing. For example, the work in [5] observed 50% to 100% uplink CoMP gain in field trials. Besides, C-RAN also facilitates other features such as carrier aggregation [16] and improves the performance in other operations such as handover procedures.

In the C-RAN, since BBUs are virtualized, the fronthaul transport networks should be accordingly composed by virtualized links between baseband instances and radio modules. Likewise previous generations, such links should transport digitized signals to and from multiple remote radio units as digitized baseband complex in-phase and quadrature samples (IQ samples), together with synchronization, control and management information. However, particularly in the C-RAN generation, network topologies are likely to demand flexible routing, so that trans-

port over optical transport networks (OTN) and Ethernet networks are of interest rather than solely using dedicated links between baseband units and RRUs [17].

1.2 Current and Future Fronthaul Solutions

There are currently three standard interfaces specified for the fronthaul: the common public radio interface (CPRI) [18]; the open base station architecture initiative (OBSAI) [19]; and the open radio interface (ORI) [20]. The three employ dedicated synchronous time-division multiplexed (TDM) links, typically but not limited to optical fiber [21]. Since synchronous links do not provide the degree of flexibility and cost-efficiency that is sought for C-RAN, novel backwards-compatible solutions are being considered, such as the radio over Ethernet standard being discussed by the IEEE 1904.3 task force[3].

Future fronthaul solutions are required to satisfy conflicting requirements in order to foster C-RAN deployment: they should be low-cost, provide high bandwidth, low delay and flexible routing between BBU pools and RRUs [16]. The low-cost requirement is the main motivation for reusing Ethernet infrastructure, while flexibility calls for packet-based networks. Note, however, that several *fronthauling* options are being considered today, among them microwave links (E-band or V-band) and wavelength-division multiplexing (WDM) schemes with ring networks [5]. The latter, for example, is able to carry several carriers in a single fiber and avoid dedicated point-to-point links. Nonetheless, a critical point addressed in the present work is the objective of leveraging on existing infrastructure for the deployment of flexible fronthaul solutions. Thus, the focus will be pointed to the ubiquitous Ethernet, and its challenges regarding the distribution of timing are going to be thoroughly investigated.

In most countries the majority of facilities are not prepared with optical fiber, so that the deployment of optical links is not a viable solution. It is commonly more cost-effective to bring fiber as close as possible to the facility and link the last remaining indoor portion via copper. This in fact is the particular strategy that maintains digital subscriber lines (DSL) systems with a representative market share in broadband access, since these systems leverage on fiber as close as possible to the user premises and, by doing so, are able to provide high-bandwidth links via twisted-pair transmission with reduced distances [22]. This work is motivated by the possibility

[3]Radio over Ethernet solutions could either consider the encapsulation of time-division multiplexing fronthaul solutions such as CPRI or the direct transportation of radio data over flexible packet-switched networks [17].

of a similar scenario, but for fronthaul. More specifically, it considers synchronous fiber links can be brought as close as possible to indoor installations and then, copper-based Ethernet can provide cost-effective links to indoor small cell radio units using existing infrastructure, e.g. the omnipresent twisted-pair cabling such as Cat4 and Cat5 in buildings.

The problem is that Ethernet networks are severely limited for the fronthaul with respect to synchronization and timing. Synchronous networks facilitate compliance with stringent timing and frequency synchronization requirements defined for different RAN technologies, specially for C-RAN architectures [23]. This is because synchronous transmissions allow the delivery of synchronization at the physical layer, for example with synchronization to the clock of the received physical layer symbols. Such a feature is not inherently possible in packet-switched (asynchronous) Eternet networks, unless ad-hoc solutions are introduced.

This work concentrates on the synchronization challenges in Ethernet-based fronthaul networks. A primary goal is to investigate the performance achieved by two synchronization solutions, one for frequency alignment and another for time alignment using the IEEE 1588 precision time protocol (PTP). In this context, the work provides a comprehensive overview of associated technologies and ultimately presents results obtained with an FPGA-based testbed.

1.3 Dissertation Outline

The following summarizes the organization of this work and its main contributions.

1. Chapter 2 introduces the elementary background with respect to synchronization. First, it elucidates the use of elastic buffers and the associated slips incurred in the absence of sufficient synchronization. Then, it contrasts synchronization approaches in asynchronous and synchronous networks. Finally, a great deal of attention is given to the understanding and modeling of clocks, which are essential to comprehend the timestamp based synchronization via IEEE 1588 that is analyzed and tested in the succeeding chapters.

2. Chapter 3 provides an overview of the interfaces involved in the system considered in this work. It first summarizes the fronthaul synchronization requirements and exposes the common public radio interface (CPRI) and the Ethernet interface with the fronthaul. Then, it exposes a literature review about the transport of radio data over Ethernet and a brief discussion about radio encapsulation. Finally, the chapter introduces the IEEE 1588 PTP and practical details about its implementation.

3. Chapter 4 thoroughly describes the two synchronization architectures developed for the testbed analyzed in this work. First, it provides guidelines for the design of the system where such architectures are embedded. Then, describes the simple synchronization method based on the use of elastic buffers. When the limitations of the latter are clarified, the chapter describes the method based on IEEE 1588 PTP. Next the issue of packet delay variation is discussed and several considerations are draw to formulate a more robust PTP-based synchronization scheme compliant to legacy Ethernet-based fronthaul.

4. Chapter 5 presents the results obtained with the developed testbed and then provides an analysis of the two solutions with respect to the requirements posed by mobile systems to the fronthaul. Based on the results, the chapter discusses the accuracies that can be achieved in practical scenarios.

5. Chapter 6 presents the conclusions and describes possible future extensions of the work.

1.4 Research Contributions

The contributions of this work fall generically (1) on the analysis of widely adopted synchronization schemes when applied in the context of Ethernet-based fronthaul that transport constant bit-rate and high-bandwidth radio streams; (2) on the details provided for their actual implementation in FPGA, also applicable to ASIC implementations; and (3) on adaptations and a combination of state-of-the-art strategies with the goal of achieving sufficient performance in the aforementioned context of interest.

First of all, this work formalizes the design considerations for radio (i.e. CPRI) encapsulation over Ethernet without introducing significant radio data loss (slips) due to unmatched production and consumption rates. Sequentially, as one of its main objectives, the work contributes with the detailed description about the design of the synchronization architectures, the buffer-based method and the PTP-based approach. Specially for PTP, this works contributes with an analysis of the combination between miscellaneous strategies against the practical difficulties originated by PDV in packet-based fronthaul networks. Finally, through measurements obtained with the testbed, the work discusses practical performance capabilities and limitations of such designs towards reliable provision of Ethernet transport in the fronthaul.

Chapter 2

Fundamentals of Synchronization in Telecommunications Networks

The previous chapter provided an overview of the evolution towards the C-RAN paradigm. It was highlighted that in this architecture, multiple centralized baseband units (BBUs) must be flexibly interconnected to multiple physically separated remote radio units (RRUs). Accordingly, it was argued that fronthaul transport networks must be made flexible and Ethernet packet networks can be an appealing solution. However, it was also stated that packet networks are inherently limited with respect to the distribution of synchronization.

This chapter aims to advance towards the understanding of synchronization solutions that can be adopted in packet-based fronthaul networks. The discussion aims to support the analysis of the architectures developed and evaluated in this work.

The chapter is divided as follows: Section 2.1 introduces the three categories of synchronization and their role in telecommunications networks. Section 2.2 introduces slip buffers, a central concept in this work, contextualized by a brief overview of the clock and data recovery mechanism. Section 2.3 clarifies the difference between synchronous and asynchronous communications with respect to synchronization. Section 2.4 provides a mathematical modeling and characterization of clock units. Finally, Section 2.5 introduces packet-based synchronization.

2.1 Definitions and the Role of Synchronization

Synchronization, if not further specified, can imply different forms of alignment between devices connected to network: in frequency, phase or time. For example, [24] defines synchro-

nization as “the ability to transfer either time, frequency or phase from one system, or clock, to another system”. Most of the jargon and definitions of synchronization in telecommunications networks are specified in [25]. An important definition for the discussion in the sequel is the one of *clock*, which in the above recommendation is defined as an equipment that provides a timing signal. Note, however, that it is common to find the term *clock* with different interrelated meanings. For example, it may also refer to the periodic signal that coordinates significant instants (e.g. rising edges) in digital circuits, or it may be used for the physical device itself that generates a clock signal (e.g. a crystal oscillator) in an electronic circuit, and finally, it can also refer to a category of network device in a hierarchical distribution of synchronization, where concepts such as *master clock* and *slave clock* exist. In this dissertation, be aware that these different connotations are inevitably used, but context should clearly distinguish them.

The concept of frequency synchronization is related to the alignment in the rate of significant instants at distinct networked devices. Since in many cases a single clock signal (producing significant instants) coordinates many clock domains within a board or circuit, frequency synchronization alone is responsible for aligning the rate in which many different processing tasks occur, such as analog-to-digital or digital-to-analog conversion (ADC or DAC) and modulation. It is achieved when a system is able to discipline its oscillator or provides an alternative scheme to synthesize the same frequency that exists in the reference system.

In telecommunications and, more specifically, mobile systems, frequency synchronization is typically related to the accuracy in the radio-frequency carrier frequency with respect to nominal values, which should comply to regulations and decoding requirements. For example, when receiver and transmitter carriers are unaligned, significant distortion can be introduced in the system, since a frequency mismatch creates a constant increasing or decreasing phase error that impacts coherent demodulation. Specifically in the case of LTE or more generally OFDM systems, which are very sensitive to carrier frequency offset (CFO), a frequency mismatch causes the loss of orthogonality between subcarriers, which implies the information in a given subchannel leaks to other subchannels, a phenomenon known as inter-carrier interference [26].

The second synchronization category is phase synchronization, which occurs when clocks in distinct networked devices produce significant instants at the same instant, not just the same rate. This imply, for example, that at the same instant both a baseband and a phase synchronized radio unit are feeding a sample in their digital-to-analog converters (DAC). Clearly, phase synchronization is an extension to frequency alignment, as highlighted in Fig. 2.1.

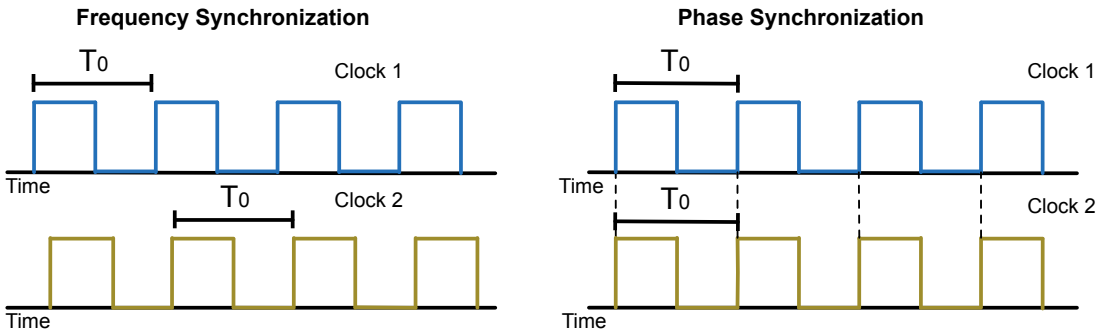


Figure 2.1: Comparison between frequency and phase synchronization.

In mobile systems, phase synchronization is typically related to requirements posed by time-division duplexing (TDD) schemes, in which uplink and downlink slots from adjacent cells are required not overlap to avoid interference [23]. It also supports the stringent requirements posed by collaborative radio techniques, such as eICIC (enhanced intercell interference coordination), joint-transmission (also known as multicell MU-MIMO) and beamforming. For example, when distinct remote radio heads (serving separate cells) irradiate signals that were jointly processed, the effectiveness of these methods strongly depend on the fact that these signals are emitted simultaneously with high accuracy.

The third category is time synchronization, which is closely related to phase synchronization. By definition, in time synchronization not only the significant instants of the devices should be aligned, but also they should correspond to the same absolute *time of day* (ToD) value (e.g. a value in coordinated universal time - UTC). In practice, time synchronization implies that the devices are equipped with modules that keep track of the absolute time, namely with real-time clocks (RTC), and that these modules internally hold the exact same ToD value at every instant, even if the individual local clock signals are not aligned. On specific cases, such as telecom networks, the ToD time series is also used to synthesize a phase-aligned local clock signal, so that time synchronization can be used to achieve phase synchronization [23].

Many networks require the use of timestamps (ToD samples), for example to track the time of requests to databases, or to coordinate the sensors and actuators in an industrial network. Some applications require very high accuracies such as sub-nanosecond precision. One example is the White Rabbit project [27], conceived to satisfy the requirements of control systems adopted in the particle accelerators researched by the European Organization for Nuclear Research (CERN). In telecommunications networks, several features require time synchroniza-

tion, such as billing or monitoring of events for management purposes. In this context, time synchronization allows different devices to generate timestamps aligned to the same time scale.

In general, frequency and phase alignment are related to requirements mandated by the physical layer, while timing synchronization is related to higher-layer features. Nonetheless, the fact that time-of-day information can be exchanged between devices creates further possibilities of providing phase and frequency synchronization. This is the idea behind synchronization through the precision time protocol (PTP), whose details are postponed to the next chapter.

2.2 CDR Circuits and Slip Buffers

In the previous section, one aspect of frequency synchronization was purposely not mentioned. It is one related to the concept of slip buffers, which deserves closer attention in this work. Consider the synchronous data transfer represented in Fig. 2.2. Neglecting all other details in this figure (later explained), for now it suffices to observe that the transmitter uses its own local clock (Tx Local Clock) to transmit a signal, that the receiver recovers the transmit clock through a mechanism known as clock and data recovery (CDR) and that it stores the incoming data stream into a circular buffer known as *slip buffer*. Then, the receiver outputs the data stream for any downstream processing with its own local clock (Rx Local Clock).

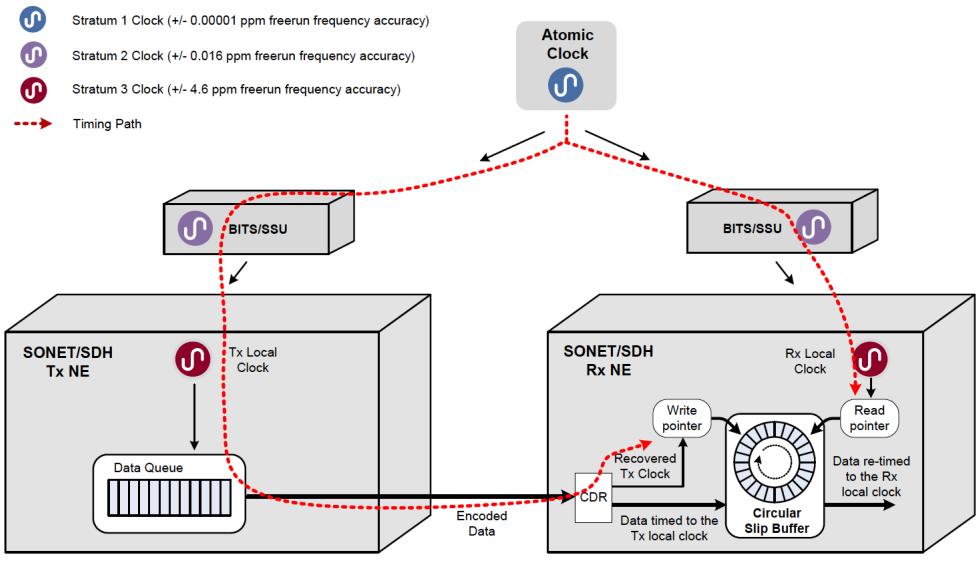


Figure 2.2: Synchronous data transfer. Taken from [1].

First of all, the receiver is required to extract the Tx clock information from the received

signal and use this clock for any subsequent processing before the slip buffer, including demodulation and detection. Since the waveforms are dispersed during transmission, the receiver must also remove the accumulated jitter in a process known as "data retiming", where "jittered" data is re-aligned at the edges of the clock, by passing through e.g. a flip-flop. Both process combined are known as clock and data recovery (CDR) [28].

A typical CDR circuit contains three main components. The first component is the phase detector, which should produce a signal that can be mapped into the phase difference between the clock and the data. As shown in [28], one example is a D flip-flop (DFF) with an inverted connection, namely with the data signal connected to the clock input and with the clock signal at the data input. Such a scheme uses the data to sample the clock. Thus, for example, if the data lags the clock, all samples of the clock signal will be at low voltage, while in the contrary case (data leading the clock), the samples will all be high voltage states of the clock square wave.

As explained in [28] such a DFF phase detector is equivalent to edge detection[1] on the data signal, followed by multiplication of a reference clock signal. For example, in a pulse amplitude modulation scheme, such as the one used in 1000BASE-T Ethernet (of interest in this work), it generates a train of pulses with a rate matched to the symbol rate or, more precisely, to the shaping pulse rising edges. Then, multiplies this pulse train by a non-synchronized clock signal, producing a "beat" that can be used to infer phase differences.

The other components of a CDR circuit are a lowpass loop filter and a controlled oscillator, typically voltage controlled (VCO), which complete the phase-locked loop (PLL) structure. As a result, the circuit outputs the recovered version of the transmit clock and the data retimed to this clock. A simple CDR circuit is exemplified in Fig. 2.3.

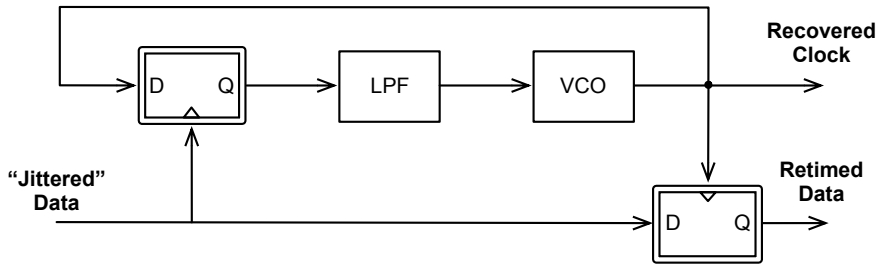


Figure 2.3: Simple clock and data recovery circuit.

Next, in the receiver of Fig. 2.2, the retimed data is written into the so-called *slip buffer*

[1] An edge detector by definition produces short pulses whenever the input signal transitions from low to high voltage or vice-versa.

or *elastic buffer*. This buffer operates between two interfaces, the write and read interfaces, which are clocked with independent frequencies, the recovered transmit clock and the receiver local clock, respectively. It is normally implemented with a dual-port memory, and leads to the First-in First-out (FIFO) structure with separate pointers for read and write operations.

An elastic buffer is used to bridge between clock domains in a circuit[2], so that the two can operate independently. However, the problem comes from the fact that read and write frequencies are normally not perfectly aligned. In this case, assuredly at some point the buffer will become empty or full, depending on the frequency difference [29].

Ideally, the buffer should be occupied with a level that leads to the lowest probability of becoming full or empty, namely at its middle point. When the buffer reaches an empty or full state, several bits can be lost, either because there is not vacant space for being written or because they are not available for reading. This loss of bits is what is known as *slip*, and due to the fact that it occurs with a certain periodicity it is common to find the metric known as *slip rate*. According to [29], based on the write and read frequencies (f_w and f_r) and the buffer size N, it is possible to compute the slip rate:

$$F_{\text{slip}} = \frac{|f_w - f_r|}{N} \text{ slip/s.} \tag{2.1}$$

Slips can cause several problems in higher layer protocols, for example call drops or retransmissions. The problem is particularly important for synchronous communications, where new transmissions are always being received by a particular node, without much time to recover from an abnormal raise or reduction in the buffer occupancy level. Thus, frequency synchronization receives a great deal of attention, to avoid slips and correspondingly bit losses. The next section advances the discussion of slip buffers by presenting a short historical overview and by contrasting synchronous and packet-based communications in terms of synchronization.

2.3 Synchronous vs. Packet-based Synchronization

Slips are a classical problem in digital transport networks, so it is useful to first contextualize the topic among historical solutions.

One of the important marks in the emergence of synchronization in telecom networks is the digitization of voice telephony services. When analog voice signals started to be sampled

[2]In the context of packet-based network protocols, a similar buffer scheme receives the use and denomination of PDV buffer or de-jitter buffer.

and pulse-code modulation (PCM) used to convey the information as bit streams over copper wires, one great advantage was the fact that PCM allowed significantly simpler implementations of time division multiplexing (TDM), thus reducing costs to transport voice over wires. This technology fostered the first all digital voice switches developed around late 1960 and early 1970 [30], so the public switch telephone networks (PSTN) began their transition to digital. Nevertheless, the growth of telephony services introduced a strong need for efficient switching schemes that would be capable of solving the consequences of the fact that signals from different systems being aggregated (multiplexed) into a switch were not synchronized.

At that time, as revealed in expression (2.1), it was seen that an obvious treatment to the slip in a clock-domain conversion buffer (also known as "elastic buffer") is to increase the buffer size to reduce the slip rate, but that would increase costs, add more latency to system and doesn't actually solve the problem (slips would still occur). Alternatively, a better solution would be to align the clock frequencies of the nodes in the network to a common source, but that would add significant costs, since it would require the distribution of synchronization via dedicated networks or interfaces within a network (like it is done today). In this context, the plesiosynchronous (*almost synchronous*) digital hierarchy (PDH) introduced by ITU-T G.702 [31] emerged as a cost effective solution. The architecture consisted of a TDM digital transport scheme in which the misalignment between clock frequencies of different nodes are accommodated by positive or negative pulse stuffing.

The biggest feature brought by the PDH technique is that it does not require the actual synchronization between the network devices. Instead, it allows relatively large frequency offsets and avoids slips by using pulse stuffing [32]. When, at some point, a time-division slot appears at the PDH multiplexer and the *tributary*[3] data is not yet available, the multiplexer inserts a bit ("stuffs a pulse") at this position and includes the information about the stuffed-pulse position in the overhead bits. In contrast, when the multiplexer is not able to accommodate the incoming information from certain fast tributaries, it temporarily stores this information (in an elastic storage) and informs the remote endpoint (demultiplexer) about the left-out bits via the overhead bits [32]. Thus, the receiving demultiplexer is capable of properly splitting the incoming signals and can reconstitute the bitstreams that should go to each destination.

The problem with PDH systems was the flexibility regarding the addition of new clients or rates, which required considerable network and equipment redesign. In the succeeding tech-

[3]Multiplexed streams in PDH are often named *tributaries*.

nology, which improved flexibility, the solution adopted to solve buffer slips was the actual distribution of synchronization through a network, in the optical interfaces of the Synchronous Digital Hierarchy (SDH) or the Synchronous Optical Network (SONET). The SDH/SONET transmission structure is the one shown in Fig. 2.2. Note that a timing path exits since the primary reference clock (PRC) Stratum 1 atomic clock (Stratum levels are described in Section 2.4.1), through the building integrated timing supply (BITS) equipments up to the SDH/-SONET receiver. Thus, all clocks in this hierarchical network are *locked* to a common enormously accurate source, the PRC, so that slips are avoided.

A different strategy is adopted for Ethernet based packet networks (IEEE 802.3). Instead of synchronizing the local receiver clocks with high accuracy, both transmit and receive local clocks operate free running (not *locked* to a reference). Then, to avoid slips, sufficiently large buffers are used and flow control is adopted to prevent overflow and underflow problems [1]. Fig. 2.4 illustrates such a packet-based transfer. Note the local clocks are much less accurate (and cheaper) than the ones adopted in the synchronous system of Fig. 2.2.

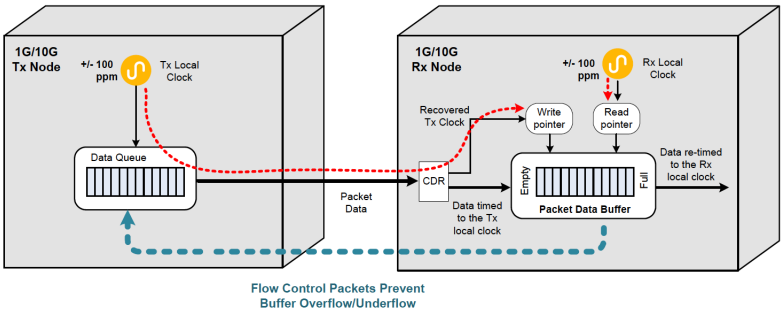


Figure 2.4: Packet-based data transfer. Taken from [1].

For a long time, early methods for distributing synchronization over the physical layer such as SDH were the preferred schemes for telecom transport networks. However, with the widespread adoption of all-IP best-effort networks in the telecom infrastructure, such as Ethernet over IP, alternatives to theses schemes gained attention. For backwards compatibility, one important issue is to allow an asynchronous transport network to be inserted within a synchronous hierarchy. Naturally, in order not to break a timing path such as the one shown in Fig. 2.2, the modules placed in the edges or interface between asynchronous and synchronous networks shall align their local clocks through a packet based method [1].

The aforementioned challenge is exactly the context of this work. More specifically, our

goal is to evaluate the feasibility of certain synchronization architectures that allow the edge equipments of a packet-based legacy Ethernet network to synchronize with the accuracy required to interface with synchronous fronthaul links. In the end of this chapter, after introducing a mathematical derivation of clocks, packet-based synchronization solutions are introduced.

2.4 Clock Model

In the beginning of Section 2.1, several commonly used connotations of *clock* were introduced. In this section, the goal is to elaborate on one of the definitions of clock, to model it and understand the commonly used metrics for its characterization. Before presenting the definition of interest here, however, it is useful to further discuss two of the aforementioned definitions of clock. This is because the two are strongly interconnected, and understanding of their relation helps to clarify the discussion.

The first definition of interest is the one that defines *clock* as the periodic electrical signal that coordinates significant instants (e.g. rising edges) in synchronous digital circuits, better distinguished if referred to as *clock signal*. In every clock implementation, a periodic electric signal comes from a physical device such as a crystal oscillator, which produces approximately a sine-wave voltage modeled as:

$$V(t) = V_0 \sin \Phi(t), \tag{2.2}$$

where V_0 is an amplitude assumed constant[4] and $\Phi(t)$ is the total time-dependent accumulated phase that can be assumed without loss of generality to start from arbitrary origin $\Phi(t = 0) = 0$, as in [34]. This oscillatory signal $V(t)$ is typically processed through phased-locked-loops (PLL) and clock buffers [35] that are used to generate the actual *clock signal* driving synchronous units in a digital circuit, which becomes a square-wave signal.

The second definition of interest is the one that defines *clock* as the module (in hardware or in software) capable of keeping track of time, which can be better recognized if referred to as *clock module* or *clock unit*. It is important, at this point, to understand that such a module is fed by the aforementioned clock signal (first definition). According to [36] and in compliance to the second definition, "a clock consists of an oscillator and a counter", where the oscillator (or the square-wave signal from the first clock definition) drives the increments performed in the latter in order to keep track of time.

[4]Most of the literature also considers amplitude fluctuations, as in [33], but we neglect for the sake of simplicity.

A *clock* of the second definition can be implemented directly in hardware or in software. When implemented as a standalone application-specific integrated circuit (ASIC) or as a co-processor, it is known as real-time-clock (RTC), a module that outputs ToD information synchronously. Regardless of the implementation, it is an unit capable of generating a time scale when fed with a periodic physical or software-based[5] signal coming directly or indirectly from an oscillator. At every significant instant (e.g. on raising edges), its counter is incremented to denote that time has passed, so that a time scale is generated.

A clock can be characterized in many aspects. Usually its time scale is compared to a reference "master" scale or an "ideal" (perfect) time scale. The comparison is useful because the slave's time scale constantly accumulates errors with respect to the reference, and the phenomena causing errors can be inferred from their interpretation. The following subsections introduce these interpretations and discuss the common clock impairments and characterizations.

2.4.1 Frequency and Time Offset

Oscillators do not oscillate at exactly the nominal frequency, since their behaviors are affected by several factors, more or less depending on their quality. For this reason, one of the most useful figures of metric for assessing the quality of a clock synchronization scheme is the fractional frequency offset[6], which is dimensionless and defined as:

$$y(t) = \frac{f(t) - f_{\text{nom}}}{f_{\text{nom}}}, \tag{2.3}$$

where $f(t)$ is the clock frequency at a particular instant t and f_{nom} is its nominal value. Alternatively, in many cases the offset is not compared to the nominal value, but to a reference clock frequency $f_{\text{ref}}(t)$ that can be time-varying, as follows:

$$y(t) = \frac{f(t) - f_{\text{ref}}(t)}{f_{\text{ref}}(t)}. \tag{2.4}$$

Some models often also assume that the reference does not vary with time, on the grounds that the reference is a very stable clock. Nonetheless, the time-varying $f_{\text{ref}}(t)$ is more general and gives a clear indication that the goal is to constantly synchronize a slave clock frequency $f(t)$ to the reference (e.g. a grandmaster), even if the latter deviates from the nominal value.

[5]The term "signal" with respect to software is just adopted for the sake of the explanation. In software, these periodic assertions that increment a counter are more commonly known as *ticks*.

[6]The fractional frequency offset is also known as normalized frequency offset or frequency deviation.

Commonly oscillators are specified in terms of the peak-to-peak frequency offset they are expected to operate in free-run mode, namely when not locked to a frequency synchronization reference. For example, in the United States, the ANSI TI.101 specifies the so-called *Stratum levels*, categories of clock source quality in terms of frequency offset and other parameters beyond the scope of this work. According to the standard, *Stratum 1* has the highest level of frequency accuracy, thus it is adopted as the PRC in a hierarchical frequency distribution network. Its frequency accuracy shall be in the order of ± 0.01 ppb, a figure achieved by atomic clocks. *Stratum 2* has the second highest quality, with an accuracy of ± 16 ppb, while *Stratum 3* operate with ± 4.6 ppm when free-running. Nonetheless, when all clocks in the network are operating in *normal mode* (c.f. [29]), namely synchronized to the PRC, they all take the accuracy of the the Stratum 1 clock. This is the objective of an hierarchy such as the one in Fig. 2.2.

The other important figure of merit is the time error function (time offset) between a clock generating time $T(t)$ and an ideal reference time $T_{ref}(t)$, which in the ideal case is equal to the real time t. In [25], it is defined as:

$$x(t) = T(t) - T_{\text{ref}}(t) = T(t) - t \tag{2.5}$$

From this function, many derivate metrics are often computed to assess the clock quality. Common metrics are the time interval error (TIE), which compares the difference between the measurements of a given time interval taken by two different entities (usually measured by how far each active edge of the clock varies from its ideal position), the maximum time interval error (MTIE) and the Allan Variance [37]. The last two are beyond the scope of this work.

2.4.2 Practical Observations about Time Scales

The time scale $T(t)$ in (2.5) depends on the definition of clock. From the definition that considers clock as the combination of an oscillator and a counter (labeled as the second definition in this section), the time scale can only increase in quantized steps. In contrast, when clock is the actual signal generated by the oscillator, the time scale $T(t)$ is a continuous function. The difference with respect to the observation point is illustrated in Fig. 2.5. Without loss of generality, the continuous $T(t)$ is adopted in the derivations that follow.

Nevertheless, it is important to realize that the continuous time scale generated by the clock signal is an abstraction, while the time scale generated by the counter is tangible, for example used by a timestamping unit in hardware. Furthermore, the staircase time scale always

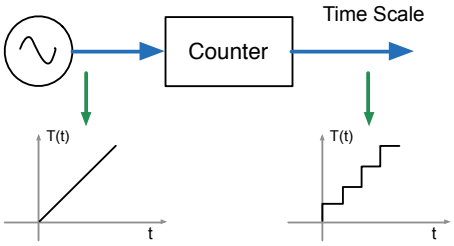


Figure 2.5: Time scales according to observation point.

depends on the arbitrary mapping adopted between the count value and the time. In practice, a counter is designed to keep track of a given accuracy, for example nanoseconds. Then, it is incremented by a number that is mapped directly to nanoseconds and corresponds to the nominal period of its driving clock signal, known by the designer. For example, let's suppose a nominal clock of 100 MHz should drive a nanosecond counter. Then, the appropriate increment number in the counter to be added in each period is 10, since the clock period is 10 ns.

Similarly, a continuous time scale $T(t)$ is associated with the phase of its clock signal. Just like each phase change of 2π in the minute hand of a watch corresponds to 60 minutes by convention, the phase of the clock signal is translated by some "rule" into time units. With respect to Fig. 2.5, it is interesting to observe that the time scale before the counter "maps" the phase of the signal in any instant to a time value (again, an abstraction), while the time scale after the counter can only track 2π revolutions in the signal that drives the counter increments.

In this context, a time function (time measured by a clock) can be defined as in [25]:

$$T(t) = \frac{\Phi(t)}{2\pi f_{\text{nom}}}. \tag{2.6}$$

The above expression comes from the fact that the time-derivative of the total phase of a sinusoid at a particular instant t relates to its instantaneous frequency by the following:

$$\omega(t) = 2\pi f(t) = \frac{d\Phi(t)}{dt}, \tag{2.7}$$

where $\omega(t)$ is the time-dependent angular frequency.

Thus, it follows that the total phase can be expressed as:

$$\Phi(t) = 2\pi \int_{-\infty}^{t} f(\tau) d\tau. \tag{2.8}$$

Note that by assuming a constant $f(t) = f_{\text{nom}}$ for $t \geq 0$ (a clock frequency that exists only for $t > 0$), one obtains (2.6).

A final observation regarding time scales is that even though the staircase time scale is limited by quantized increments, the actual increment value can be varied. This is the way that frequency synchronization is generally implemented in practice when real time clocks are employed. For example, in the previous nanosecond counter driven by a 100 MHz clock signal, if the actual clock driving the counter becomes higher than the nominal value (more increments during the same timeframe), the increment value can be reduced as a compensation. Ultimately, by adjusting the increment value a counter continues to keep track of time (ideally with enough precision) and other clocks can be generated by comparing the counter value to a specific value.

In practice, packet-based synchronization algorithms such as the network time protocol (NTP) and PTP employ counters (with staircase timescales) and periodic processing of timestamps taken from them. In these systems, typically the processing period is designed with a small enough value in which the clock signal frequency can be assumed stable. Thus, a discrete model of (2.8) assumes piece-wise constant values for $f(t)$, yielding:

$$\Phi_k = 2\pi \sum_{m=0}^{k} f_m \Delta T_m, \tag{2.9}$$

where the subscript k denotes evaluation at the k-th processing cycle and $\Delta T_k = T_k - T_{k-1}$. Moreover, to reinforce the observation above, note that not only f_k can be varying with time, but also the "step" of the counter ΔT_k can be tuned to compensate frequency mismatches.

In attempt to isolate the k-th time value T_k, (2.9) can be rewritten as:

$$\Phi_k = 2\pi \left[\sum_{m=0}^{k-1} f_m \Delta T_m + f_k \left(T_k - T_{k-1} \right) \right],$$

so that

$$T_k = T_{k-1} + \frac{\Phi_k}{2\pi f_k} - \frac{1}{f_k} \sum_{m=0}^{k-1} f_m \Delta T_m. \tag{2.10}$$

Finally, note again that assuming $T_0 = 0$ and a constant $f_k = f_{\text{nom}}$ for all k, similar to (2.8), the above reduces to the discrete version of (2.6):

$$T_k = \frac{\Phi_k}{2\pi f_{\text{nom}}}. \tag{2.11}$$

2.4.3 Instantaneous Phase Model

The previous expressions were based on the general total time-dependent instantaneous phase $\Phi(t)$. Henceforth, however, models of this instantaneous phase will be presented and

used to analyze clock characteristics. A first simplistic approach neglects any assumption about the clock frequency and models the total phase as the sum between a component due to the nominal frequency and a noise term, as follows:

$$\Phi(t) = 2\pi f_{\text{nom}} t + \phi(t), \tag{2.12}$$

where $\phi(t)$ is the so-called phase noise or residual phase [37], a random process that includes the deviations with respect to the nominal frequency f_{nom} due to noise inherent to the elements processing or generating the clock (e.g. the oscillator itself, a voltage controlled oscillator or a phase detector) [38]. The latter is usually modeled as a stationary random process and better understood using statistics and frequency domain analysis (using e.g. power spectral density). A thorough treatment in this regard is presented in the seminal works of David W. Allan [39] and James A. Barnes et.al. [33], but only a brief introduction is given in Section 2.4.5.

Then, the expression in (2.7) becomes:

$$\omega(t) = \frac{d}{dt}\left[2\pi f_{\text{nom}} t + \phi(t)\right], \tag{2.13}$$

so that the instantaneous frequency can be expressed as:

$$f(t) = \frac{\omega(t)}{2\pi} = f_{\text{nom}} + \frac{1}{2\pi}\frac{d\phi(t)}{dt}. \tag{2.14}$$

2.4.4 Fractional Frequency Offset Interpretation

The definition in (2.14) allows an insightful interpretation of the fractional frequency offset. First, substituting the above frequency expression in (2.3), yields:

$$y(t) = \frac{1}{2\pi f_{\text{nom}}}\frac{d\phi(t)}{dt}. \tag{2.15}$$

Then, inspection of (2.15) reveals that the numerator corresponds to an undesirable infinitesimal change in phase (in radians) due to noise within an infinitesimal interval of time, while the term $2\pi f_{\text{nom}}$ in the denominator corresponds to the desirable infinitesimal phase change due to the nominal angular frequency (also in radians) within the same interval. Hence, the fractional frequency offset can be interpreted as the ratio between the phase error accumulated over a given period and the nominal phase change in this period.

Another interpretation can be stated by observing that a phase change has an associated time deviation. By substituting (2.12) in (2.6), one obtains:

$$T(t) = t + \frac{\phi(t)}{2\pi f_{\text{nom}}}. \tag{2.16}$$

Then, by substituting the time offset definition in (2.5), yields:

$$x(t) = \frac{\phi(t)}{2\pi f_{\text{nom}}}. \tag{2.17}$$

Ultimately, the time-derivative of (2.17), when compared to (2.15), reveals the relation between the fractional frequency offset and the time offset function $x(t)$:

$$y(t) = \frac{dx}{dt}. \tag{2.18}$$

In another words, the fractional frequency offset corresponds to the time offset or error accumulated over a given period of time with respect to the reference.

From (2.18), it can also be seen that the fractional frequency offset can be obtained by observing the time offset function. In fact, practical implementations such as PTP estimate the frequency offset by using periodic measurements of the time offset. In such cases, a discrete model follows as an approximation to (2.18):

$$y_k = \frac{x_k - x_{k-1}}{\Delta t}, \tag{2.19}$$

where x_k and y_k are the k-th time-offset measurement and frequency offset estimation, respectively. The interval Δt, in particular, is chosen by design to allow a reasonable approximation.

Finally, note that in the literature, typically when the figure of merit $y(t)$ is presented in "part-per notation" (e.g. ppm or ppb), it is referred to as fractional frequency offset (as adopted in this section). However, when it is presented in terms of the time offset accumulated per second, it is also referred to as the *clock drift* (not the same as *frequency drift rate*) [24].

2.4.5 Wander, Jitter and Phase Noise

The model in (2.12) highlights the fact that the revolutions in the phase of the sinusoidal signal have some randomness in their periods over time. Hence, the model tacitly says that the significant instants coordinated by such a clock in a digital circuit deviates from their ideal instants or periods. These variations are loosely known as jitter, but also commonly classified into two broad groups: long-term variations and short-term variations. The former is known as *wander*, and refers to variations in the significant instants that occur at frequencies (offsets from nominal values) less than 10 Hz, while the latter is strictly known as timing jitter, and comes from variations with frequencies higher than 10 Hz.

Wander can be caused by frequency offsets between a measured clock and a reference, or by changes in their relative phase due to temperature variations. It can also be originated by

very low-frequency phase noise in a clock oscillator [40]. As a consequence, wander leads to cumulative time or phase errors and at some point causes slip buffers to fill or empty [29]. In contrast, jitter introduces eye-closure in the horizontal axis of an eye-diagram, thus reduces the noise margin in a system and introduces distortion. As a result, demodulation performance can be reduced, namely a penalty on the probability of bit errors.

The concept of jitter is more tightly coupled to time domain measurement or analysis of a clock signal, and just like all the other characterizations in this section, it is a relative measure. For example, it is commonly measured using root-mean-square (RMS) or peak-to-peak values of the timing errors with respect to the reference clock signal (e.g. a trigger source in an oscilloscope). Although it is a time domain effect, the jitter phenomenon comes from the phase noise term present in the model of (2.12). The latter modulates the phase of a sinusoid and, in contrast to jitter, is more commonly analyzed in the frequency domain.

A commonly used figure for frequency analysis is the so-called $\mathcal{L}(f)$ ratio, known as the single sideband phase noise referenced to carrier. It is computed as the ratio between the single sideband power at a particular 1 Hz band offset from the nominal frequency and the total signal power, and is used to define the spectral purity of the signal. It can be shown that this ratio approximately obeys the following expression [38]:

$$\mathcal{L}(f) \approx \frac{S_\phi(f)}{2}, \tag{2.20}$$

where $S_\phi(f)$ is the spectral density of the phase noise.

By observing $\mathcal{L}(f)$, it is possible to infer certain characteristics of the clock, for example the type of noise that is dominant. The power-law model [41] is widely adopted for this purpose. It reveals the composition of the phase noise effects from white phase noise, fickler noise, white frequency noise, flicker frequency noise or random-walk frequency noise. Nonetheless, inspection at this level is beyond the scope of this work.

When analyzing jitter in the frequency domain, it is clear that this mis-timing source is easier to filter than wander [36]. This is because wander introduces a deviation that is very close to the actual frequency of interest. Thus, in a PLL scheme that is normally used to attenuate mis-timing, very sharp and narrow band loop filters would be required. However, filters with narrow bandwidth present much slower responses (higher lock times), which are undesirable.

2.4.6 Linearly Drifting Noisy Clock Model

The model in (2.12) separates the nominal component from the noise component. In the sequel, the goal is to derive a model compliant to one commonly presented in the literature (see [34,36,42]), more specifically the model defined by the ITU-T *definitions and terminology for synchronization networks* [25]. The model assumes most variations present in oscillators, and different to the previous generic model, has assumptions about the clock frequency.

One of the assumptions of the following model is that the frequency generated by the oscillators drifts in the long-term, due to temperature variations that cause thermal expansion or contractions in their physical dimensions. The metric known as *fractional frequency drift* (or *relative drift rate*) models how the frequency offset changes with time due oscillator aging effects [25] and environmentally induced effects [36]. By definition[7], it is the rate of change of the fractional frequency offset with time (in units of $1/s$), given by:

$$D(t) = \frac{dy(t)}{dt}. \tag{2.21}$$

Furthermore, the model assumes the frequency drift is linear, which is generally a reasonable approximation for quartz crystal oscillators [29], but not for all oscillators [34]. A linear fractional frequency drift implies the rate of change in the frequency offset has constant slope. Thus, the instantaneous frequency of a linearly drifting clock is modeled deterministically as:

$$f(t) = f_{\text{nom}}(1 + Dt) + \varepsilon_0, \tag{2.22}$$

where ε_0 is a deterministic term representing a constant error (offset) relative to the nominal frequency (also known as syntonization error[8]) and D is the linear fractional frequency drift.

By substituting (2.22) in the fractional frequency offset definition of (2.3), yields:

$$y(t) = \frac{f_{\text{nom}}(1 + Dt) + \varepsilon_0 - f_{\text{nom}}}{f_{\text{nom}}} = Dt + \frac{\varepsilon_0}{f_{\text{nom}}}. \tag{2.23}$$

Furthermore, note that the right-most term is in its own a constant normalized frequency offset y_0, so that the expression can be alternatively given as:

$$y(t) = Dt + y_0. \tag{2.24}$$

[7]The fractional frequency offset is commonly referred to as simply *drift* in the literature. However, note that *drift* is not the same as the *fractional frequency drift*. The latter is the time derivative of the former.

[8]Practically sometimes this deterministic offset term comes from the so-called "settability" of the clock [25], namely the ability to precisely "set" the clock frequency to a given value.

Moreover, when the time-varying clock frequency is assumed to follow a deterministic model, the total instantaneous phase in (2.8) is better understood by the following (c.f. [42]) expression:

$$\Phi(t) = \Phi_0 + 2\pi \int_0^t f(\tau)d\tau + \phi(t), \tag{2.25}$$

where Φ_0 is the initial phase offset (previously assumed zero due to the lower integration limit from $-\infty$) and $\phi(t)$ is the phase noise. In contrast to (2.8), the above model has $f(t)$ explicitly.

By replacing the frequency model of (2.22) in (2.25), yields:

$$\begin{aligned}\Phi(t) &= \Phi_0 + 2\pi \int_0^t \left[f_{\text{nom}}(1 + D\tau) + \varepsilon_0\right] d\tau + \phi(t) \\ &= \Phi_0 + 2\pi f_{\text{nom}} t + 2\pi f_{\text{nom}} D\frac{t^2}{2} + 2\pi\varepsilon_0 t + \phi(t)\end{aligned}$$

Then, by substituting $\varepsilon_0 = f_{\text{nom}}\left(\frac{\varepsilon_0}{f_{\text{nom}}}\right) = f_{\text{nom}} y_0$, the expression reduces to:

$$\Phi(t) = \Phi_0 + 2\pi f_{\text{nom}}(1 + y_0)t + \pi f_{\text{nom}} D t^2 + \phi(t), \tag{2.26}$$

which is exactly the total instantaneous phase model presented in [25].

Next, from (2.6) the total phase in (2.26) can be mapped to the following time scale:

$$T(t) = T_0 + (1 + y_0)t + \frac{D}{2}t^2 + \frac{\phi(t)}{2\pi f_{\text{nom}}} \tag{2.27}$$

where T_0 is the initial time of the scale, given by $T_0 = \frac{\Phi_0}{2\pi f_{\text{nom}}}$.

Finally, by the definition of the time offset function in (2.5), the instantaneous time offset can be modeled as:

$$x(t) = x_0 + y_0 t + \frac{D}{2}t^2 + \frac{\phi(t)}{2\pi f_{\text{nom}}}. \tag{2.28}$$

where x_0 is the initial time offset of the scale, $x_0 = T_0$.

The model in (2.28) implicitly indicates the concepts and categories of deviations of a clock from the ideal. For example, note that the frequency drift introduces a quadratic term in the time error function, reason why it is often the predominant cause of time deviation [34]. Note also that the initial time offset x_0, the constant frequency offset y_0, the relative drift rate D and the phase noise $\phi(t)$ all cause the clock time scale to offset from the reference.

2.5 Packet Network Synchronization

Section 2.3 highlighted the synchronization issue that arises with the insertion of a packet network within a synchronous TDM network. To summarize, the problem is that Ethernet based networks operate using free-running clocks with inferior accuracy (i.e. ± 100 ppm), while synchronous networks rigorously distribute the synchronization through a hierarchical timing path such as the one shown in Fig. 2.2. Thus, a packet network breaks the synchronization distribution chain when placed amidst TDM networks, potentially preventing radio units from synchronizing with base stations.

One natural way to solve this is by providing a hierarchical frequency distribution path (known as *line timing*) throughout an asynchronous network. This is the approach of the Synchronous Ethernet (SyncE). In SyncE, instead of a single ± 100 ppm free-runing clock, every node in the chain is equipped with a Stratum 3 clock (± 4.6 ppm free-running) and a PLL that locks to a PRC Stratum 1 clock using as reference the clock received through the physical layer. Then, an edge SyncE node can seamlessly transfer data towards a TDM node, namely SyncE is interoperable with SONET/SDH.

However, one downside of SyncE is that it requires all the intermediate nodes to be capable of recovering and re-transmitting frequency synchronization. Thus, legacy Ethernet equipment are not suitable and for most small cell fronthaul deployment scenarios it would require significant network upgrade. Another disadvantage is that it only provides frequency synchronization, but not phase or time synchronization.

In contrast to SyncE, the IEEE 1588 PTP provides time synchronization and only requires endpoint equipment to contain its capabilities. The approach of PTP is to periodically extract time of day information from local RTCs and exchange this information through regular asynchronous frames. With the timestamps, the communicating endpoint equipments are able to estimate time and frequency offsets, then generate a time-aligned clock signal. The downside is that it relies on estimations that are severely affected by packet delay variation (PDV).

The network time protocol (NTP) provides a similar solution. However, NTP equipments typically do not employ hardware timestamps, instead are purely software-based [43]. Thus, NTP generally achieves less accuracy. Besides, in contrast to PTP, NTP does not have provisions for mitigating the effects of PDV along a network.

ntext should clearly distinguish them.

Chapter 3

System Overview

In order to enhance the flexibility of the C-RAN fronthaul and foster small cell fronthaul deployments, in some cases TDM fronthaul solutions such as the Common Public Radio Interface (CPRI) may need to be replaced or adapted to packet-switched networks. The system considered in this work considers such a case of fronthaul deployment over legacy Ethernet networks available in most buildings. Thus, at the BBU side, it is assumed that the system interfaces with backhaul delivering Stratum 1 traceable clocks. On the RRU side, an adaptation layer must recover synchronization via a packet-based method and interface with a legacy CPRI unit. Finally, the Ethernet network should transport encapsulated radio data. Fig. 3.1 is a system overview that also highlights the scope of the testbed developed in this work.

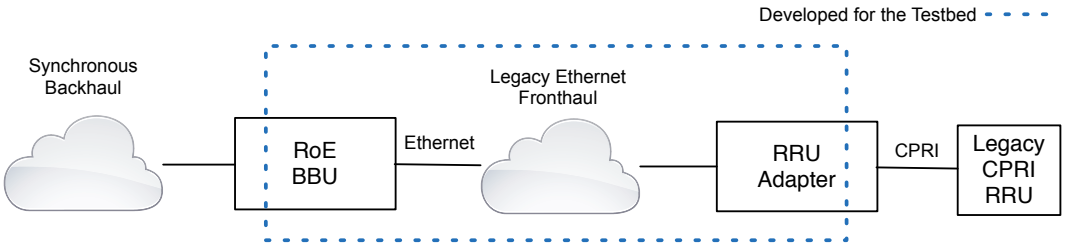


Figure 3.1: System overview.

This chapter provides an overview of the interfaces present in the system of Fig. 3.1. Section 3.1 starts by presenting the requirements arising at the fronthaul interface and the corresponding requirements at the air interface. Section 3.2 introduces the CPRI interface. Next, Section 3.3 exposes brief considerations about the Ethernet interface. Section 3.4 discusses the system transmission and encapsulation of CPRI or radio data over Ethernet. Finally, Section 3.5 details the IEEE 1588 precision time protocol (PTP).

3.1 Fronthaul Requirements

The transport of radio data over a fronthaul interface has to satisfy several requirements specified for radio (e.g. LTE) transmissions. In this section, the frequency alignment, phase alignment and fronthaul delay requirements are analyzed, so that the objectives that should be satisfied by the system architecture of interest in this work can be formulated.

3.1.1 Frequency Alignment

The LTE technical specification in [44] defines the allowed frequency offset for the base station (BS) transmit frequency. The requirements are given according to the BS class, as shown in Table 3.1. Moreover, the requirements are posed to the air interface and are related to the OFDM subcarrier spacing. In contrast, at the input to base stations (BBUs and RRUs), due to noise and holdover[1] budget, a more tight ± 16 ppb is normally required [24].

Table 3.1: Maximum frequency error requirement for LTE.

BS class	Accuracy (ppb)
Wide Area BS	50
Local Area BS	100
Home BS	250

In comparison to LTE, CPRI requires an even higher accuracy of 2 ppb per link. However, as indicated in [45], this error should be satisfied at the output of the CPRI link, while LTE specifications must be satisfied at the RRU's local clock. This difference comes from the fact that several CPRI links can be used from BBU to RRU.

3.1.2 Time and Frame Alignment

Time alignment error (TAE) is defined by 3GPP as the largest relative timing difference between any two signals, e.g. signals between macro and small cell or coordinated radio units. In order to ensure the quality of radio signals, several different TAE requirements are defined by 3GPP for different applications. The tightest requirement is for MIMO or Tx diversity transmissions, in which at each carrier frequency TAE must not exceed 65 ns [44].

[1] Holdover is the clock operation mode in which it is not locked to an external reference, but maintains its accuracy with respect to the last known frequency comparison with a synchronization reference.

As pointed in [26], the difference in radio frame start timing is designed to ensure that the combination of synchronization error, propagation delay and multipath delay spread remains less than the smallest cyclic prefix length of the OFDM system. For TDD systems, this condition prevents interference at the uplink/downlink switching points within the TDD radio frame. For collaborative radio techniques, it guarantees the functionality of joint signal processing.

Since TAE can be constrained to accuracies shorter than the sampling period, one of the requirements to achieve frame alignment is to distribute clock phase or time throughout the network. The other requirement is to guarantee equal fronthaul delays to coordinated radio units, which can be achieved by using buffers to hold the quicker frames until a given delay is "triggered". Note, however, that the latter issue is beyond the scope of this work.

3.1.3 Fronthaul Delay

The fronthaul one-way delay is defined as the delay from BBU to RRU or vice-versa. Although the limits for latency in the fronthaul are not defined by RAN standards [21], it is generally constrained to less than 150 microseconds. In [23], for example, it is claimed that a 75-microsecond one-way delay is typical for a 15-km fronthaul.

The principle governing this generalization is related to the hybrid automatic repeat request (HARQ) scheme [46] employed in LTE. In frequency-division duplexing (FDD) mode, for example, the BBU has to process an uplink (UL) subframe within the duration of 3 subframes and transmit an ACK or NACK message to the requesting user equipment (UE) in the fourth subframe (after 3 ms). If, then, a margin of 200 microseconds is left for the round trip in the fronthaul and 2.8 ms are reserved for BBU processing, a budget of 100 microseconds is left for one-way delays in UL and downlink (DL) directions.

3.2 The Common Public Radio Interface

The CPRI specification [18] defines the point-to-point link between a BBU and an RRU, respectively known as radio equipment control (REC) and radio equipment (RE) in the specification, or between two cascaded RRUs. It focuses on layer 1 (physical layer) and layer 2 (link layer) for single-hop and multi-hop topologies. The PHY layer supports both an electrical interface and an optical interface, while the link layer includes features such as media access control, flow control and protection of the data in the control and management flows.

CPRI is composed by three different information flows: user plane, synchronization plane, and control and management (C&M) plane. The C&M plane carries control data used for call processing and information for the operation, administration and maintenance of the CPRI link and the nodes; the user plane carries the actual baseband data transferred from REC to RE and vice versa, while the synchronization plane carries the data flow which transfers synchronization and timing information between nodes. The user plane data is transported in the form of IQ data, with flexible widths between 4 and 20 bits for I and Q in the uplink, and between 8 and 20 bits in the downlink. Each of these IQ data flows reflect the data of one antenna for one carrier, the so-called antenna-carrier (AxC), which are, then, arranged in a time-division multiplexing scheme. Control and management information are also time multiplexed with the IQ data, and vendor specific information are by specification allocated to most vacant bit slots.

The CPRI interface is compatible with radio base stations consisting of one REC and one or more REs under most of the current radio standards, such as GSM/EDGE, Universal Terrestrial Radio Access (UTRA) FDD, Evolved-UTRA (LTE) and the Worldwide Interoperability for Microwave Access (WiMAX). Furthermore, the interface supports a continuous operation range of cable lengths between master and slave ports, from zero to tens of kilometers, and uses separate transmission media (electrical cables or optical fibers) for uplink and downlink.

CPRI-compliant equipment has to support at least one of the following line bit rates (in Mbps): 614.4, 1228.8, 2457.6, 3072.0, 4915.2, 6144.0, 9830.4, 10137.6 and 12165.12. All CPRI line bit rates have been chosen in such a way that the basic UMTS chip rate of 3.84 MHz (or Mcps) can be recovered in a cost-efficient way from the line bit rate. The bit rates are altered by changing the length of a basic carrying unit, the *basic frame* (BF), whose rate is fixed and equal to the chip rate. In summary, a 10 ms frame (known as *CPRI 10 ms frame*) always contains 150 units known as *hyperframes*, each carrying 255 BFs. A BF, in turn, is composed by a fixed number of 16 words, whose width (in bits) is the only parameter configured in order to change the CPRI line rate option. For example, for basic frame words of 4 bytes, the CPRI line bit rate is 2457.6 Mbps, while when the words are composed by a single byte the rate is 614.4 Mbps. The word sizes and the corresponding line rates are summarized in Table 3.2 [18].

A BF carries multiplexed IQ data for distinct AxC within the so-called AxC containers, which carry a number of IQ samples corresponding to the oversampling ratio of the sampling frequency with respect to the chip frequency. The multiplexing scheme is designed to comply

[2]Line rates #8 and #9 adopt 64B/66B line coding, while the others use 8B/10B.

Option	#1	#2	#3	#4	#5	#6	#7	#8	#9
Line Rate	614.4	1228.8	2457.6	3072.0	4915.2	6144.0	9830.4	10137.6	12165.12^2
Word Size (bytes)	1	2	4	5	8	10	16	20	24
BF Size (bytes)	16	32	64	80	128	160	256	320	384

Table 3.2: CPRI line bit rate options and corresponding basic frame (BF) word sizes.

with the specific throughput requirements of the radio access standards. Table 3.3 summarizes the typical CPRI data rates corresponding to 1 AxC for different radio technologies [3].

RAN	GSM 1T1R	GSM 1T2R	WCDMA 1T1R	WCDMA 1T2R	LTE 10MHz 2x2	LTE 10MHz 4x2	LTE 20MHz 2x2	LTE 20MHz 4x2
Line rate (Mbps)	12.304	24.608	307.2	614.4	1228.8	2457.6	2457.6	4915.2

Table 3.3: Typical CPRI data rates for different radio technologies [3].

Calculation of the baseband data rate R transported over a CPRI link is based on the following expression:

$$R = \begin{cases} MW\left(\frac{10}{8}\right)\left(\frac{16}{15}\right) f_s, & \text{for 8B/10B line coding} \\ MW\left(\frac{66}{64}\right)\left(\frac{16}{15}\right) f_s, & \text{for 64B/66B line coding} \end{cases} \tag{3.1}$$

where M is the number of antennas per sector, W is the combined sample bit width of the in-phase (I) and quadrature-phase (Q) data, $\frac{16}{15}$ accounts for the control word inserted for each 15 IQ words [3] and f_s is the sampling rate (samples/s).

To illustrate the computation in (3.1) consider, for example, a 20MHz LTE system (sampled at 30.72MHz) with 2x2 MIMO and 30 bit IQ samples. In this case, the data rate is $R = (2)(30.72 \times 10^6)(30)(10/8)(16/15) = 2457.6$ Mbps. Such high data-rate is because baseband data is transported, not simply the original bits of application layer information. The difference is that baseband data corresponds to a fixed-point representation of the sample values originated by the particular radio modulation scheme, including all higher layer encapsulated data and the sources of redundancy (e.g channel coding and cyclic extension). This is the reason why the link rates required at the fronthaul are 10 times higher than at the backhaul [17].

An useful analysis is that of the maximum sampling frequency and corresponding LTE bandwidth that can be "served" by a given CPRI line rate option. The computation starts by

chosing a given a line rate option R and a sample width W. Then the number of AxC is set to the minimum ($M = 1$) and the maximum sampling frequency that can be used while still satisfying R as per (3.1) is obtained. Table 3.4 summarizes the results for $W = 30$ and $W = 32$, for the line rate option #1 and #2.

IQ Sample Width (Bits)	CPRI Line Rate (Mbps)	Maximum LTE f_s (MHz)	LTE BW (MHz)
$W = 30$	$R = 614.40$	$f_{\text{max}} = 15.36$	10
$W = 32$	$R = 614.40$	$f_{\text{max}} = 7.68$	5
$W = 30$	$R = 1228.80$	$f_{\text{max}} = 30.72$	20
$W = 32$	$R = 1228.80$	$f_{\text{max}} = 23.04$	15

Table 3.4: Maximum LTE Bandwidths that can be served by the CPRI line rate options #1 and #2 for IQ samples of 30 and 32 bits, respectively.

Most importantly for this work, CPRI specifies that REs shall operate with a clock recovered from the incoming signal transmitted over the fronthaul. Aside from generating a "DC-balanced" bit stream, the required 8B/10B or 64B/66B line coding is specifically tailored such that, at the expense of overhead, the 10-bit or 66-bit code words guarantee a tolerable maximum run length[3] of 1 bits or 0 bits, so that CDR circuits can work properly. This requirement, adopted in the CPRI specification, is because the existence of long string of 0 or 1 bits would reduce the so-called "transition density" of the code, which ultimately causes the data sample point of the clock recovery circuit to drift from the ideal position, namely the center of an eye diagram [47], causing bit errors. In another words, long run induces the clock recoverd through a CDR circuit [28] to drift and, therefore, generates jitter.

Regarding clock synchronization, CPRI specifies a maximum allowed cut-off frequency of 300 Hz for the loop filter of the synchronization mechanism employed at the RE (see Section 2.2), which should allow a standard crystal oscillator to be used as the master clock of an RE for improved cost-efficiency. It also establishes the maximum frequency offset contribution of a CPRI link to the frequency accuracy budget of the RE specified by the adopted RAN standard (e.g. the LTE requirements presented in Section 3.1). In general, CPRI specifies stringent clock stability and noise requirements for its PHY layer, which should support the requirement of a maximum bit error rate (BER) of 10^{-12}. Any electrical or optical physical layer capable

[3]Run length is defined as the length of a string or sequence of consecutive equal bits.

of meeting these requirements can in principle be employed, but the specification suggest a few variants. For example, PHY layers based on 1000BASE-CX (shielded balanced copper cable), 1000BASE-SX (multi-mode fiber) and 1000BASE-LX or 10GBASE-LX (single-mode fiber).

It is important to emphasize that the CPRI synchronization plane provides frame timing alignment, not strictly clock phase alignment. It contains certain control words (within the one control word in certain BFs among the 255 in an HF) that are used to keep track of the incoming stream. For example, the control word at the first BF of an HF consists of a special codes (e.g. K28.5) whose unique pattern allows finding the alignment of 8B/10B or 64B/66B codes, so it indicates the start of the hyperframe. The other synchronization control words (at BFs 64, 128 and 192) are used to keep track of the hyperframe number (HFN) and the number corresponding to the 10 ms frame, known as eNodeB frame number (abbreviated as BFN).

In Section 3.4, the CPRI requirements and features described in this section are further discussed with the goal of assessing the possibility of CPRI encapsulation over Ethernet.

3.3 1000BASE-T Ethernet Interface

The Ethernet specification is not directly in the scope of this work. However, since encapsulation of CPRI over Ethernet is desired, it is important to understand the structure of the Ethernet frames. Furthermore, since the hardware implementation of a "radio over Ethernet" module presented in Chapter 4 interfaces directly to an Ethernet MAC, it is important to clarify the organization of the MAC and PHY layers and interfaces.

3.3.1 Physical Layer and Synchronization

The modulation scheme adopted in the gigabit Ethernet 1000BASE-T is the 5-PAM. Such PAM modulation transmits symbols mapped from five distinct amplitudes, four of which are used to convey two-bit symbols and one (the fifth) to support forward error correction (FEC). Then, each one of its four wire pairs attains 250 Mbps throughput by transmitting with baseband signaling at 125 Mbauds, so that the system transmits 1 Gbps. Therefore, for 1000BASE-T the system clock frequency used to generate symbols is 125 MHz.

As described in Section 2.5, IEEE 802.3 specifies Ethernet clocks to be within ± 100 ppm [24]. These clocks are used to transmit a frame at one side, and to read the frame from a slip buffer at the receiver side, as highlighted in Fig. 2.4.

3.3.2 Ethernet Framing

An Ethernet frame is composed by the following elements: preamble, destination and source address, type field (known as *EtherType*), data and cyclic redundancy check (CRC) bits. The extended Ethernet frame format defined in IEEE 802.1Q also contains the so-called VLAN tag in its header. The following summarizes the elements that deserve mention in this work.

The *preamble* is used by the CDR circuitry at the receiver-side to lock onto the transmitter's clock. The *EtherType* indicates the type of network protocol encapsulated within the Ethernet frame and is used in the system to multiplex "radio over Ethernet" frames and PTP frames. Finally, the *data field* carries the encapsulated baseband radio data. The field shall be minimally of 46 bytes and maximally 1500 bytes long. Section 3.4.2 discusses how much radio data can be encapsulated in the Ethernet frame's data field for different CPRI line rates.

3.3.3 PHY Sublayers and Interfaces

The physical layer of Ethernet systems is composed by three sublayers defined among several clauses of the IEEE 802.3 specification: the physical coding sublayer (PCS), the physical medium attachment (PMA) sublayer and the physical medium dependent (PMD) sublayer. The PCS is responsible among other tasks for link rate negotiation, line encoding (in the direction from MAC to PMA) and decoding (from PMA to MAC), and scrambling/descrambling. The PMA sublayer serializes (deserializes) data to (from) the PMD and provides alignment of the line encoded data. Finally, the PMD sublayer defines the details of transmission and reception of individual bits (regardless of their relation established in the PCS) on a physical medium.

The PCS sublayer is the one that interfaces to the MAC layer. This interface is provided via the standardized Gigabit Media Independent Interface (GMII), which consists of 8-bit parallel data lines and several control lines [48], or other variants such as the reduced gigabit media-independent interface (RGMII) and the serial gigabit media independent interface (SGMII) [49]. Likewise, the PMD sublayer (bottom most PHY sublayer) interfaces with the physical medium via the so-called media dependent interface (MDI). Finally, there is also a standard interface for exchanging control information between the PHY and the MAC layers, the management data input/output (MDIO). Such standard interfaces allow interoperability of PHY and MAC chips and also between the PHY and the physical media. Fig. 3.2 summarizes the sublayers and interfaces within the Ethernet PHY.

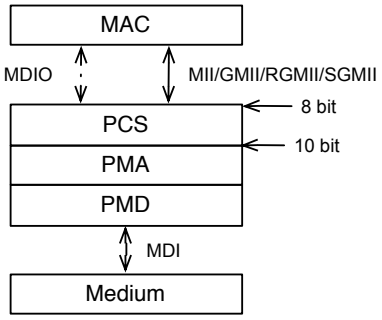


Figure 3.2: Ethernet PHY sublayers and interfaces.

3.4 CPRI over Ethernet

The importance of discussing radio encapsulation over Ethernet in this work is to define the packet stream that will be sharing the Ethernet network with PTP data. The system of interest transmits one or more constant bit rate (CBR) CPRI streams packetized on Ethernet frames and recovers synchronization by using the CBR stream itself or two-way PTP packet exchange. In the sequel, an overview of the previous art in such architectures is presented.

3.4.1 Previous Art

The concept of transporting CPRI as Ethernet packets is present in a number of patents published since 2009 [50–53]. The earliest of such patents [50] proposes a thorough removal of unnecessary bits in a CPRI stream during encapsulation of the radio data into Ethernet frames. It also proposes the usage of IEEE 802.1Q tagging to allocate higher priority for CPRI traffic and suggests aggregation of Ethernet links for utilization of the existing infrastructure. The approach claims to yield more than 20% in bandwidth savings.

Differently, patent [51] proposes an adaptation layer between CPRI and (preferably) 10-Gigabit Ethernet (10-GbE) with the goal of reducing the number of fiber cables between RRUs and BBUs. This adaptation layer is combined with Synchronous Ethernet (SyncE) having a slightly different clock frequency of 153.6 MHz (a multiple of LTE's sampling rate), which is proposed by the invention in order to minimize the filling bits. This enables the RRU to align in frequency with the BBU and meet the radio performance requirements. Nevertheless, the scheme requires all intermediate nodes between the RRU and BBU to implement SyncE functionality, which precludes legacy infrastructure reutilization.

In contrast, reutilization is exploited in patent [52], which proposes CPRI-to-Ethernet

(synchronous-to-asynchronous) adapters that minimize the delay variations experienced by the CPRI frames as they are transported over asynchronous links. The adapters employ a dynamic buffering mechanism for incoming packets, which compensates PDV by adjusting the time each packet is held on a buffer prior to being forwarded to the adjacent CPRI-compliant equipment such that the overall delays between BBU and RRU are maintained at static and symmetric values. Since timestamps are necessary for this, a parallel bidirectional IEEE 1588 flow is also used to synchronize the adapters in time and frequency.

An architecture for radio over Ethernet has also been recently published in patent [53], which adopts a master-slave hierarchical distribution of clocks. The method employs a hybrid physical-layer-based and packet-based solution for phase and frequency alignment. In particular, SyncE is adopted to maintain frequency synchronization between modules via the physical layer and a timestamping protocol such as (but not mandatorily) PTP is employed for phase alignment via timestamped packets exchanged between BBUs and RRUs.

Another effort is the relatively recent IEEE 1904.3 task force created to standardize encapsulations and mappings of baseband data into Ethernet. An initial feasibility study contributed to the group [54] shows that the fronthaul delay over Ethernet is acceptable, but the PDV requirement is more challenging to be satisfied: Ethernet alone, with or without a preemption scheme could not meet the minimum PDV requirement. Furthermore, the work suggests Ethernet with preemption, when combined with de-jitter buffering at the edges, could meet the PDV requirement. This is because buffering can effectively eliminate PDV, and even though it may increase the one-way delays, the increase is tolerable while still satisfying CPRI requirements.

A similar effort is carried in the standardization of CPRI over optical transport networks (OTN), in ITU-T recommendation G.709 [55]. The recommendation provides a mechanism for encapsulating CPRI into OTN frames (for CPRI data-rate options 1 to 6 only). However, it currently does not provide the transport of timing and synchronization signals.

3.4.2 Radio Encapsulation and Mappings

The first question that should be addressed is why actually encapsulate IQ data into CPRI and then CPRI into Ethernet, if IQ data could be directly encapsulated into Ethernet? The answer in this work is to avoid the need for equipment upgrades and to focus on a backwards-compatible and cost-efficient solution.

The second question is how much CPRI data should be contained within Ethernet frames.

The answer depends on a few variables. One of them is the probability of frame loss due to CRC error detection, in which case the frame is simply discarded at the receiver. If too many CPRI BFs are packetized in an Ethernet frame, a single frame loss causes many CPRI bit errors. Another variable is obviously the Ethernet data field length, which restricts the number of CPRI basic frames that can be transported. Finally, a third regards the effects of the frame lengths on synchronization packets exchanged for example by PTP, which will be investigated in this work.

According to Table 3.2, the maximum BF length is of 384 bytes, so that any line rate option can fit into Ethernet frames. Table 3.5 summarizes the maximum number of BFs per Ethernet packet for each CPRI line rate option, considering line coding is not necessary when CPRI data is encapsulated.

Line Rate (Mbps)	614.4	1228.8	2457.6	3072.0	4915.2	6144.0	9830.4	10137.6	12165.12
Max BFs	93	46	23	18	11	9	5	4	3

Table 3.5: Maximum number of CPRI BFs per Ethernet packet for each CPRI line rate option.

The number of BFs in Table 3.5 could be further reduced with IQ data compression or by avoiding unnecessary bit stuffing in the BFs. However, such pursuit is out of scope and this work solely aims at maximum compatibility with legacy CPRI equipment.

3.5 IEEE 1588 Precision Time Protocol

The previous sections introduced the interfaces and protocols that compose a BBU or RRU compliant to an Ethernet-based fronthaul. However, since the focus of this work is on the provision of synchronization in these systems and one of the considered approaches relies on a protocol, the latter should be addressed before proceeding into further details. The protocol is the so-called precision time protocol (PTP), specified in IEEE 1588 [2].

PTP defines a generic two-way exchange of timestamped packets to provide time synchronization between two clock units. It consists of a master-slave architecture, namely a node in the network can operate in either master or slave mode (or both). The master must provide very accurate time-of-day information, which the slave can rely on to discipline its local clock.

An *ordinary clock* (OC) is a PTP node that contains a single physical port (e.g. a single Ethernet interface) and, therefore, can only operate as master or slave. For example, BBUs and

RRUs are OCs. In contrast, a *boundary clock* (BC) or *transparent clock* (TC) is a PTP node with multiple ports, some of which are master in their links and one that is a slave. It is the case of an Ethernet switch. Ideally, like in the ITU-T 8275.1 profile [56], BCs or TCs would be used in the network because of their features to avoid the effects of PDV. However, nothing prevents an end-to-end architecture with regular switches to be used. The latter is the case of interest in this work, since the goal is to allow usage of legacy network.

When operating in master mode, a PTP node initiates a PTP processing cycle by emitting the so-called SYNC message to the slave. This message can be sent with a minimal rate of 1 every 16 seconds and or a maximum rate of 128 packets-per-second, which ideally is sufficient with respect to the frequency of physical variations in telecom-grade oscillators. Upon the reception of a SYNC packet, the slave device in the link immediately sends the DELAY_REQ message back to the master. Finally, the master replies with the DELAY_RESP message. This exchange is known as the *delay request-response mechanism*, which is illustrated in Fig. 3.4.

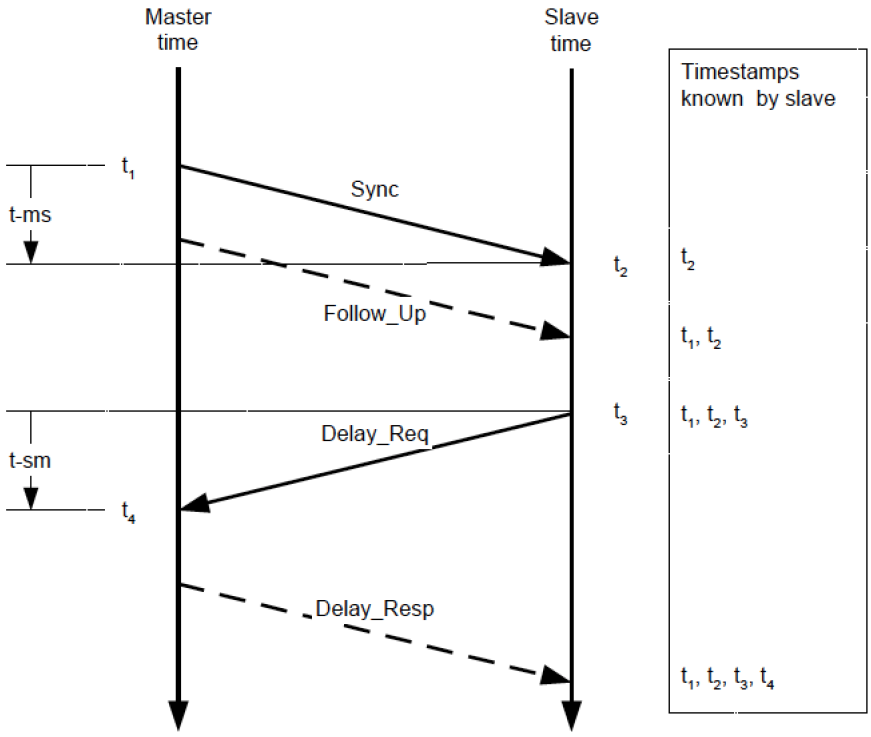


Figure 3.3: Basic message exchange in the IEEE 1588 delay request-response mechanism [2].

The above operation involves the exchange of four timestamps: t_1, t_2, t_3 and t_4. Timestamps t_1 and t_4 are taken in the master, while t_2 and t_3 are taken in the slave. The first timestamp (t_1) represents the time the master emits the SYNC message. It can be conveyed either along the SYNC message itself (defined as *one-step mode*) or in the so-called FOLLOW_UP message

(*two-step mode*), depending on the accuracy of the embedded timestamping unit. Upon the reception of a SYNC message, the slave timestamps the arrival time t_2. Next, the slave sends a DELAY_REQ message to the master and stores its transmit time t_3. Finally, when the DELAY_REQ message is recognized at the master, the arrival time t_4 is taken and replied back to the slave in the DELAY_RESP message. The timestamps are also illustrated in Fig. 3.4.

The above timestamps are specified [2] to be composed of a seconds and a nanoseconds part. The seconds part should be represented by 48 bits and the nanoseconds by 32 bits, but always kept less than 10^9, namely within 30 bits[4]. Fractional nanoseconds are not sent over wires except on the so-called correction fields. Furthermore, messages exchanged in the protocol (and therefore timestamps) have an associated source port identity, which informs the egress PTP node[5] and interface within the node. A responder replies back this information so that the requestor can check whether a valid exchange of timestamps was carried.

Since the master is supposed to be locked to a PRC, for example by synchronizing to a clock received through a synchronous interface, the slave is the one who does clock correction. The slave PTP node computes the time and frequency offset of its local RTC module with respect to the master clock. The estimations and corrections are explained in the sequel.

3.5.1 Link-delay and Time Offset Estimations

Assuming the master clock as an ideal reference clock, the slave's clock has both a time offset $x(t)$ a frequency offset $y(t)$ relative to the master. Thus, from (2.5), the "true" (or master) times of the timestamps at a particular processing cycle are:

$$\begin{cases} t'_1 = t_1, \\ t'_2 = t_2 - x(t_2), \\ t'_3 = t_3 - x(t_3), \\ t'_4 = t_4, \end{cases} \tag{3.2}$$

In possession of the four timestamps, the slave can assume the master-to-slave delay d_{ms} and the slave-to-master delay d_{sm} are given by (3.3) and (3.4), respectively.

$$d_{ms} = t'_2 - t'_1 = t_2 - t_1 - x(t_2). \tag{3.3}$$

[4]The remaining 2 bits are useful to perform arithmetic with the 30 bit nanosecond timestamps.

[5]A node participating in the PTP is defined in the protocol as simply "clock".

$$d_{sm} = t'_4 - t'_3 = t_4 - t_3 + x(t_3). \tag{3.4}$$

The next step comes from additional noteworthy assumptions. The first is that the master-to-slave and slave-to-master delays are symmetric. In practice, this is hardly true, but as the specification states, the asymmetry in many cases is small relative to the propagation delays and the corresponding error in some estimations is normally within the acceptable budget.

The second assumption is that the time-offsets $x(t_2)$ and $x(t_3)$ can be assumed equal. From the model in (2.28), this is not unreasonable considering the phase noise has small variance (e.g. of 10^{-14} in [42]), the oscillator has very low drift rate and the latency from SYNC reception to DELAY_REQ transmission is very small and controlled by design. For example, assuming the hardware takes 160 ns to reply, an initial frequency offset of 100 ppm, a relative drift rate of 10^{-7} per day that is typical for quartz crystal oscillators [29], neglecting phase noise yields a time offset change of a few picoseconds for a very long period of time (the difference depends on t due to the quadratic term). In fact, oscillators change their parameters slowly with respect to typical PTP periods [57], so that the assumption can be well approximated.

Then, by making $x(t_2) = x(t_3) = x_k$, the link delay can be estimated as:

$$\hat{d}_k = \frac{d_{ms} + d_{sm}}{2} = \frac{(t_{4,k} - t_{1,k}) - (t_{3,k} - t_{2,k})}{2} \tag{3.5}$$

where $t_{1,k}$, $t_{2,k}$, $t_{3,k}$ and $t_{4,k}$ are the timestamps conveyed in the k-th PTP processing cycle.

With the link delay estimation in (3.5), the master-slave time offset during the k-th PTP cycle can be estimated. Recall the time offset defined in (2.5) is estimated at the slave node as:

$$\hat{x}(t) = T_S(t) - \hat{T}_M(t) \tag{3.6}$$

where $T_S(t)$ is the exact slave time at time t and $\hat{T}_M(t)$ is an estimation of the reference (master) time at time t, respectively.

Thus, the time-offset can be effectively computed by:

$$\hat{x}_k = t_{2,k} - \left(t_{1,k} + \hat{d}_k + \gamma_k\right), \tag{3.7}$$

where the master time by the time its SYNC message arrives at the slave is obtained by adding the most recent link delay estimation $\hat{d}_k$ to t_1 and a correction field γ_k that accounts for eventual latencies in intermediate nodes[6]. That is, it is estimated as $\hat{T}_M(t_{2,k}) = t_{1,k} + \hat{d}_k + \gamma_k$.

[6]The correction field can include the residence times added by transparent clocks along the network.

A mechanism different than the *delay request-response* is the *peer-delay mechanism*, which is adopted in this work. In this mode, each PTP node in the network estimates the delay between itself and its link peer, instead of only the slave node computing the overall delay from master. The approach is to exchange the similar set of messages in Fig. 3.4. Each node periodically sends a PDELAY_REQ message to its peer, which in turn responds with PDELAY_RESP and potentially a follow-up message. In the end, the node that initiated the request obtains all four timestamps, so that it can estimate the delay using (3.5).

When the peer-delay mechanism is used, time offset estimation can be promptly performed whenever a SYNC (and if applicable a FOLLOW_UP) message reaches a slave node, with no need for it to first request a delay measurement (send a DELAY_REQ) back to the master. This implies delays are estimated with two-way exchanges, but time (and frequency) offsets only need one-way transmissions emanated from the master. Furthermore, since delays vary much less frequently than RTC offsets, the peer-delay mechanism can be configured with lower periodicity than the one-way SYNC messages that lead to offset corrections.

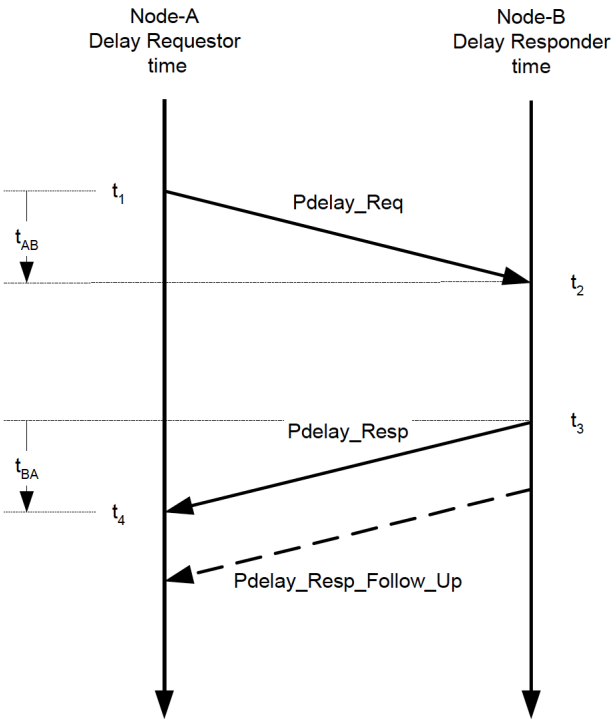


Figure 3.4: Peer-delay mechanism, from [2].

3.5.2 Frequency Offset Estimation

Based on the sequence of time error estimations carried after each pair of SYNC and FOLLOW_UP messages is received, it is possible to estimate the instantaneous clock frequency offset. From the discrete approximation of the frequency offset in (2.19), it can be estimated based on the difference between consecutive time error estimations and their relative interval:

$$\hat{y_k} = \frac{\hat{x}_k - \hat{x}_{k-1}}{T_M(t_{2,k}) - T_M(t_{2,k-1})}, \tag{3.8}$$

where the denominator is the interval from one offset estimation to the other, measured in the reference's (master) timescale. From (3.7), $T_M(t_{2,k})$ is the *corrected master event timestamp* (c.f. terminology adopted in [2]), which is equal to $t_{1,k} + \hat{d}_k + \gamma_k$.

By substituting the definition of offset in (3.6), yields:

$$\hat{y_k} = \frac{[T_S(t_{2,k}) - T_S(t_{2,k-1})] - \left[\hat{T}_M(t_{2,k}) - \hat{T}_M(t_{2,k-1})\right]}{\hat{T}_M(t_{2,k}) - \hat{T}_M(t_{2,k-1})}. \tag{3.9}$$

What essentially the above expression says is: measure two intervals concurrently according to both the slave and the master timescale. The difference between the two measurements, divided by the actual interval, corresponds to the frequency offset estimation. Therefore, by the same argument used in (3.7), the estimation can be re-stated as:

$$\hat{y_k} = \frac{\Delta T_S - \Delta T_M}{\Delta T_M}, \quad \text{where} \tag{3.10}$$

$$\begin{cases} \Delta T_S = t_{2,k} - t_{2,k-1}, \\ \Delta T_M = \left(t_{1,k} + \hat{d}_k + \gamma_k\right) - \left(t_{1,k-1} + \hat{d}_{k-1} + \gamma_{k-1}\right) \end{cases}$$

3.5.3 Clock Corrections

Once the frequency offset is estimated, the typical approach for frequency correction is to adjust the increment value of the local RTC counter and attempt to keep it at the value corresponding to the exact clock period. A frequency driving an RTC corresponds to the number of increments performed in the RTC in one second, but an RTC must count exactly one second in one second. Thus, clearly the solution is to count by steps corresponding to the clock period.

A frequency offset y_k implies that the slave clock frequency during the k-th PTP processing cycle corresponds to $f_{s,k} = (1 + y_k)\, f_{m,k}$, where $f_{m,k}$ is the master frequency during the cycle (ideally equal to f_{nom}). Then, it follows that the slave clock period $\tau_{s,k}$ during the k-th

processing cycle is related to the master period $\tau_{m,k}$ by:

$$\tau_{s,k} = \frac{\tau_{m,k}}{(1 + y_k)}. \tag{3.11}$$

The actual period at the master is unknown, but given it should be locked to a PRC the slave can safely assume that it is the nominal value τ_{nom}. Then, the k-th frequency offset estimation implies the slave increment value should be adjusted by:

$$\Delta\tau_k = \tau_{s,k} - \tau_{s,k-1} = \left(\frac{1}{1 + y_k} - \frac{1}{1 + y_{k-1}}\right)\tau_{\text{nom}}, \tag{3.12}$$

which can be approximated by the following to avoid division:

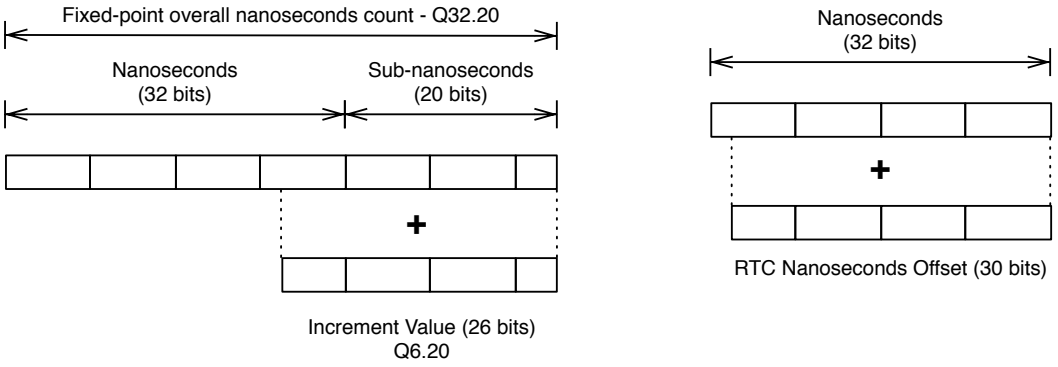
$$\Delta\tau_k \approx (y_{k-1} - y_k)\,\tau_{\text{nom}}. \tag{3.13}$$

However, every time the RTC increment value is altered, since the actual frequencies driving the RTC are not being disciplined (i.e. the oscillators are free-running), the slave's perception of the frequency offset changes due to a change in the ΔT_S measurement of (3.10). If, in particular during the k-th estimation cycle it can be assumed that the offset estimation $\hat{y}_{k-1}$ was already corrected, then the new estimation of the frequency offset attempts to approximate $y_k - \hat{y}_{k-1}$. Substituting this in (3.13), yields the expression used for increment adjustments:

$$\Delta\tau_k \approx -\hat{y}_k\tau_{\text{nom}} \tag{3.14}$$

Since $\hat{y}_k$ is a number in $[0, 1]$, the above adjustment effectively works as a compensation. It is negative and reduces the increment value for positive offsets (slave with faster frequency), while it is positive and increases the increment for negative offset (slave with slower frequency).

The key for a precise adjustment using (3.14) is to have sub-nanosecond bits at the increment value and in the RTC itself. Even though PTP exchanges timestamps given as integer nanoseconds (again, except when using the correction field), nothing prevents the hardware to keep track of time using a resolution finer than nanoseconds. By using a sub-nanosecond increment value, eventually the summation of fractional nanoseconds in the RTC will propagate to integer nanoseconds through a carry. For example, a nominal clock of 25 MHz driving an RTC with -100 ppb offset implies an actual frequency of 24.9975 MHz, whose corresponding period is bigger than the nominal period by approximately 4 ps. In this case, by using a number of sub-nanoseconds bits capable of representing 4 ps, after 250 increments of the RTC an additional nanosecond will propagate to the integer part.



(a) Syntonized clock. RTC with adjusted increment value using sub-nanoseconds precision.

(b) Synchronized clock. RTC value summed with the offset register.

Figure 3.5: RTC control.

In a counter with 32 bits for integer nanoseconds and 20 bits for sub-nanosecond, as used in the system of this work, the nanosecond numbers tracked in the counter are Q32.20 fixed-point numbers (c.f. [58]), whose resolution is 2^{-20}, i.e. approximately 9.537×10^{-4} ps. For the increment value, the bits representing integer nanoseconds must be enough to keep the clock period, known by design. In the system of interest, which can be clocked from 25 MHz (40 ns period) to 125 Mhz (8 ns period), 6 bits are sufficient. Thus, the increment value can be represented by a Q6.20 number, as shown in Fig. 3.5a.

The time offset computed periodically through (3.7) could also be used to adjust the local RTC time. However, the system adopted in this work separates time and frequency adjustments, so two RTC values are made available, the *syntonized* (controlled in frequency) and the *synchronized* (same frequency and time) RTC, depicted in Fig. 3.5. The former is obtained by solely correcting the RTC increment value , while the latter (synchronized) is obtained by adding the value in the dedicated time offset registers to the syntonized clock.

The synchronized clock timescale is generated by periodically updating (increasing or decreasing) the always-positive time offsets value stored in internal registers. Such updates are usually termed “step adjustments”, for example used to set the epoch[7]. The updates result from the estimations carried through (3.7), which is based on timestamps taken from the *syntonized* RTC (where time-offsets are not corrected). By arranging this away, the continuously estimated time-offsets are progressive and the frequency offset in (3.10) can be obtained.

[7]Epoch is defined as the origin of the timescale, normally defined by the upper 16 bits within the 48 bits used to count seconds.

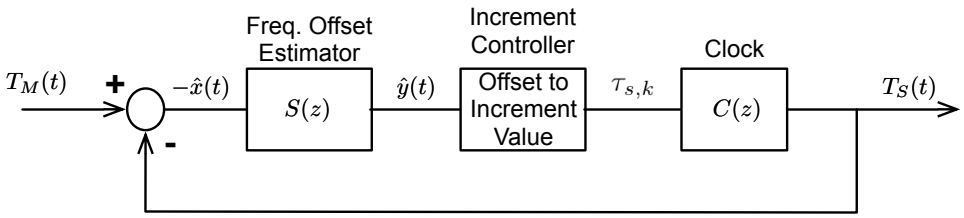


Figure 3.6: Close-loop system for clock rate correction.

The fact that the increment value of the syntonized clock is iteratively adjusted implies a servo control scheme for the clock. PTP servo control schemes can be found in many implementations, for example in [59,60] and the popular *linuxptp* implementation [43]. One example is the structure illustrated in Fig. 3.6. Note the first component takes the negative of the time offset in its input and outputs the frequency offset estimation computed using (3.8). It can be thought as a cascade compensator, whose Z-transform can expressed as:

$$S(z) = -\frac{(1-z^{-1})}{\Delta T_M} = \frac{1}{\Delta T_M}\frac{1-z}{z}.$$

where ΔT_M is the interval given in (3.10).

The second component in Fig. 3.6 is the controller, which takes in its input the frequency offset estimation and outputs as control signal the increment value $(\tau_{s,k})$ from (3.14), which could also be scaled to implement a proportional controller with a non-unitary constant. The last component is the plant, the actual clock system, which produces a time scale based on the increment value. The following difference equation models the slave time:

$$T_s(t_k) = T_s(t_{k-1}) + \tau_{s,k-1}\frac{\Delta T_M}{\tau_{\text{nom}}} \tag{3.15}$$

Hence, the Z-transform of the clock plant is:

$$C(z) = \frac{\Delta T_M}{\tau_{\text{nom}}}\frac{1}{(z-1)} \tag{3.16}$$

Finally, note that since the estimations and the model above are all dependent on computations carried using timestamps, the accuracy of the timestamps is determinant for precision achieved with clock correction. Hence, timestamps should be taken in hardware, right after a start of frame delimiter is recognized in the interface between the PHY and the MAC layer. Furthermore, the latency in the PHY should be controlled, such that its value can be subtracted from timestamps for further improvements in the accuracy.

3.5.4 Derivation of a Synchronized Clock Signal

The correction presented so far only adjusts the pace in which a timescale generated by an RTC progresses. The increment rate is totally related to frequency and, therefore, its adjustment is termed frequency correction. However, increment correction does not generate a syntonized clock signal in hardware, only implies that the syntonized RTCs count with the same rate.

The generation of a syntonized local clock signal is an additional feature to a PTP system and of interest in this work's application, since RRUs are required to generate a local clock signal syntonized with the BBU's local clock. One approach is to compare the RTC nanoseconds counter to a number corresponding to half of the desired output clock period in nanoseconds. Then, by using modulo arithmetic, any time the nanoseconds count reaches half period, the output signal is toggled, effectively generating a clock in the desired reference frequency. In the end, since the same derived clock can be generated at the master, the two clock signals are said to be syntonized, just like their corresponding RTCs.

The output clock can also be derived from the synchronized (not just syntonized) nanoseconds count. The advantage of this approach comes from the fact that RTC increment correction will invariably leave some residual frequency offset in the system (unless infinite sub-nanoseconds bits were available). The problem is that the residual offset provokes cumulative time offsets, as predicted by (2.28). As a solution, when the synchronized RTC is used, the residual time offsets are periodically corrected, so if the residual frequency error is small enough, the time offsets are corrected before any siginficant error is accumulated. In the end, the derived clock signal becomes aligned in phase with the reference clock signal at the master, at the expense of jitter due to the step changes in the RTC. For example, if a 100 ppb offset is left and the periodicity of the correction is 0.125 s, a phase error of 12.5 ns is acumulated before the correction, so the clock becomes aligned in time within ± 12.5 ns.

Chapter 4

Architectures for Synchronization of Radio over Ethernet

This chapter describes the time and frequency synchronization architectures designed in the FPGA-based testbed developed in this work. Although a significant part of the development was done using hardware description language (HDL), more specifically VHDL, the low-level details of the entire system are not under focus. Instead, this chapter covers the system level behavior dictated by a combination of software and hardware and exposes hardware details only for the modules considerably involved with the synchronization solutions.

The chapter describes and contrasts two methods, for which both the clock misalignment estimator and the clock correction mechanism are described. Note, on the one hand, it is not sufficient to accurately detect the phase and frequency mismatches, since this can only provide information, but not actually solve the problem. On the other hand, it may be too expensive to guarantee the desirable clock nominal frequency and certainly not possible to phase align without any mechanism for detecting the error with respect to a master clock. Hence, both elements are of interest, specially because they introduce distinct limitations, for example the resolution of the frequency offset detection and the *settability* of the frequency synthesizer.

The main contributions of this chapter include the guidelines for the design and implementation of both the simple buffer-based and the PTP-based synchronization architecture. Specifically for PTP, a combination of miscellaneous techniques for improving the synchronization accuracy is proposed for an end-to-end PTP solution over legacy Ethernet infrastructure. Furthermore, the chapter presents important design considerations tailored specifically to the radio over Ethernet stream in order to prevent undesirable disturbances into the synchronization

scheme. Finally, the chapter discusses the expectations in terms of accuracy and the limitations of such a design for fronthaul synchronization.

The first synchronization architecture of interest employs a detection of frequency mismatches between BBU and RRU clock frequencies based on the fill (occupancy) level of a slave-side (RRU-side) elastic buffer. It has the advantage of simplicity in the implementation, specially because the elastic buffer is a necessary component in the receive chain regardless of the synchronization structure. However, it does not provide time or phase alignment, and therefore can not be sufficient to fulfill all the requirements presented in Section 3.1.

An architecture based on the buffer occupancy is a classical solution employed for clock recovery, but revisited here in the context of CPRI over Ethernet fronthaul transport. For example, patent [61] (dated from 1983) exposes an architecture that aims to equalize the rates at which data is stored in an elastic FIFO with the rate that data is read out. It proposes the periodic storage of output identifiable bits into the FIFO and measurement of their residence time (in the fifo), which indicates the buffer occupancy and can be mapped into voltages that drive a VCO and, ultimately, generate a smooth read-side syntonized clock.

Buffer occupancy based systems are also encountered in other synchronization contexts. For example, to maintain the continuity of multimedia streams [62] and inter-stream synchronization [63]. More inter-related with this work, buffer fill levels are at the core of the functionality of the packet-based flow control scheme in IEEE 802.3 (Clause 31) [64]. The scheme attempts to prevent overflows in the slip buffers (see again Fig. 2.4) of a slower receiver by issuing a pause request to the faster transmitter when the fill level is above a certain threshold.

The usage of a half-full buffer for local clock synchronization is in contrast to usual flow control. The difference is that rather than pausing a transmission, the transmission is kept running and the clock is instead adjusted online. A motivation for this approach is that flow control is not sufficient for the transport of radio data originated synchronously and transported in a TDM manner. This is because radio data latencies in the fronthaul can not vary in the order that otherwise would if flow control asynchronous “pause requests” were communicated back to the transmitter and if the transmit effectively ceased transmission for the requested period.

Furthermore, note that the goal with buffer-based synchronization and the general synchronization schemes presented in this chapter is not to prevent slips in the internal Ethernet MAC buffer, as flow control would. Instead, the goal is to prevent overflows and underflows in the buffer that is at the interface between the Ethernet MAC and the “radio over Ethernet”

module that will be presented, which is read with a clock in the CPRI domain. Advantageously, if no other heavy Ethernet traffic is exchanged between the two nodes, synchronization of the latter buffer implies a seamless operation of the Ethernet MAC buffer.

In contrast to the buffer-based method, the second architecture of interest in this chapter provides both phase and frequency alignment by implementing the IEEE 1588 precision time protocol (PTP). The protocol has wide use in packet-based telecom transport networks and accordingly two PTP *profiles*[1]. More specifically, the ITU-T Telecom Profile [65], which specifies frequency synchronization (aiming at mobile backhaul applications), and more recently the profile for phase/time synchronization with full PTP support in all nodes within the network [56]. The system considered in this work aims at a proof-of-concept and evaluation of the distribution of synchronization when multiplexed with CPRI streams, but does not intend to strictly adhere to the aforementioned profiles. In the end, the performance achieved with the PTP-based architecture is compared to the buffer-based approach.

The IEEE 1588 PTP provides a solid foundation for network clock synchronization, except for the limitation introduced by packet delay variation, which affects many essential assumptions of the estimations carried through the protocol. In the literature, many algorithms and approaches are proposed to deal with the PDV in telecom networks. A common treatment is to filter the estimations and what is known as *packet selection*, namely the ability to select specific PTP messages to carry the estimations, discarding other messages regarded as more affected by PDV. A brief survey about approaches to overcome PDV difficulties is postponed to Section 4.3.2, after the issue becomes clear in the explanation of the PTP architecture.

This chapter is organized as follows: Section 4.1 gives an overview of the hardware design in which the synchronization architectures are inserted, including all relevant system components and peripherals, specially the radio over Ethernet core, which coordinates the CPRI encapsulation into Ethernet, and the CPRI emulator module. Next, Section 4.2 describes the synchronization architecture based on the half-full buffer. Then, Section 4.3 describes the PTP-based architecture. Finally, Section 4.4 compares the two architectures.

[1]The IEEE1588 recommendation allows the so-called *profiles* to define features and specifications of interest to peculiar requirements of certain industries.

4.1 Hardware Overview

Fig. 4.1 presents an overview of the hardware design developed for the implementation of the synchronization architectures. At the core of the design lies the developed Radio over Ethernet intellectual property (IP), which interfaces via the AMBA AXI Stream protocol [66] to an "off-the-shelf" Ethernet MAC IP (Xilinx AXI 1G/2.5G Ethernet Subsystem [67]), implemented in the FPGA programmable logic fabric. The Ethernet MAC, in turn, interfaces to an external (to the FPGA) Ethernet PHY chip via an SGMII (see Section 3.3.3).

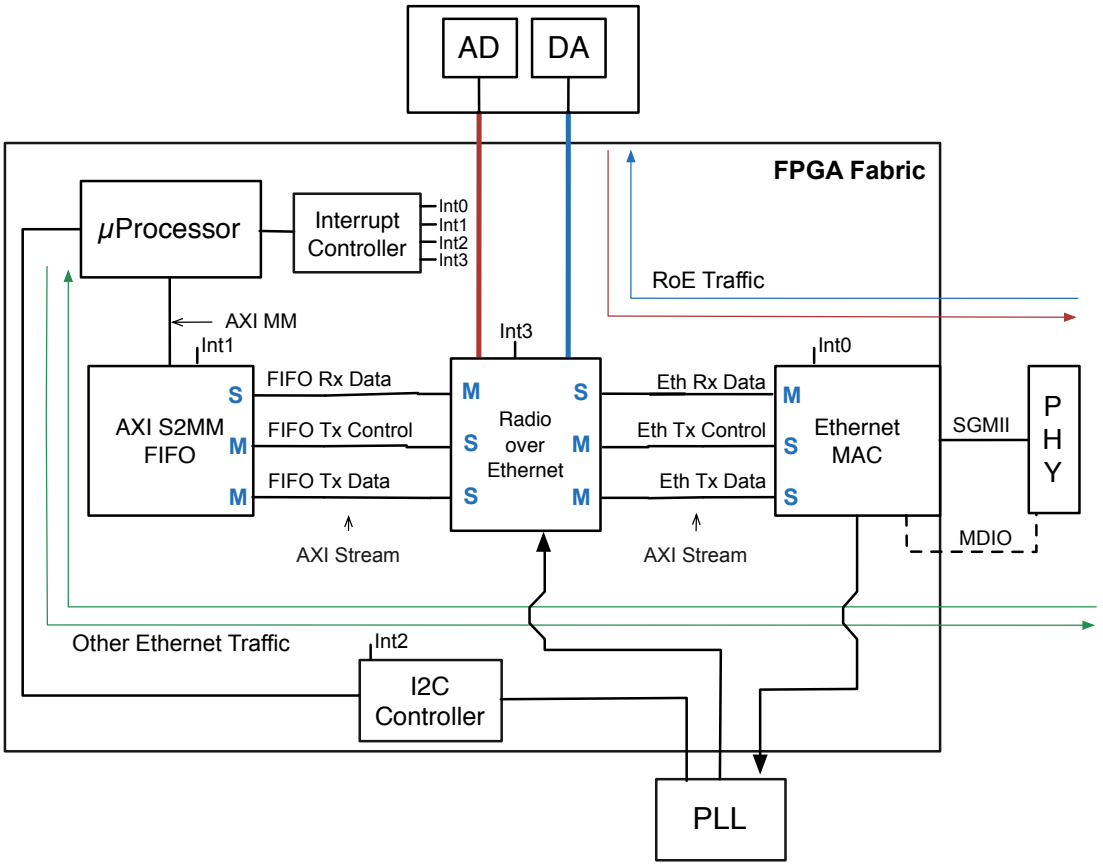


Figure 4.1: Hardware design overview.

All the traffic towards and from the Ethernet MAC passes through the Radio over Ethernet (RoE) IP. However, the module only intercepts the traffic marked as "RoE", by using an internal software-configurable packet filter developed in VHDL, which captures EtherType octets within the Ethernet frame header and checks whether they should be processed by the module. Other traffic types are just normally relayed to an AXI Stream to Memory Mapped (S2MM) FIFO, whose purpose is to buffer streams of data received via the AXI Stream protocol and avail this data to a microprocessor that reads punctual ranges of data using the slightly different memory

mapped AXI protocol (AXI MM) [66]. By doing so, several different traffics can be exchanged by the hardware, for example it is possible to multiplex RoE and background traffic.

The Ethernet MAC presented in Fig. 4.1 has three AXI Stream interfaces, two for data and one for control. The two data buses are used for transmit and receive data. The former, labeled as "Eth Tx Data", transports data originated at the microprocessor or coming from e.g. the air interface via the ADC (through the Radio over Ethernet IP) towards the Ethernet PHY for transmission. Thus, at the MAC module, Tx data represents a slave interface (labeled "S"), since it receives data from the RoE core. In the reverse direction, the data received via the Ethernet PHY is transported from the Ethernet MAC towards the Radio over Ethernet IP or the microprocessor. Hence, the traffic labeled "Eth Rx Data" at the MAC is output via a master (labeled "M") AXI Stream interface. Finally, a control stream labeled as "Eth Tx Control" is transmitted by the processor towards the MAC, where it configures some parameters of the transmission stream. The same pattern is observed at the AXI S2MM FIFO.

The RoE traffic (labeled "Radio Traffic" in the figure) has the DAC interface as its final destination or the ADC as its origin in the reverse direction, both of which synchronously provided (acquire) new samples towards (from) the air interface. Alternatively, if for example an Ethernet-to-CPRI adapter is desirable, the AD/DA synchronous interface could be replaced by a synchronous CPRI interface, which could be connected to a legacy CPRI equipment. Nevertheless, the specific interfacing application is out of the scope of this work and instead the system of interest uses an internal synchronous CPRI traffic emulator module developed in VHDL.

Another important element of the system is the external PLL used for controlled frequency synthesis. This PLL receives as a reference the 125 MHz Ethernet clock[2] (see Section 3.3.1), which is free-running and whose frequency is accurate to within ± 100 ppm. Then, it outputs a frequency based on the configurations assigned to its internal dividers, which are configured via the Inter-Integrated Circuit (I^2C) protocol.

Furthermore, an interrupt controller module is provided to detect and acknowledge the interrupt signals that can be asserted by modules in the FPGA. As highlighted in Fig. 4.1, interrupts can come from the Ethernet MAC, the Ethernet FIFO, the I2C controller and the RoE module. They are necessary for calling interrupt service routines (ISRs) with specific command for each module. The interrupts of the RoE module will be discussed in the succeeding sections.

[2]The Ethernet MAC subsystem has several internal clocks and two of them are referred in this text. The 125 MHz clock represents the baud rate at each twisted-pair. The other clock is the 100 MHz clock that drives the AXI Stream buses in the system.

Finally, the microprocessor shown in Fig. 4.1 is responsible for running the drivers that configure all the peripherals via AXI, including the developed RoE core. In particular, it configures and initializes the Ethernet MAC and the PHY via the MDIO (see Section 3.3.3), the Ethernet S2MM FIFO, the interrupt controller (by connecting interrupt handlers to each interrupt), the I^2C controller and the external PLL. For the "off-the-shelf" IP modules (all but the RoE and the PLL modules), functional drivers availed by Xilinx were adapted and put together into a single program. For the other two modules, custom drivers were developed in C.

4.2 Half-full Elastic Buffer Architecture

The design guidelines summarized in the end of the previous section prevent any disturbances from the system modules into the synchronization architecture. They also provide a realistic scenario for testing the conditions of the receiver elastic buffers, for which slips should be avoided through proper synchronization. Thus, with these pre-requisites, this section advances into the buffer-based synchronization method.

The half-full elastic buffer architecture aims at simplicity in the implementation. More specifically, the incoming downlink data stream (coming from the fronthaul) into each RRU fills the ingress elastic buffer and, based on the fill level, decisions are taken regarding whether the hardware clock frequency should be increased or decreased. Advantageously, such a process does not require any additional stream to be multiplexed with CPRI Ethernet frames.

The method is applicable at the RRU side, namely the slave in the synchronization hierarchy, which receives RoE frames sent by the BBU transmitter. It is based on the fact that the input packets are generated by a constant bit rate (CBR) stream at the BBU side, where radio data is packetized in the pre-defined average fixed intervals Thus, the RRU is expected to receive new packets within some variance (due to PDV) around the inter-departure intervals, so that the rate in which receive data is written into the elastic buffer can be roughly predicted.

If at any moment the FIFO becomes less than half full by more than the quantity of data packed in one Ethernet frame, it can be inferred that the read (consumption) rate at the RRU is faster than the generation rate at the BBU, namely that $\widehat{\text{clk}}_c > \text{clk}_c$. In this case, the RRU local frequency should be decreased. In the contrary case, when the buffer becomes more than half full by more than the quantity of data in a frame, the local frequency should be increased. This orchestration is carried in a combination of software (in the RoE driver) and hardware in the

"occupancy controller" module developed in VHDL, which is detailed in the sequel.

4.2.1 Occupancy Controller

The purpose of the occupancy controller is two-fold: to prevent drastic empty or full conditions from affecting the downstream/upstream radio chain and to inform the processor about the events that should trigger an immediate clock correction. To accomplish this, the module asserts and de-asserts the signals that enable writes and reads into the buffer and provides an interrupt signal connected to the processor for immediate indications. Additionally, the occupancy controller copies the occupancy levels indicated in the dual-port memory to its internal register, which can be read at any moment by the software running in the microprocessor.

The module divides the buffer into 7 regions within the hardware, as depicted in Fig. 4.2, each corresponding to a threshold established in hardware ("hard threshold"). During the initialization phase, the module waits until the buffer is written with data up to the middle occupancy (50%) level, within the so-called "safe zone". At this point, the module enables the read-side (CPRI side) to fetch the radio data from the buffer and the state machine is considered to be in the "started" state. From this point onwards, any transition in the outwards direction with respect to the safe zone (dashed arrows in Fig. 4.2) triggers an interrupt.

Whenever an interrupt is triggered, the controller stores in its internal register the information regarding the current threshold (mapped into a 3-bit code), the time measured between the transition from the previous threshold level and the current occupancy. Then, the interrupt handler, which was initially connected to the interrupt signal in the driver initialization, fetches this information via an AXI Memory Mapped read transaction. Ultimately, the driver is able to command the appropriate frequency offset correction, as described in the next subsection.

In the "well-behaved" case, the frequency offset will be low enough such that the buffer operates within the safe zone for a relatively long period (in the order of seconds or even minutes). For example, 1 ppm and 100 ppm offsets in the 7.68 MHz clock frequency would lead to a rough[3] prediction of 9 minutes and 9 s, respectively, to leave the safe-zone according to (2.1). In light of this fact, the drivers also provide a mechanism for expediting the synchronization phase. The approach is to constantly poll the occupancy levels kept at the occupancy controller, so that decisions can be taken before the hardware interrupt signals are asserted. In contrast,

[3]As pointed in [68], it is not trivial to determine the instantaneous occupancy of a FIFO, since data insertion and removal events may occur arbitrarily close together in time.

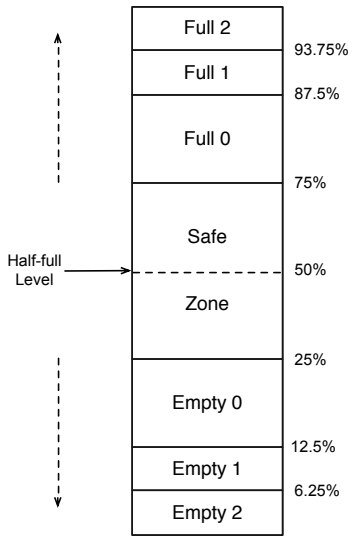


Figure 4.2: Occupancy thresholds implemented in hardware for triggering interrupt signals.

interrupts are left for more drastic ("badly-behaved") conditions that occur when the offset is high enough such that the buffer leaves the safe zone very rapidly.

At the software layer, arbitrary "soft" thresholds can be easily programmed. Since the buffers are designed with 32 bit words in each position, one alternative, used in the design analyzed in the next chapter is to choose single full and empty thresholds at the boundaries corresponding to the number of 32 bit words in an Ethernet frame, as given by the line rate option and the target number of BFs. For example, for line rate option #1, $N_{BF} = 64$ (256 words of 32 bit in total), $\text{clk}_c = 7.68$ MHz and $\text{clk}_E = 100$ MHz, it takes $2,560$ ns to unpack and buffer the entire frame content. However, meanwhile approximately 19 words are read by the CPRI interface, so the occupancy is not expected to reach the aforementioned boundaries. By doing so, any abnormal fill trend is captured fast and the synchronization is expedited.

4.2.2 Local Clock Correction

Once the trend in the buffer occupancy is detected, the drivers can command clock corrections. As explained in the overview of Section 4.1, an external PLL system configurable via I2C is used for this purpose. Then, the accuracy in the synchronization becomes limited by both the accuracy in the offset detection and the correction resolution.

In order to comply with the requirements presented in Section 3.1, the external controllable frequency synthesizer should provide a resolution in the order of parts-per-billion. Provided this is satisfied, it is only necessary to collect all the sets of parameters for its configuration

and avail them in a look-up table form, which the driver can use to apply the target corrections.

The approach adopted in the testbed was to create two look-up tables, one for positive and the other for negative offsets, both with monotonically increasing offset magnitudes. Then, a single signed integer variable is used to track the current frequency level. Whenever the buffer occupancy reaches a full (hard or soft) threshold, the variable is incremented, while when reaching an empty threshold the variable is decremented. Finally, lower layers apply a PLL configuration from the look-up table chosen by the sign of the variable and from the entry indexed by the corresponding absolute value of the variable.

This scheme implies that in normal conditions the system will operate in closed loop and pass through as many configurations as necessary until the frequency becomes sufficiently aligned. For example, if the look-up tables contain parameters spaced by 10 ppb and the system clocks are running with 1 ppm error, then the system would have to transition 100 times until the frequencies are aligned. The downside is that this can be a very slow process, specially because the closer the frequency gets with respect to the master frequency, the longer the time the fill level takes to reach the thresholds that trigger new frequency adjustments.

The adopted Si5324 jitter attenuator from Silicon Laboratories provides a resolution for corrections up to approximately 1 to 3 ppb, depending on the center frequency, so it is adequate for achieving the fronthaul requirements. It is capable of providing a very wide range of corrections due to the range of possible values for its internal dividers and multipliers, which determine the output frequency by the following expression (c.f. the device's datasheet [69]):

$$f_{out} = f_{in} * \left(\frac{1}{N31}\right) \frac{(N2_{HS})(N2_{LS})}{(N1_{HS})(NC1_{LS})}, \tag{4.1}$$

where f_{in} is the input reference frequency and the other parameters are integers within the ranges $1 \leq N31 \leq 2^{19}$, $4 \leq N1_{HS}, N2_{HS} \leq 11$, $1 \leq NC1_{LS} \leq 2^{20}$ and $2 \leq N2_{LS} \leq 2^{20}$.

For the testbed, a batch of the parameters in (4.1) was obtained by the DSPLLsim software available at the manufacturer website, with all the closest matches relative to a target frequency up to a given normalized frequency error. Then, the parameters were filtered off-line to avoid duplicate configurations and to select only the particular sets of parameters that lead to frequencies in intervals corresponding to the target resolution. Finally, the collected sets of parameters were organized in look-up tables and included in the driver.

4.3 PTP-based Synchronization Architecture

The PTP-based architecture introduces important changes with respect to the design in Fig. 4.1. To illustrate the difference, Fig. 4.3 presents an overview of the new Ethernet MAC structure. The remarkable changes are the PTP engine enabled in the Ethernet MAC subsystem, the RTC, also introduced within the subsystem, the additional interrupt signals and the new clock signal of 8 kHz.

Both the RTC and the PTP engine are instantiated within the FPGA fabric, using the implementation of the audio-video bridge (AVB) mode available in the Xilinx Ethernet MAC subsystem [67], which is compliant to the IEEE 802.1AS standard [70]. It should be pointed that the Ethernet MAC subsystem IP provided by Xilinx also allows a standard IEEE 1588 implementation, but in this case the implementation of the RTC and the drivers that implement the entire IEEE 1588 PTP protocol are not available. Thus, the AVB implementation was adopted for the sake of simplicity in the hardware and software design, even though it is not tailored specifically for telecom networks. This choice does not compromise the conclusions that can be drawn from this work with respect to fronthaul synchronization, specially because the most important here with respect to PTP are the algorithms added to the standard implementation.

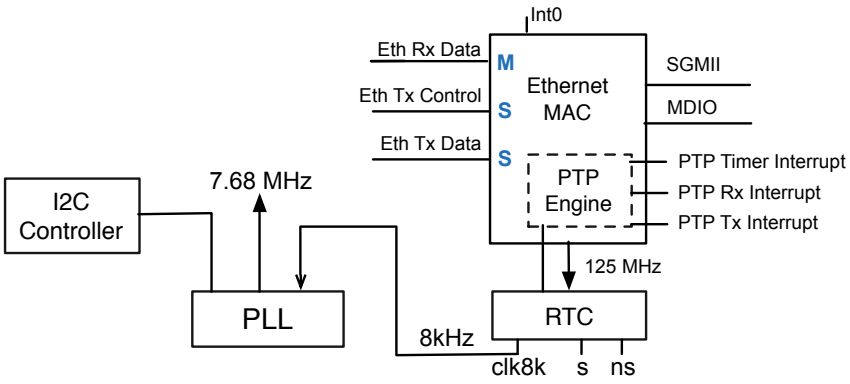


Figure 4.3: PTP-Enabled Ethernet MAC with 8 kHz clock derived from synchronized RTC feeding PLL.

The PTP engine is responsible for the exchange of PTP messages, while the RTC is used by the engine for timestamping and for orchestrating periodical events, such as the transmissions of SYNC messages and the initiation of peer-delay cycles. The *PTP Timer Interrupt* is the signal that coordinates these events, since it is asserted every $1/128$ seconds, which is the minimum period that can be configured for a PTP message. This interrupt signal and the other two introduced in Fig. 4.3 are connected to specific interrupt handlers in software, so that appropri-

ate course of action is taken after a PTP event is signaled, such as responding a PDELAY_REQ with a PDELAY_RESP, computing RTC time offsets and estimating increment values.

In order to accelerate the transmission or reception of PTP messages, the PTP engine is featured with an internal buffer pre-loaded with transmit PTP messages and another circular buffer dedicated to storage of received PTP messages, so that messages are processed in the hardware very close to the point of transmission. When a message is to be transmitted, it is simply loaded from the buffer and time-stamped, while when a message is received, it is stored in the buffer and left there until the driver completes its processing. The drivers are responsible for triggering a PTP message transmission and when a transmission effectively completes, the *PTP Tx Interrupt* signal is asserted to notify the processor. Similarly, when a PTP message is received, the *PTP Rx Interrupt* is asserted.

The PTP engine applies and takes time-stamps from all PTP messages recognized by the EtherType. They are applied or taken at the client interface with the Ethernet MAC, after the Start of Frame Delimiter (SFD) is recognized. Since this timestamping position is in contrast to the standard recommendation of sampling the RTC when the SFD is recognized at the GMII between the PHY and the MAC, the latency within the MAC shall be discounted. This is not a problem for the system, since the MAC latency is fixed by design, so the drivers can effectively subtract the MAC latency (of 80 ns) from the timestamps at received packets and add the MAC latency to the timestamps at transmitted messages (sent through follow-up messages). Ultimately, the timestamps are made available in the transmit and receive buffers.

The RTC module in Fig. 4.3 is dimensioned according to the example used in Section 3.5.3. More specifically, it counts nanoseconds using Q32.20 fixed-point numbers and uses an increment value represented as a Q6.20 number, which provides fine degree of accuracy. The RTC outputs the synchronized nanoseconds and seconds count to the external logic (`s` and `ns` pins) and uses theses values to derive an 8 kHz frequency, labeled `clk8k`, according to the approach described in Section 3.5.4. This synchronized clock signal, then, feeds the Si5324 clock multiplier [69], whose internal registers are configured to output a 7.68 MHz clock from an 8 kHz clock reference[4].

The RTC is driven by the free-running clock of 125 MHz in the Ethernet MAC. Thus, its nominal increment value corresponds to 8 ns, and is represented in Q6.20 by the hexadecimal

[4]The Si5324 configuration of the parameters in (4.1) that converts a 8 kHz clock into 7.68 MHz is $N1_{HS} = 11$, $NC1_{LS} = 60$, $N2_{HS} = 5$, $N2_{LS} = 126720$, $N31 = 1$

`0x800000`. The increment rate is then adjusted as described in Section 3.5.3, leading to the syntonized RTC, which is used by the timestamping logic. In addition to the increment rate, the driver periodically estimates the instantaneous time offset and updates the RTC time offset registers that are summed with the syntonized RTC to produce the synchronized RTC (recall the distinction highlighted in Fig. 3.5). As discussed in 3.5.4, the synchronized RTC provides an extra degree of accuracy, since it corrects any cumulative time errors (wander) generated by a residual frequency offset left in the system due the lack of enough resolution in the frequency correction. For this reason, the synchronized RTC is the one used to derive the `clk8k` signal.

The PTP driver made available by Xilinx, as noted in Section 3.5.1, estimates the link delay every time a peer-delay cycle is completed and estimates RTC time offsets every time a pair of SYNC and FOLLOW_UP messages is received. Then, for every two time offset estimations, it re-estimates the frequency offset using (3.10). In this context, faster convergence is generally achieved by sending SYNC messages at the highest allowed rate of 128 packets-per-second.

4.3.1 Limitations in a Standard PTP Implementation

The most important practical limitation with the standard implementation for time and frequency correction availed in the Xilinx' driver is that every RTC correction is deliberately independent from the previous correction. In another words, at every new PTP message exchange, a new estimation is computed and this value is instantaneously used to update the RTC increment and time-offset registers. Thus, when an estimation presents high deviation with respect to the actual values, the erroneous estimation is still applied as correction to the RTC.

In contradiction to this strategy, the estimations are expected to vary significantly on a cycle-by-cycle basis when transmitter and receiver communicate through a network, specially when legacy switches are used. The most significant source of noise in this case is PDV. The latter in theory is introduced by variations in all delay sources, but since for a fixed network path and fixed packet sizes the propagation and transmission delays are constant, PDV comes from variations in processing and queuing delays, specially the latter due to its larger magnitude [71].

The variations in the queing delays are explained by the fact that Ethernet switches use store-and-forward transmission[5]. In another words, they wait until an entire frame is received before beginning the transmission of its first bit in the appropriate outbound link. Thus, if a PTP packet arrives at the switch right before (piggybacking) an RoE frame, the PTP message

[5]The so-called "cut-through" switching can be used, but it is not standardized.

has to wait until the entire RoE frame transverses the switch. Furthermore, even if a preemption scheme is adopted, when a PTP packet arrives while an RoE frame is already being transmitted, queuing is non-preemptive [71] and, therefore, larger queuing delays are still exhibited.

In comparison to the interval, PTP messages are much shorter, for example SYNC and FOLLOW_UP messages are alone of 44 bytes and, when encapsulated in Ethernet frames, of 70 bytes. This corresponds to a transmission delay of 0.56 µs, which represents only 3.36 % of the inter-departure interval. Hence, as depicted in Fig. 4.4, the PTP message can be located in any position at the idle space within the RoE inter-departure interval.

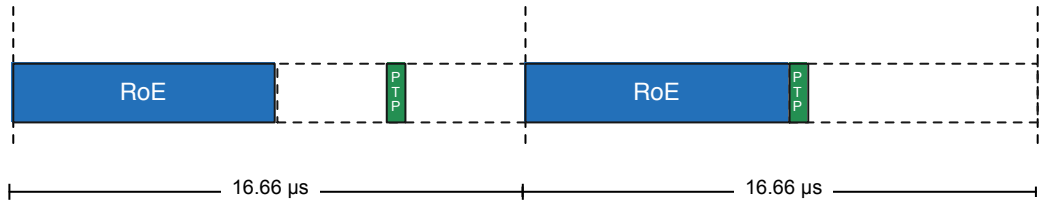


Figure 4.4: Examples of PTP message locations within the RoE inter-departure interval.

Depending on the position of the PTP message in the idle interval, a different delay is experienced. The minimum delay is experienced when the message is triggered early enough with respect to the end of the previous RoE transmission, so that throughout its entire propagation it waits a minimum (or zero, if possible) amount due to store-and-forward of previous RoE frames. Conversely, if the PTP message is triggered right after an RoE frame, it will wait twice the transmission delays of both itself and the RoE frame in all switches along the network.

In light of these facts, it is safe to assume significant PDV affecting PTP messages. The problem is that, as pointed in Section 3.5.1, the delay estimation is based on the assumption that the master-to-slave and slave-to-master delays are symmetrical, which fails due to PDV. Thus, the delay estimations are expected to vary and to deviate from correct values. Consequently, since both the time offset and frequency offset estimations depend on the delay estimation, the two become noisy, and so does the recovered `clk8k` signal.

Furthermore, since the delay estimation is carried by packets (e.g. PDELAY_REQ) other than the one used to compute time and frequency offsets (e.g. SYNC), even if a given delay estimation is perfect, that does not mean the subsequent SYNC messages will experience this exact same estimated delay. Additionally, even if the same packets were used to estimate both the delay and the time-offset, as when using the *delay request-response* mechanism, a non-symmetrical delay due to PDV corrupts the estimation in (3.5) and the error parcels due to the

time offset and due to the actual delay are indistinguishable.

Clearly the PDV effects on the delay estimations should be avoided by some manner. Regarding radio over Ethernet transmission, from Fig. 4.4 it is clear that by keeping RoE frames short, less idle space is left and delay variation is reduced, but overhead increases. Another alternative is to use transparent clocks and their updates on the PTP correction fields that are used to discount the residence times within each intermediate node from the timestamp values. However, since the current work assumes legacy networks, the PDV shall be treated in the endpoint transmitters and receivers using different strategies. The following sections elaborate on the previous art about PDV solutions and present the strategy adopted in this work.

4.3.2 Review of Solutions to Packet Delay Variation

Mitigation of PDV affecting time-sensitive traffic transported over asynchronous networks is generally achieved through three main approaches: circuit emulation, traffic scheduling and frame preemption. The former, can be implemented for example by fixed time slot allocations, such as the scheme in hybrid packet-switched optical networks [72]. Frame preemption, in turn, is a classical quality-of-service (QoS) strategy, but only recently proposed as standard (IEEE 802.1CM) to meet the specific requirements of packet-switched fronthaul streams.

Nevertheless, all such strategies for PDV alleviation require upgrades in the network equipment. For this reason, in the context of PTP, the PDV is not necessarily physically avoided, but instead treated at the endpoints in an algorithm (software) level. Packet pre-selection and filtering applied to PTP estimations are the most widely adopted solutions.

Packet selection consists in using only a subset of the PTP messages to estimate clock offsets. Although it is vendor-dependent, this technique is generally assumed to be employed. In fact, even PTP specifications normally provide PDV performance requirements considering only a small portion of PTP packets. For example, ITU-T G.8261.1 [73] specifies a hypothetical reference model in which only the fastest 1% of the PTP packets must not exceed 150 µs of PDV amplitude. Such schemes usually partition the time into non-overlapping windows and, for each window, select the packets that experience minimum propagation delay in the network for synchronization computations [74], what is referred as sample-minimum filter [75].

A formal statistical treatment of packet selection is presented in [71], where the packet transit time is modeled by a fixed component (transmission and propagation delay) and noise (queuing delay). The work derives the distribution of the transit time random variable (Erlang

shape[6]) and presents a model for the variance in the transit time of the packets that are effectively selected within each window. Then, based on this variance, the validity of a sample-minimum filter is evaluated. It concludes that in some cases (heavy network loads) the distribution of the PDV is such that the optimal selection is instead the mean delay or the packet whose delay is maximum. In this context, the same author proposes in [75] an adaptive scheme that dynamically evaluates the instantaneous optimal criteria (minimum, mean or maximum).

Differently, filtering algorithms applied on the estimations attempt to smoothen the fluctuations in the estimations and corresponding corrections. The rationale is that variations in the oscillator are slow relative to the PTP periods, so instantaneous variations in the estimations are likely to be erroneous. Many filters have been proposed and studied, in some cases applied to the time offset estimations and in others to the frequency offset. For example, [76] evaluates exponential-smoothing, linear programming and Kalman filtering strategies. Similarly, the work in [57] evaluates Kalman filtering applied to frequency offset estimations.

4.3.3 Link Delay and RTC Increment Filters

The system developed in this work filters the estimations of both the one-way delay and the RTC increment values, from (3.5) and (3.10), respectively. The rationale is the fact that these two parameters change relatively slowly. First, it can be assumed than an RoE fronthaul has a fixed topology and is dedicated solely for the purpose of transporting radio. Then, the packet delay has a constant statistical distribution such as the ones shown in [75]. Second, since the RTC increment corresponds to the frequency offset of the local oscillator with respect to the master and the oscillator does not change rapidly, in the steady-state the RTC increment value should be constant for a much larger period than the interval between PTP cycles.

For both the filters, a simple moving-average is adopted. For example, the delay effectively used for subsequent time and frequency estimations is notated $\tilde{d}_k$ and given by:

$$\tilde{d}_k = \tilde{d}_{k-1} + \frac{1}{L}\left[\hat{d}_k - \hat{d}_{k-L}\right], \tag{4.2}$$

where $\hat{d}_k$ is the estimation carried with (3.5) in the k-th peer-delay cycle, which is stored in the moving window. The same is valid for the RTC increment.

As usual in DSP implementations, the moving-average window in (4.2) is sized with a power of 2 length, so that the division by L is performed by a shift operation. It is important to

[6]The Erlang shape is explained by the fact that the queuing delay in each switch has an exponential distribution.

keep the arithmetic efficient, so that corrections are applied as close as possible to the instant that they are detected and faster convergence time is achieved. Furthermore, for a faster lock time, during the transitory phase the mean is updated every time the number of occupied filter taps is a power of 2. To illustrate why this is important, consider the peer-delay mechanism being processed once per second and a filter of length 128. If the latter was not provided, it would take 128 seconds to compute the first mean value and stabilize the PTP estimations.

Specifically for the delay, in addition to the moving-average the structure adopted in the implementation developed for this work can also provide optional moving-minimum and moving-maximum filters. Their importance becomes clear in the context of packet selection.

4.3.4 Packet Selection Rationale

The two aforementioned filters for the RTC increment value and delay are normally sufficient to achieve a stable time alignment accuracy when two PTP endpoints communicate directly. However, when one or more hops are inserted between the two, namely when store-and-forward switches are introduced in the path, the PDV affecting the time-stamped packets introduces significant noise in the time offset estimations and eventually also in the frequency-offset estimations. Since frequency offsets estimations can be filtered (averaged), the strongest impact is in fact in the time-offset estimations.

First of all, when the master-to-slave and the slave-to-master delays are asymmetric due to PDV the sum between the two one-way delays used in (3.5) is noisy and given by:

$$d_{ms} + d_{sm} + \zeta_k = (t_{4,k} - t_{1,k}) - (t_{3,k} - t_{2,k}) \tag{4.3}$$

where ζ_k is the asymmetry random variable.

Furthermore, the one-way delay is a biased random variable that can be modeled as:

$$d = d_{\text{prop}} + d_{\text{trans}} + d_{\text{process}} + d_{\text{queuing}} \tag{4.4}$$

where d_{prop} is the propagation delay, d_{trans} is the transmission delay, d_{process} is the processing delay (assumed constant for simplicity) and d_{queuing} is the random queuing delay. The latter, in turn, can be modeled as:

$$d_{\text{queuing}} = \mu_q + n_q \tag{4.5}$$

where μ_q is the queuing delay bias and n_q is a zero-mean random variable that models the fluctuations in the queuing delay.

By using (4.4) and (4.5), the random asymmetry between two delays ζ_k in (4.3) can be interpreted as the sum of two identically distributed random variables:

$$\zeta_k = 2(\mu_q) + n_q^{\text{fw}} + n_q^{\text{bw}}, \tag{4.6}$$

namely the sum of the queuing noise in the forward and backward paths. Ultimately, the distribution of ζ_k becomes the convolution between the two identical queuing delay distributions. Fig. 4.5 shows the delay asymmetry distribution for two illustrative cases of network load explored in [71], assuming a mean queuing delay of 5 µs.

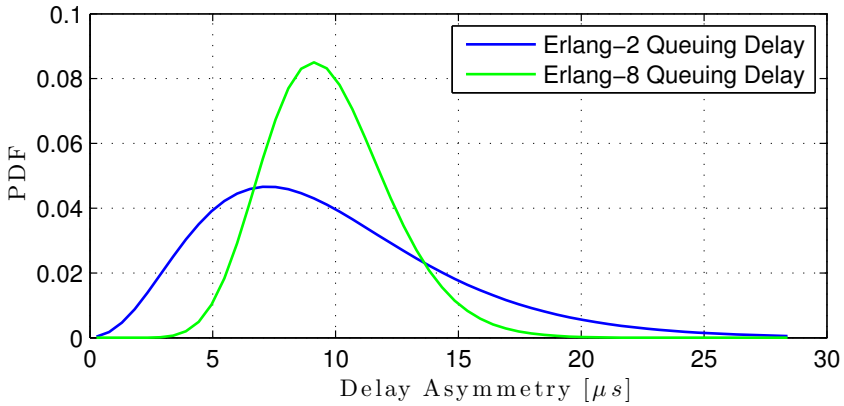


Figure 4.5: Distribution of the delay asymmetry ζ_k for a mean queuing delay of 5 µs, for both Erlang-2 (light network load) and Erlang-8 (heavy load) queuing delay distributions.

Therefore, when the right-hand side of (4.3) is divided by two in the actual one-way delay delay estimation of (3.5), the estimation deviates by half the instantaneous realization of the asymmetry ζ_k:

$$\hat{d}_k = \frac{d_{ms} + d_{sm}}{2} + \frac{\zeta_k}{2}. \tag{4.7}$$

At this point, the value of the random delay estimations to be used in succeeding offset computations must be decided, for example the moving-mean mentioned in the previous section. This choice must be consistent with the packet selection strategy, as clarified in the sequel.

By using the model in (4.4), consider the actual time-offset that should be computed by the estimation carried through (3.7):

$$x_k = t_{2,k} - \left(t_{1,k} + d_{\text{prop}} + d_{\text{trans}} + d_{\text{process}} + d^k_{\text{queuing}}\right), \tag{4.8}$$

where d^k_{queuing} is the realization of the queuing delay that the k-th SYNC was subject to.

Thus, (4.8) reveals that the delay estimation choice determines the pattern in the time offset estimation error. The error can be stated as:

$$\begin{aligned}\epsilon_x &= \hat{x}_k - x_k \\ &= \left(d_{\text{prop}} + d_{\text{trans}} + d_{\text{process}} + d^k_{\text{queuing}}\right) - \hat{d}_k - \gamma_k. \end{aligned} \tag{4.9}$$

For example, the minimum delay estimation could be chosen, namely the delay estimations originated by the realizations of the asymmetry random variable ζ_k with minimum magnitude. Then, further assuming the minimum queuing delay is zero (e.g. in a scenario similar to the one discussed for Fig. 4.4), after sufficient observation a moving-minimum delay estimation $\hat{d}_k$ should approach $d_{\text{prop}} + d_{\text{trans}} + d_{\text{process}}$, yielding:

$$\epsilon_x = d^k_{\text{queuing}} - \epsilon_{d,\text{min}} - \gamma_k, \tag{4.10}$$

where $\epsilon_{d,\text{min}}$ is the error associated to the estimation $\hat{d}_k$ of the minimum delay in the system:

$$\epsilon_{d,\text{min}} = \hat{d}_k - \left(d_{\text{prop}} + d_{\text{trans}} + d_{\text{process}}\right). \tag{4.11}$$

Ideally, the correction field γ_k should perfectly correct the queuing delay realization in each transmission, so only the error in the estimation of the minimum delay would be left. However, when legacy networks are assumed, the correction field is not updated along the network, so that the time-offset estimation errors in (4.10) can be expressed as:

$$\epsilon_x = d^k_{\text{queuing}} - \epsilon_{d,\text{min}}. \tag{4.12}$$

Finally, the rationale of packet selection can be clearly stated. When a non-overlapping window of packets (and their corresponding timestamps) is accumulated and the time-offset is computed based on each pair of t_1 and t_2 time-stamps in the window, each individual estimation is subject to a distinct error. Then, the objective of packet selection is to use only the single estimation in the window that is interpreted as the least erroneous for correcting time-offset and frequency offset. For each window, following the selection a single time-offset is estimated and every two non-overlapping windows one frequency offset is estimated.

In general, the selection strategies aim to minimize the time-offset fluctuations by selecting in such a way that the only noise left is the one between the delay estimation and the target delay (minimum, mean or maximum). For example, a peculiar characteristic of the delay estimation choice of the minimum is that it leaves a non-negative queuing delay parcel d^k_{queuing}

in the error of (4.12). Then, from (4.12), to maximize the signal-to-noise ratio in the time offset estimation, the minimum estimation within the window shall be selected. If the minimum queuing delay is effectively zero, the error in the selected time-offset approaches $\epsilon_x = -\epsilon_{d,\text{min}}$.

By using the model in (4.5), it can be similarly shown that the fluctuations associated with the windowed time-offset estimations when the mean delay is chosen is:

$$\epsilon_x = n_q^k - \epsilon_{d,\text{mean}}, \tag{4.13}$$

where n_q^k is the k-th realization of the zero-mean random queuing fluctuation and $\epsilon_{d,\text{mean}}$ is the error associated with the mean delay estimation adopted in time-offset computations, given by:

$$\epsilon_{d,\text{mean}} = \hat{d}_k - \left(d_{\text{prop}} + d_{\text{trans}} + d_{\text{process}} + \mu_q\right). \tag{4.14}$$

In this case, since n_q^k is by definition zero-mean, the optimal selection within the window is the mean of the time-offsets, which tends to leave $\epsilon_x = -\epsilon_{d,\text{mean}}$.

4.4 Comparison between the Architectures

The buffer-based architecture is a simplistic approach and has the advantage of not requiring any actual stream to be sent. However, it has several fallacies. One disadvantage is that it provides much slower convergence time with respect to PTP, since the threshold levels need to be reached in order to trigger a clock correction, and often many corrections will be necessary before stabilizing in the "locked" frequency. Secondly, the buffer-based method does not provide phase or time alignment, so it can not satisfy the radio requirements in its entirety.

Another problem with the buffer-based method is the fact that it introduces a change in the PLL for every correction. Thus, every time the PLL has to re-achieve lock and if its transitory is long, the receive elastic buffer may be subject to overrun or underrun while locking. As a result, the buffer-based method is less suited for online frequency correction.

Yet another problem in the buffer-based method is that its response to small frequency variations is too slow. Any small instantaneous deviations in the synthesized frequency are only corrected after the buffer occupancy reaches a threshold. Then, even with high resolution availability in the PLL, the target accuracy may remain unsatisfied during a long period after sudden oscillatory changes (e.g. due to temperature).

In contrast, the PTP-based method is expected to overperform in all the above aspects. It has low convergence time, it aligns the frequency without undergoing unstable states or requir-

ing PLL reconfigurations and it responds very rapidly to sudden clock changes. Nevertheless, the scheme has its own limitations, mainly relative to the the sub-nanoseconds resolution in the increment value and the PDV noise affecting the time-offset estimations.

For example, considering the RTC is clocked at 125 MHz, the 20 bits employed for sub-nanoseconds bits lead to a minimum frequency resolution of approximately ± 119 ppb, which is not within the requirements of Section 3.1. However, as pointed recurrently in this chapter, the residual frequency offset is solved by the time-offset corrections. The latter, in turn, is affected by PDV, so in the end PDV becomes the strongest impairment in the PTP-based architecture.

Chapter 5

Testbed Results

This chapter evaluates the two synchronization architectures presented in Chapter 4. The approach adopted for assessment of the results is both based on measurements and on captures of values kept internally to the hardware. More specifically, the results are based on the the buffer occupancy levels, which are affected by frequency mismatches; the actual frequency offset measurements; spectrum analysis of the recovered clocks and in the specific case of PTP, the jitter introduced in the time-aligned recovered clock. The measurements are carried using two equipments from Agilent Technologies, the Infiniium MSO 9104A oscilloscope capturing with a sampling rate of 10 GSample/s and the EXA N9010-A signal analyzer.

This chapter is organized as follows: Section 5.1 introduces the two evaluated network topology scenarios. Section 5.2 presents the results obtained with the buffer-based method. Then, Section 5.3 presents the synchronization results obtained by using PTP and discusses the PDV introduced due to store-and-forward of the RoE stream. Finally, Section 5.4 draws remarks contrasting the obtained results with fronthaul requirements.

5.1 Test Scenarios

In this chapter, two distinct scenarios are evaluated. In both cases, the BBU and the RRU are implemented in separate Xilinx VC707 boards (with Virtex-7 FPGA), but the two FPGAs are programmed with the exact same bitstream and distinctions are configured solely in software. The 1000BASE-T Ethernet PHY used for the test is provided by the on-board Marvel 88E1111 chip and Cat5e cables are used in each link.

The first scenario adopts BBU and RRU endpoints communicating through a direct Eth-

ernet link, as illustrated in Fig. 5.1. It is not a practical scenario, but it is evaluated only with the purpose of providing a baseline that resembles an ideal condition in which PDV is negligible. In contrast, the second scenario consists in the communication between the BBU and the RRU through 1 hop, as illustrated in Fig. 5.2. This scenario introduces the PDV that is normally present over packet switched networks, with enough strength to impose limitations in the PTP method, specially due to the constant bit rate RoE stream. A more practical scenario with several hops is left for future work.

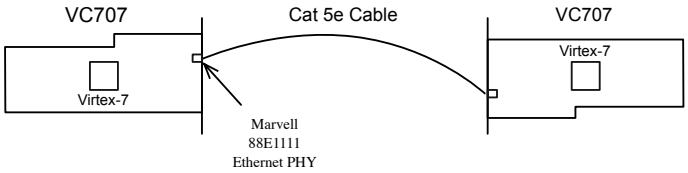


Figure 5.1: Scenario 1: BBU and RRU communicating directly.

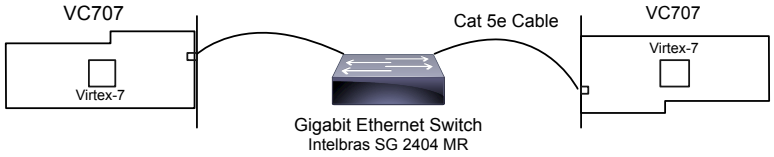


Figure 5.2: Scenario 2: BBU and RRU communicating through one hop.

For both scenarios and both synchronization schemes, the tests are carried using the same system-level parameters with respect to the RoE interface, except for a few forthcoming modification adopted for PTP. The parameters are summarized in Table 5.1.

5.2 Buffer-based Scheme

The buffer-based method is evaluated using the specific parameters summarized in Table 5.2, which are related to the Si5324 PLL. In particular, $4,000$ sets of configuration parameters to determine the output frequency as in (4.1) were collected and look-up tables organized in the driver. The sets allowed a resolution of 10 ppb and a correction range of ± 20 ppm with respect to the nominal value.

Parameter	Value
CPRI Clock (clk_c)	7.68 MHz
Corresponding LTE BW	5 MHz
IQ Sample Width (W)	30 bits
CPRI Line Rate Option	#1 (614.4 Mbps)
Number of BFs (N_{BF})	64
Ethernet AXI Clock (clk_E)	100 MHz
Nominal Ethernet inter-departure (T_E)	$16,666$ µs
Alternating $M_{E,1}$	1666
Alternating $M_{E,2}$	1667

Table 5.1: Parameters adopted in the system where synchronization methods are tested.

Parameter	Value
Si5324 Correction Range	± 20 ppm
Si5324 Resolution	10 ppb

Table 5.2: Parameters adopted in the buffer-based synchronization scheme.

5.2.1 Results for Scenario #1: Direct Link

Fig. 5.3 presents the buffer occupancy levels while the method is walking through all the available configurations in the look-up table, starting at the moment the software is initiated in the processor. The process is very long and requires hours due to the fine-grained resolution of 10 ppb. Furthermore, the closer the configuration is to the actual instantaneous frequency offset, the longer the time to transition to another configuration. Once a configuration closer to the steady state is reached, the buffer levels become stable as shown in Fig. 5.4. From this point onwards, new reconfigurations are occasionally observed due to variations in the room temperature and corresponding deviations in the synthesized frequency.

The peaks observed in the occupancy levels can be interpreted as the point where the PLL is re-configured. The reason why they occur is because of the transitory state in the PLL. Whenever a PLL re-configuration is triggered, the system is programmed to sleep for approximately 1 minute, so that during the lock of the PLL[1] the occasional abnormalities in the buffer do not trigger other re-configurations. Meanwhile, RoE traffic continues to be exchanged unstoppable

[1]The PLL adopted in this work is a very low bandwidth PLL for phase noise attenuation. Thus, the transitory time required for lock after a re-configuration is relatively long.

and the buffer levels are affected by the transitory frequencies in the lock phase, which can be higher (overshoot) or lower (undershoot), as seen in the up and down peaks. At the end of the sleep time (PLL reconfigured), the RoE system is reset, so the buffer goes back to half-full state.

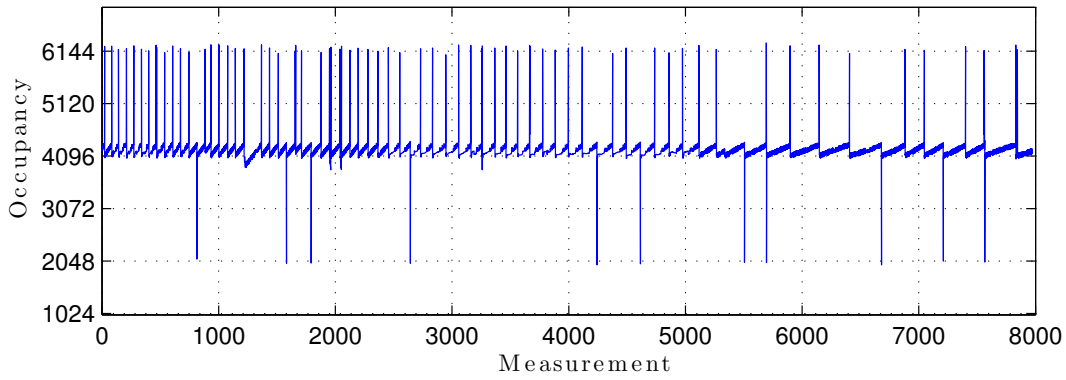


Figure 5.3: Buffer occupancy levels while the clock is searching the appropriate PLL configuration with the buffer-based method. Approximately 120 minutes of capture.

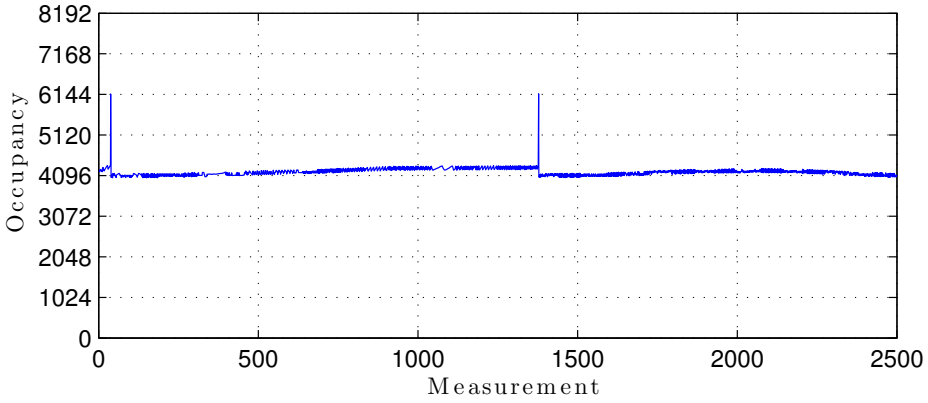


Figure 5.4: Buffer occupancy levels when a nearly stable configuration is achieved. Approximately 15 minutes of capture.

It should be noted that the horizontal scale in both occupancy plots and the ones that follow in this chapter solely indicates the occupancy measurement number and does not follow a perfect linear time-scale. The reason is two-fold: first because the measurements were captured from printings carried by the software running in the processor and transmitted through the UART interface to the host computer, but some prints are occasionally preempted by other characters and not adequately captured. The script capturing the measurements filters such cases and only stores the successful captures. Nonetheless, such failure cases are rare and the

scale gives a reasonable indication of the relative time taken between occupancy measurements, specially because the period between each measurement is nearly the same (approximately 400 milliseconds). Secondly, in the particular case of the buffer-based method, the horizontal time-scale is also not linear because of the sleep times after each PLL re-configuration.

Fig. 5.5 presents the spectrum of the BBU clock and the clock recovered at the RRU side measured in the spectrum analyzer. On the same plot, two markers corresponding to the clock spectrum peak are overlapped, one of the BBU (Mkr 1) and the of the RRU (Mkr 2). Underneath the spectrum measurement, a table lists the values corresponding to each marker. Note a frequency offset of approximately 6 ppb is observed between the FPGAs. This is as expected, since the syntonization resolution is of 10 ppb and the actual frequency error may be closer to the configuration points (as in this case).

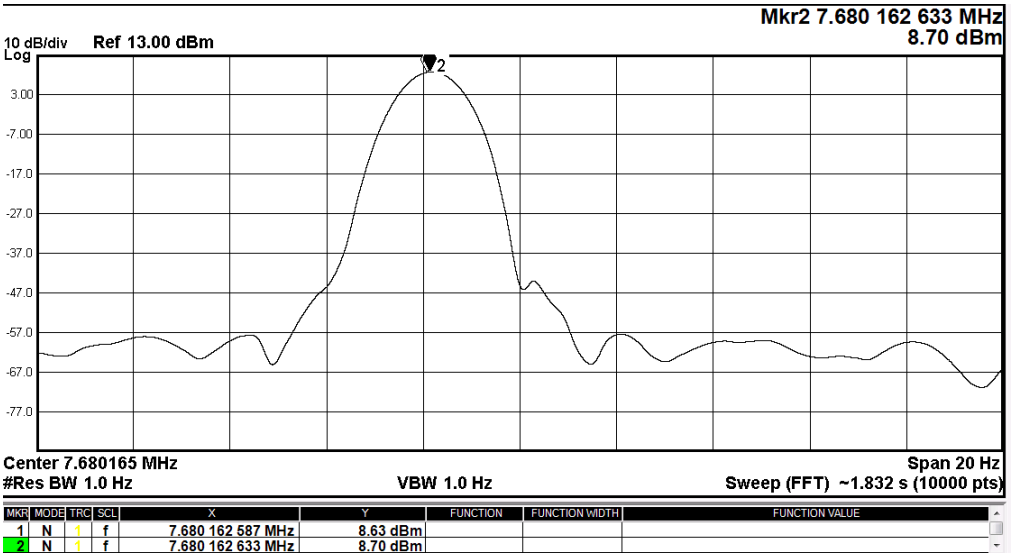


Figure 5.5: Spectrum peak measurements carried with the buffer-based syntonization over direct link for both the BBU (MKR 1) and the RRU (MKR 2).

5.2.2 Results for Scenario #2: 1-hop

When the buffer-based method is employed in the 1-hop topology, advantageously the pattern followed by the occupancy levels is very similar with respect to the one obtained with a direct link. Fig. 5.6 presents the capture with a initial guess computed offline based on previous experiments. The crystal oscillator goes through a change of temperature and during this process several re-configurations are carried. In the end, when the temperature stabilizes, the clock

re-adjustments become sparse again and the buffer remains half-full.

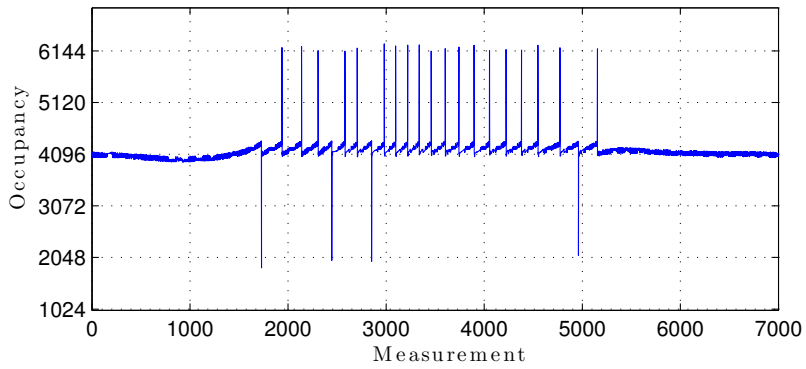


Figure 5.6: Buffer occupancy while the clock is being syntonized with the buffer-based method through 1 hop.

Hence, it can be stated that the method is robust with respect to varying queuing delays in store-and-forwards switches. Fig. 5.7 presents the spectrum of both the the master (BBU) and slave (RRU) clocks in this scenario. Note a frequency offset of approximately -3.6 ppb was observed at the RRU with respect to the BBU.

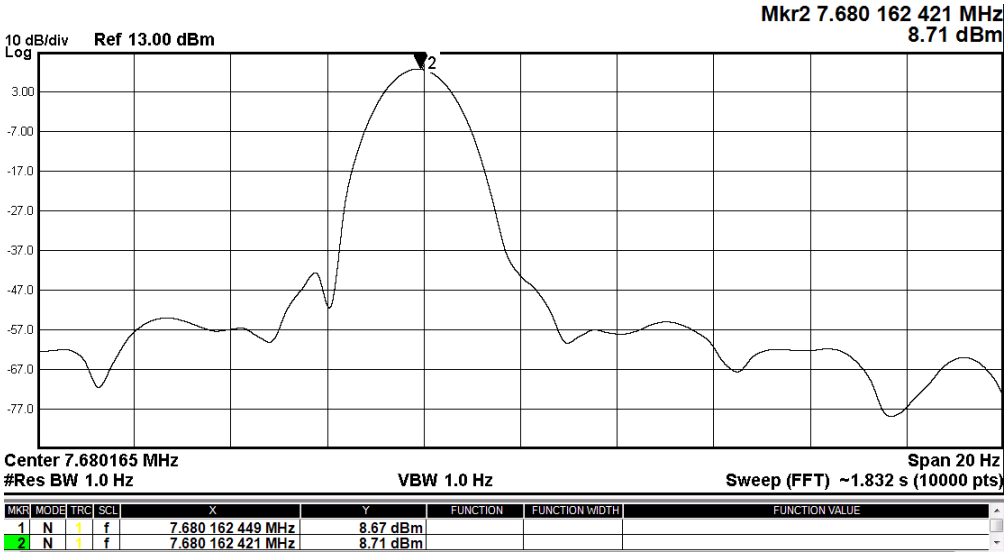


Figure 5.7: Spectrum peak measurements carried with the buffer-based syntonization over 1 hop for both the BBU (MKR 1) and the RRU (MKR 2).

5.3 PTP-based Scheme

In the tests carried for the PTP-based synchronization scheme, the FPGA representing the BBU operates as PTP master and the FPGA representing the RRU operates as PTP slave. Table 5.3 summarizes the PTP parameters adopted for tests. The noteworthy parameters concern the lengths employed for the moving-average filters presented in Section 4.3.3.

Parameter	Value
RTC Increment Representation	Q6.20
Peer-delay Rate	8 packets-per-second
SYNC Rate	128 packets-per-second
RTC Increment Filter Length	128
Delay Filter Length	256

Table 5.3: Parameters adopted in the tests with the PTP-based method.

5.3.1 Results for Scenario #1: Direct Link

In the case of the direct link, the distribution of the delay is expected to be nearly constant around the sum between the propagation and the transmission delays. This can be observed from the histogram of the delay estimations carried at the PTP master, which is shown in Fig. 5.8. A mean delay estimation of 1.659 µs was observed with a standard deviation of only 6.3 ns.

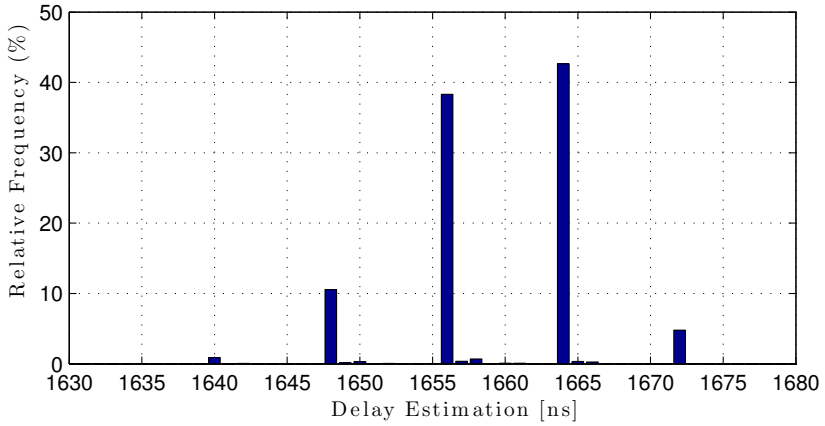


Figure 5.8: Histogram of the link delay estimations in the direct link scenario with PTP.

Due to the nearly absent PDV, the PTP method is expected to operate reliably. In fact,

this can be seen from the buffer occupancy levels shown in Fig. 5.9. Note the buffer remains consistently half full over the entire measurement period. This is provided from a combination of both the successful recovery of the optimal RTC increment value at the slave and the reliable time-offset corrections that are seamlessly feasible in the absence of PDV.

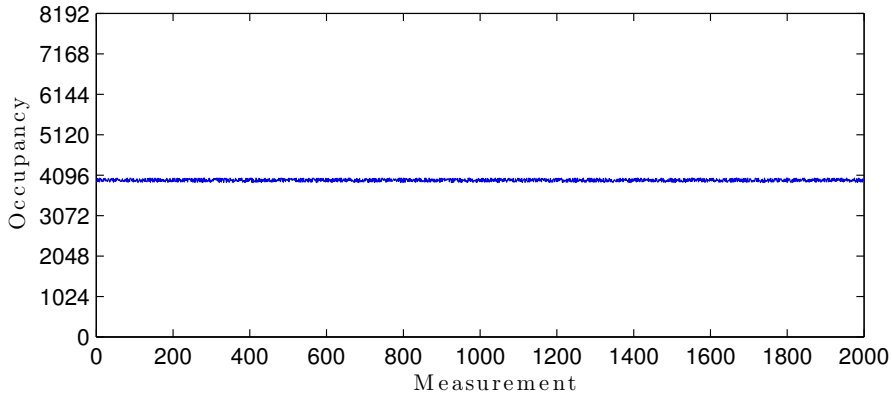


Figure 5.9: Buffer occupancy with PTP in the direct link scenario.

The achieved frequency accuracy can be observed from the measured master and slave clock spectra, respectively presented in Fig. 5.10 and Fig. 5.11, where the peak frequency value is highlighted in a red box. According to the measurements, master and slave clocks are aligned in frequency with approximately a remarkable 2.6 ppb offset. However, the inherent jitter in the slave introduces significant phase noise in the RRU recovered clock, which can be observed from the spectrum "smear" in Fig. 5.11.

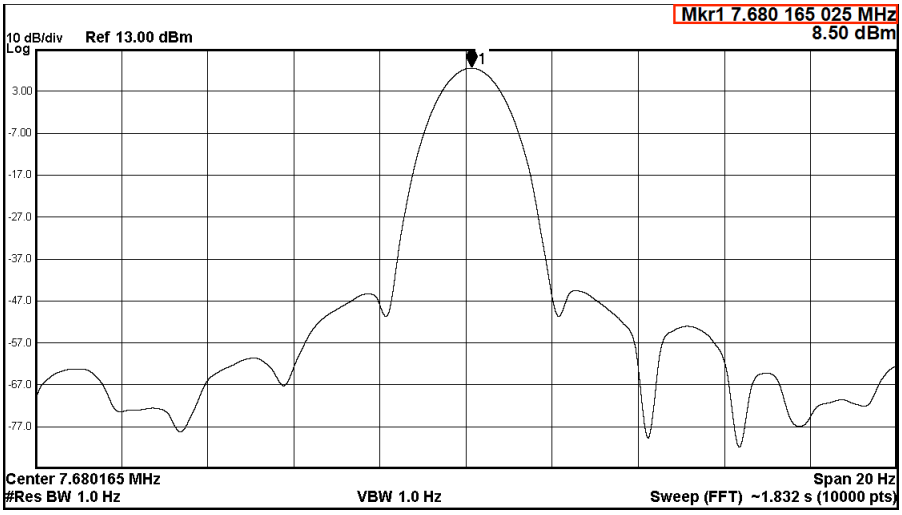


Figure 5.10: Spectrum of the PTP master clock in the direct link scenario.

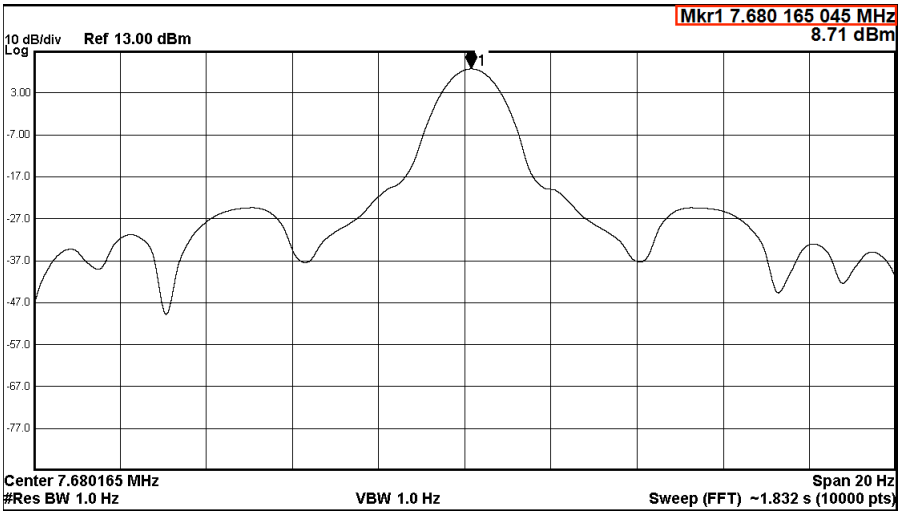


Figure 5.11: Spectrum of the recovered PTP slave clock in the direct link scenario.

To illustrate the accuracy with respect to time-alignment, Fig. 5.12 shows the two clocks in the oscilloscope with infinite persistence after a capture of minutes. The clock at the top is the one used as trigger source and coming from the FPGA representing the BBU, while the one at the bottom of the display is the clock recovered at the RRU via PTP. The colors indicate the number of occurrences of each line, so that it is possible to infer the jitter affecting the recovered clock. Since the tiles represent 20 ns, a jitter of approximately ± 10 ns is observed.

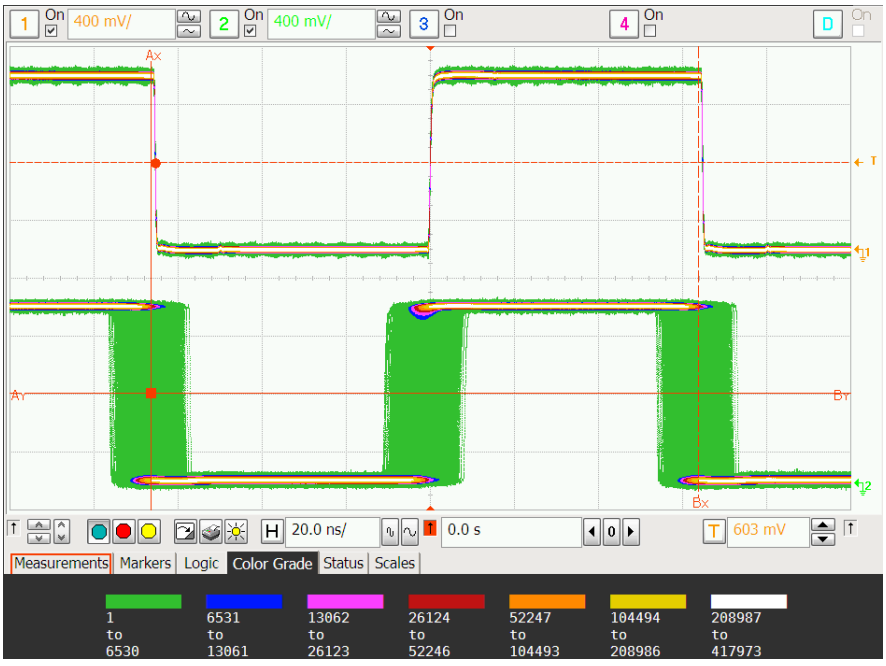


Figure 5.12: Time alignment with the PTP-based method over direct link.

5.3.2 Results for Scenario #2: 1-hop

When the PTP-method is evaluated in the 1-hop scenario, packet selection guarantees that the frequency alignment continues to be very accurate, as observed from the BBU and RRU spectra in Fig. 5.13 and 5.14, respectively. The peak frequency of the BBU was measured at 7.680166843 MHz and the peak at the RRU was measured in a particular instant[2] at 7.680166863, which implies an offset of 2.6 ppb. Furthermore, by comparing Fig. 5.14 to the the RRU clock spectrum in the direct case (Fig. 5.11), it can be seen that phase noise is even weaker in the 1-hop case, namely that the spectrum smear is narrower.

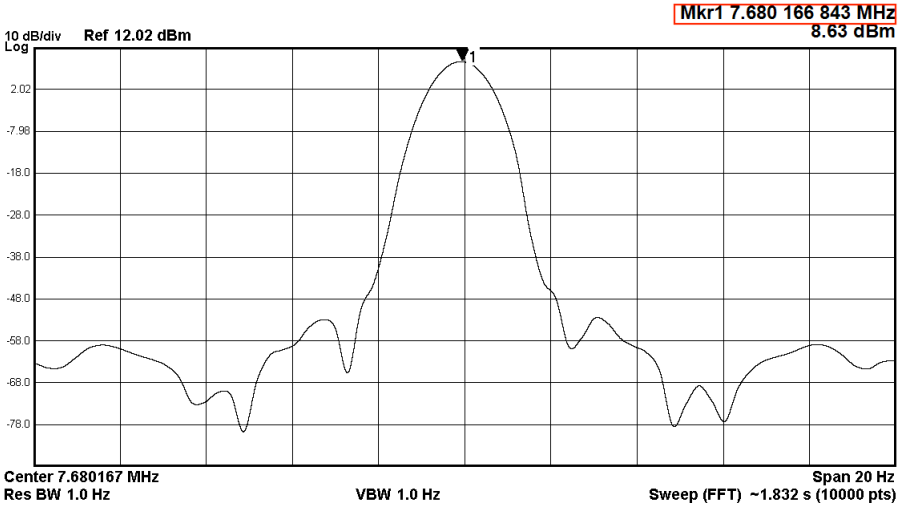


Figure 5.13: Spectrum of the clock at the BBU side.

Accordingly, with the packet selection strategy, the buffer occupancy was kept at the desired half-full state. Fig. 5.15 presents a capture of the occupancy in the master side and contrasts it to the one that results when packet selection is not employed, only the moving-average filters that were used the in direct-link scenario. Note that in the absence of packet selection (red curve), the time and frequency offset estimations deviate strongly due to PDV and consequently the buffer constantly suffers underrun[3] (it could have been overrun too). In contrast, packet selection effectively improves the signal to noise ratio in the time-offset estimations, resulting in reliable occupancy levels through the entire capture.

Finally, the jitter was measured to be higher than one period of the clock signal, so that a persistence plot with the 7.68 MHz clock is not informative (it would be entirely filled). Hence,

[2]The peak in the spectrum keeps moving due to constant corrections, but it remains within desirable accuracy.

[3]The reason why the occupancy goes back to half the buffer length when underrun is recognized is due to a protection mechanism developed to the hardware.

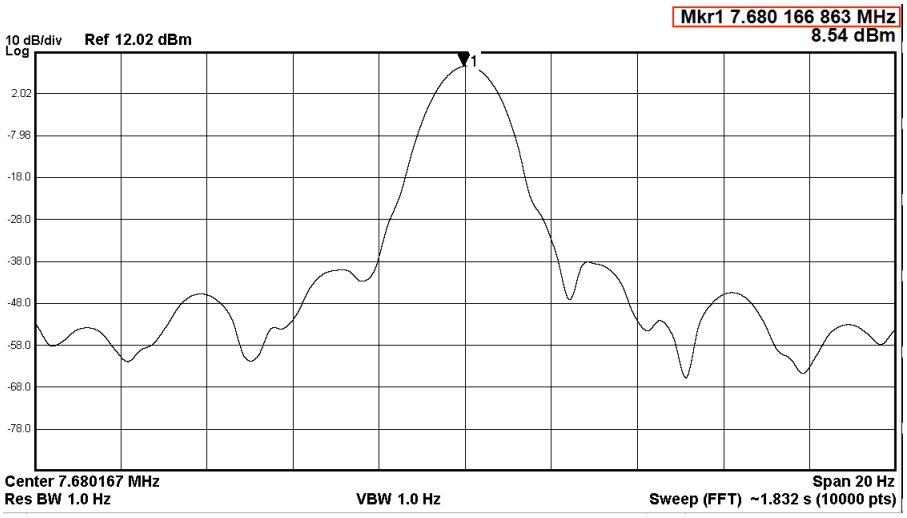


Figure 5.14: Spectrum of the clock recovered with PTP at the RRU side through 1-hop.

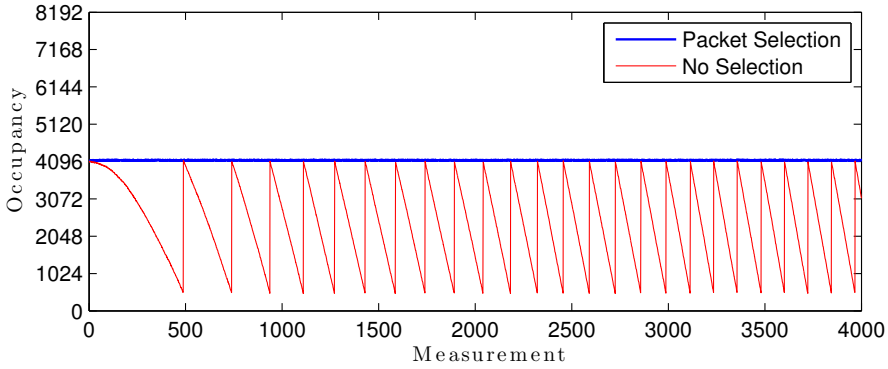


Figure 5.15: Buffer occupancy with PTP in over 1 hop.

instead the jitter is analyzed using the original 8 kHz clock signal that drives the Si5324 (see Fig. 4.3), which has a much higher period (125 µs) and allows proper observation of the jitter.

Fig. 5.16 presents a histogram of the TIE measurements computed between the center of the master clock (labeled Ax) and the edge of the recovered clock (labeled Bx). Note the difference presented a peak-to-peak variation of approximately 200 ns and a mean yielding a jitter close to ±100 ns. However, note such peak variations occurred very few times, only 52 hits, which can be compared for example with the 9738 hits in the mode. In fact, this can also be seen from the standard deviation which was measured at 39 ns.s A similar jitter of ±110 ns is observed in the infinite persistence plot of Fig. 5.17, which was captured for several minutes as seen from the number of realizations in the white color grade (one million).

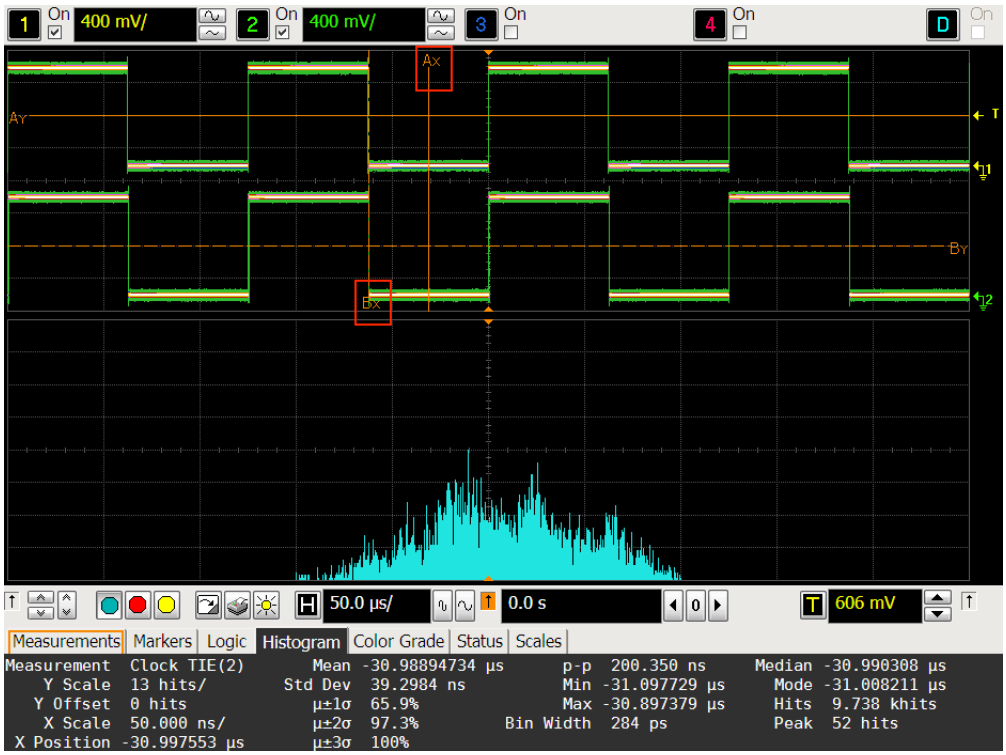


Figure 5.16: Time interval error at the RRU side in the 1-hop PTP scenario.

5.4 Final Remarks

According to the results, the buffer-based method is capable of achieving high frequency alignment accuracy, but as observed before, its convergence and response to variations are too slow, and it may introduce radio data loss during clock configuration. In contrast, PTP provides

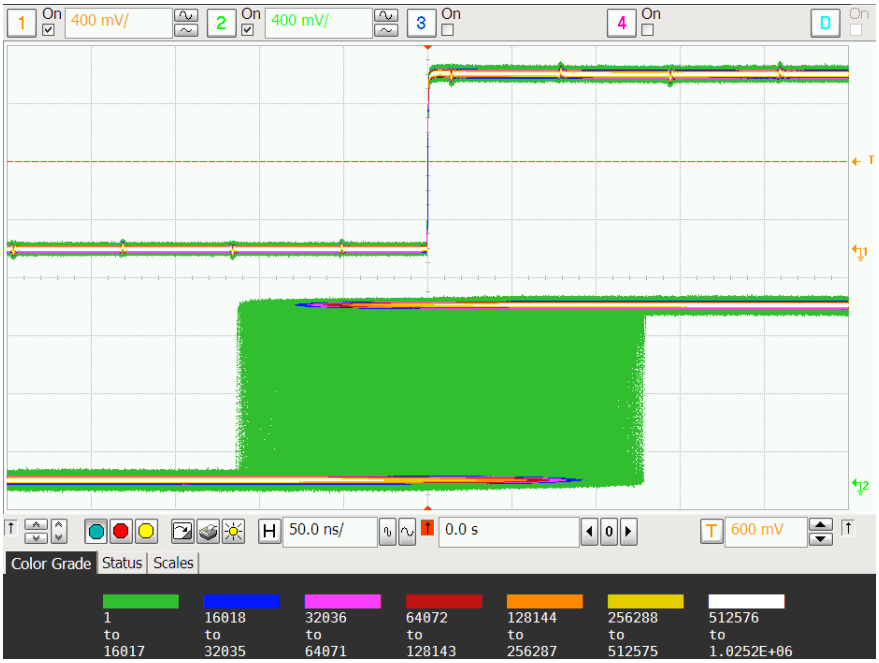


Figure 5.17: Time alignment with the PTP-based method over 1 hop.

high accuracy and fast response, but requires an extra stream that suffers significantly from the PDV introduced by the own RoE stream.

Most of the accuracies achieved in the preceding results obey the requirements listed in Section 3.1. First, the frequency alignment error is below the 16 ppb budget. Second, for PTP that also provides time alignment, the jitter is within the 65 ns required for MIMO or Tx diversity transmission for the direct link and in the case of the 1-hop path, it is likely satisfy the 130 ns requirement for intra-band contiguous carrier aggregation. The problem remaining in a scenario with significant PDV is that even with the protections provided by the packet selection strategy, sudden phase deviations can still occur and reach the peak jitter. The latter is expected to be a consequence of the fact that the evaluated packet selection strategy gives full weight to each new time-offset estimation carried after a packet selection. Therefore, one possibility to avoid such step transitions is to also average the "step-by-step" fine-grained corrections computed at the end of each window, for example using exponential smoothing.

It should also be noted that the scenario explored in this chapter are illustrative cases. In practice, more complex network topologies with several switches can be expected. Hence, the methods presented in this work should be further evaluated on more drastic conditions.

Chapter 6

Conclusions

This work provides a comprehensive view of two clock synchronization solutions applicable to Ethernet-based fronthaul: a zero bandwidth buffer-based method and a PTP-based method. It thoroughly describes their actual design in a combination of hardware and software, evaluates them and discusses their limitations. Additionally, this work shows the particular considerations regarding the radio over Ethernet encapsulation in hardware towards compliance with the synchronization schemes. For example, it was first shown that the inter-departure interval employed for the frames carrying radio data should be carefully designed to prevent additional apparent frequency offsets other than the ones truly present in the oscillators. Then, it was shown that PTP and RoE stream can be seamlessly multiplexed, but the latter strongly influences the PDV that affects the former if not properly dimensioned.

According to theoretical expectations and the evaluation presented in this work, it can be concluded that the PTP-based method is more accurate, converges faster and presents quicker response to any small clock deviations. Based on the results acquired in the simple but illustrative network topologies studied in this work, it can be concluded that PTP is a feasible solution for Ethernet-based fronthaul. It only requires careful hardware and software considerations to avoid its main impairment introduced by PDV.

Regarding software, it was shown that a combination of selection and filtering techniques known in the PTP literature (mostly based on simulations) is sensible for avoiding the noise introduced by PDV in the time and frequency offset estimations. First, the work illustrated the importance of filtering the RTC increment value and the delay estimations to keep jitter constrained. Additionally, based on the derivations presented for packet selection strategies, it was shown that the delay chosen for window-wide time-offset estimations should be compliant to

the packet selection strategy and the latter, in turn, made in observance to the network statistics.

Meanwhile, with respect to hardware, aside from the proper RoE frame length choice, another feature that can be concluded to aid against PDV is the sub-nanoseconds resolution allowed in the RTC increment value. A high increment resolution alleviates the burden of the time-offset corrections that are required to compensate any residual frequency offset after the best increment value is found through the PTP estimations. Then, since a slower wander implies time-offset estimations with reduced magnitude, the probability of estimating and applying severely erroneous time-offset estimations is reduced. For example, in the hardware evaluated in this work, the RTC increment allowed a relatively coarse resolution of approximately 120 ppb. For this reason, the time-offsets presented relatively high magnitude and the system became more prone to errors in the presence of PDV, as observed for the 1-hop network.

6.1 Future Work

Future extensions of this work shall investigate the architectures over more practical network topologies such as the ones described in ITU-T G.8261 [65]. For example, a topology with a chain of switches and background traffic injected on each hop towards the next hop. Such cases are expected to present much stronger PDV, so their evaluation allows studies of the algorithms under stress conditions.

Two other points left for future work are the extension in the number of subnanoseconds bits up in the RTC increment value and the protection to abnormal phase changes eventually observed after the packet selection enters the so-called locked state. The former should be extended to approach a resolution closer to the LTE frequency accuracy budget of ± 16 ppb. The latter should maintain the clock recovered in the presence of PDV reliably and steadily under the jitter limits imposed by LTE requirements for joint-processed transmissions.

Finally, in the future more tests should be performed using further equipments for system evaluation. For example, tests using Stratum-1 clock as reference in the network should be carried and PTP metrics should be acquired with specific measurement equipments for assessment of packet-based synchronization. Besides, the system should be evaluated using transparent clocks to contrast the performance that can be achieved with PTP support along the network.

Bibliography

[1] S. Laboratories, "SyncE and IEEE1588: Sync Distribution for a Unified Network," AN 420, 07 2014.

[2] I. Instrumentation and M. Society, "IEEE 1588-2008: Standard for a precision clock synchronization protocol for networked measurement and control systems," jul, 2008.

[3] P. Chanclou, A. Pizzinat, F. Le Clech, T.-L. Reedeker *et al.*, "Optical fiber solution for mobile fronthaul to achieve cloud radio access network," in *Future Network and Mobile Summit (FutureNetworkSummit), 2013*, July 2013, pp. 1–11.

[4] I.-H. Kim, "Why do we need antenna-integrated RRH (Remote Radio Antenna, RRA)?" NETMANIAS TECH-BLOG, mar 2015.

[5] C.-L. I, J. Huang, R. Duan, C. Cui, J. Jiang, and L. Li, "Recent progress on C-RAN centralization and cloudification," *Access, IEEE*, vol. 2, pp. 1030–1039, 2014.

[6] C. Lu, M. Berg, E. Trojer, P.-E. Eriksson, O. V. T. Kim Laraqui, and H. Almeida, "Connecting the dots: small cells shape up for high-performance indoor radio." Ericsson Review, dec, 2014.

[7] J. Markendahl and A. Ghanbari, "Shared smallcell networks multi-operator or third party solutions - or both?" in *Modeling Optimization in Mobile, Ad Hoc Wireless Networks (WiOpt), 2013 11th International Symposium on*, May 2013, pp. 41–48.

[8] 3GPP, "Coordinated Multi-Point Operation for LTE," 3rd Generation Partnership Project (3GPP), TR 36.819, sep 2011.

[9] S. Namba, T. Warabino, and S. Kaneko, "Bbu-rrh switching schemes for centralized ran," in *7th International Conference on Communications and Networking in China*. IEEE, 8 2012, pp. 762–766.

[10] Ericsson, "Ericsson mobility report," P. Cerwall, Ed., jun 2015.

[11] J. G. Andrews, S. Buzzi, W. Choi, S. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5g be?" *Arxiv preprint*, pp. 1–17, 2014.

[12] M. Dohler, R. W. Heath, A. Lozano, C. B. Papadias, and R. A. Valenzuela, "Is the phy layer dead?" *IEEE Communications Magazine*, vol. 49, no. 4, pp. 159–165, 2011.

[13] N. Bhushan, J. Li, D. Malladi, R. Gilmore, D. Brenner, A. Damnjanovic, R. T. Sukhavasi, C. Patel, and S. Geirhofer, "Network densification: The dominant theme for wireless evolution into 5g," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 82–89, 2014.

[14] D. Sabella and P. Rost, "Ran as a service: Challenges of designing a flexible ran architecture in a cloud-based heterogeneous mobile network," *Future Network and Mobile Summit (FutureNetworkSummit)*, pp. 1–8, 2013.

[15] Y. Beyene, R. Jantti, and K. Ruttik, "Cloud-ran architecture for indoor das," *Access, IEEE*, vol. 2, pp. 1205–1212, 2014.

[16] Y. Zhiling, Z. Xiaowen, Y. Fei, X. Jiyan, H. Zongying, Z. Xu *et al.*, "White paper of next generation fronthaul interface," june 2015.

[17] R. M. Rao, "Cloudrans and fronthauling." Xilinx Inc., oct 2015. [Online]. Available: http://eecatalog.com/4g/2015/10/01/cloudrans-and-fronthauling/

[18] "Common Public Radio Interface (CPRI) specification v6.1," http://www.cpri.info/downloads/CPRI_v_6_1_2014-07-01.pdf, jul, 2014.

[19] "Open base station architecture initiative (OBSAI) specification v2.0," http://www.obsai.com/specs/OBSAI_System_Spec_V2.0.pdf, Apr. 2006.

[20] "Open radio equipment interface (ORI) release 4," http://www.etsi.org/deliver/etsi_gs/ORI/001_099/001/04.01.01_60/gs_ORI001v040101p.pdf, 2006.

[21] A. Pizzinat, P. Chanclou, F. Saliou, and T. Diallo, "Things you should know about fronthaul," *Lightwave Technology, Journal of*, vol. 33, no. 5, pp. 1077–1083, mar, 2015.

[22] J. Maes, M. Guenach, K. Hooghe, and M. Timmers, "Pushing the limits of copper: Paving the road to ftth," *IEEE International Conference on Communications*, pp. 3149–3153, 2012.

[23] D. Bladsjö, M. Hogan, and S. Ruffini, "Synchronization aspects in LTE small cells," *Communications Magazine, IEEE*, vol. 51, no. 9, pp. 70–77, sep, 2013.

[24] J. Ferrant, M. Gilson, S. Jobert, M. Mayer, L. Montini, M. Ouellette, S. Rodrigues, and S. Ruffini, *Synchronous Ethernet and IEEE 1588 in Telecoms: Next Generation Synchronization Networks*, ser. ISTE. Wiley, 2013.

[25] ITU-T, "Definitions and Terminology for Synchronization Networks," International Telecommunication Union, Rec. G. 810, aug 1996.

[26] S. Sesia, I. Toufik, and M. Baker, *LTE - The UMTS Long Term Evolution: From Theory to Practice*, 2nd ed. Wiley, 8 2011. [Online]. Available: http://amazon.com/o/ASIN/0470660252/

[27] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt, and G. Gaderer, "White rabbit: Sub-nanosecond timing distribution over ethernet," in *Precision Clock Synchronization for Measurement, Control and Communication, 2009. ISPCS 2009. International Symposium on*, Oct 2009, pp. 1–5.

[28] B. Razavi, *Design of Integrated Circuits for Optical Communications*. Wiley, 2012.

[29] D. Smith, *Digital Transmission Systems*. Springer US, 2003.

[30] J. L. Ferrant and S. Ruffini, "Evolution of the standards for packet network synchronization," *IEEE Communications Magazine*, vol. 49, no. 2, pp. 132–138, 2011.

[31] ITU-T, "General aspects of digital transmission systems: digital hierarchy bit rates," International Telecommunication Union, Rec. G. 702, nov 1988.

[32] B. Lathi and Z. Ding, *Modern Digital and Analog Communication Systems*, ser. Oxford series in electrical and computer engineering. Oxford University Press, 2010.

[33] J. A. Barnes, A. R. Chi, L. S. Cutler *et al.*, "Characterization of frequency stability," *Instrumentation and Measurement, IEEE Transactions on*, vol. IM-20, no. 2, pp. 105–120, May 1971.

[34] D. Allan, "Time and frequency (time-domain) characterization, estimation, and prediction of precision clocks and oscillators," *Ultrasonics, Ferroelectrics, and Frequency Control, IEEE Transactions on*, vol. 34, no. 6, pp. 647–654, Nov 1987.

[35] A. Srivsatava, *Semidigital Clock-data Recovery System and Bandwidth Extension for Esd-protected High-speed Io Circuits*. Ankit Srivastava.

[36] J. Eidson, *Measurement, Control, and Communication Using IEEE 1588*, ser. Advances in Industrial Control. Springer London, 2006.

[37] E. Gerber and A. Ballato, *Precision frequency control*, ser. Precision Frequency Control. Academic Press, 1985, no. v. 2.

[38] D. R. Stephens, *Phase-Locked Loops for Wireless Communications: Digital, Analog and Optical Implementations*, 2nd ed. Springer, nov 2001.

[39] D. Allan, "Statistics of atomic frequency standards," *Proceedings of the IEEE*, vol. 54, no. 2, pp. 221–230, Feb 1966.

[40] A. Technologies, "Understanding Jitter and Wander Measurements and Standards," Tech. Rep., 03 2003.

[41] J. Rutman, "Characterization of phase and frequency instabilities in precision frequency sources: Fifteen years of progress," *Proceedings of the IEEE*, vol. 66, no. 9, pp. 1048–1075, Sept 1978.

[42] Y. Huang, T. Li, X. Dai, H. Wang, and Y. Yang, "Ts2: a realistic ieee1588 time-synchronization simulator for mobile wireless sensor networks," *Simulation*, vol. 91, no. 2, pp. 164–180, 2015.

[43] R. Cochran, C. Marinescu, and C. Riesch, "Synchronizing the linux system time to a ptp hardware clock," in *Precision Clock Synchronization for Measurement Control and Communication (ISPCS), 2011 International IEEE Symposium on*, Sept 2011, pp. 87–92.

[44] 3GPP TS 36.104, "Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) radio transmission and reception," 2014.

[45] T. Diallo, A. Pizzinat, P. Chanclou, F. Saliou, F. Deletre, and C. Aupetit-Berthelemot, "Jitter impact on mobile fronthaul links," in *Optical Fiber Communications Conference and Exhibition (OFC), 2014*, mar, 2014, pp. 1–3.

[46] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC)protocol specification," 3rd Generation Partnership Project (3GPP), TS 36.321, 06 2010.

[47] D. R. Stauffer, J. T. Mechler, M. A. Sorna *et al.*, *High Speed Serdes Devices and Applications*, 2009th ed. Springer, 10 2008.

[48] J. Frain, "1000BASE-X Physical Coding Sublayer (PCS) and Physical Medium Attachment (PMA) Tutorial," Tech. Rep., Jun. 1998.

[49] Y.-C. Chu, "Serial-gmii specification," *Revision*, vol. 1, p. 3, 2001.

[50] H. Kroener, "Transmission of Ethernet packets via CPRI interface," jul 2009, US Patent App. 11/988,743.

[51] W. Liu, K. Chen, W. Ma, and S. Norberg, "Remote radio data transmission over Ethernet," sep 2014, US Patent 8,842,649.

[52] G. Irvine, "Methods and apparatuses for maintaining synchronization between a radio equipment controller and an item of radio equipment," dec 2013, US Patent 8,599,827.

[53] T. Ryan, "Radio over Ethernet," feb 2014, US Patent App. 13/763,426.

[54] T. Wan and P. Ashwood, "A performance study of CPRI over ethernet," *IEEE 1904.3 Task Force*, 2015.

[55] ITU-T, "Interfaces for the optical transport network," International Telecommunication Union, Rec. G. 709, feb 2012.

[56] ——, "Precision time protocol telecom profile for phase/time synchronization with full timing support from the network," jul, 2014.

[57] Z. Chaloupka, N. Alsindi, and J. Aweya, "Clock skew estimation using Kalman filter and IEEE 1588v2 PTP for telecom networks," *Communications Letters, IEEE*, vol. 19, no. 7, pp. 1181–1184, jul, 2015.

[58] S. M. Kuo, B. H. Lee, and W. Tian, *Real-Time Digital Signal Processing: Fundamentals, Implementations and Applications*, 3rd ed. Wiley, 10 2013.

[59] K. Correll, N. Barendt, and M. Branicky, "Design considerations for software only implementations of the ieee 1588 precision time protocol," in *Conference on IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, 2006.

[60] H. Abubakari and S. Sastry, "Ieee 1588 style synchronization over wireless link," in *Precision Clock Synchronization for Measurement, Control and Communication, 2008. ISPCS 2008. IEEE International Symposium on*, Sept 2008, pp. 127–130.

[61] R. G. Cease, C. J. Favaloro, and J. G. Hackendorf, "Asynchronous data clock generator," jun 1983, US Patent 4,596,026A.

[62] M. Daami and N. D. Georganas, "Client based synchronization control of coded data streams," in *Multimedia Computing and Systems '97. Proceedings., IEEE International Conference on*, Jun 1997, pp. 387–394.

[63] M. Kato, N. Usui, and S. Tasaka, "Stored media synchronization based on buffer occupancy in phs," in *Personal, Indoor and Mobile Radio Communications, 1997. Waves of the Year 2000. PIMRC '97., The 8th IEEE International Symposium on*, vol. 3, Sep 1997, pp. 1049–1053 vol.3.

[64] IEEE Computer Society, "IEEE Std 802.3 - 2008 Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications," 2012, section 2, Clause 31.

[65] ITU-T, "Precision time protocol telecom profile for frequency synchronization," out, 2010.

[66] Xilinx, Inc., "AXI Reference Guide," mar 2011, UG761.

[67] ——, "AXI 1G/2.5G Ethernet Subsystem v7.0," PG 138, sep 2015.

[68] W. Coates and R. Drost, "Congestion and starvation detection in ripple fifos," in *Asynchronous Circuits and Systems, 2003. Proceedings. Ninth International Symposium on*, May 2003, pp. 36–45.

[69] *Si5324: Any-frequency Precision Clock Multiplier/jitter Attenuator*, Silicon Labs, 1 2014, rev. 1.1.

[70] IEEE Computer Society, "802.1AS-2011 - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks," mar 2011, IEEE Standard for Local and metropolitan area networks.

[71] I. Hadzic and D. Morgan, "On packet selection criteria for clock recovery," in *Precision Clock Synchronization for Measurement, Control and Communication, 2009. ISPCS 2009. International Symposium on*, Oct 2009, pp. 1–6.

[72] S. Bjornstad, D. Hjelme, and N. Stol, "A packet-switched hybrid optical network with service guarantees," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 8, pp. –107, aug, 2006.

[73] ITU-T, "Packet delay variation network limits applicable to packet-based methods (frequency synchronization)," feb, 2012.

[74] A. Lometti, G. Cazzaniga, S. Frigerio, and L. Ronchetti, "Synchronization techniques in backhauling networks," in *Transparent Optical Networks (ICTON), 2013 15th International Conference on*, jun, 2013, pp. 1–6.

[75] I. Hadžić and D. Morgan, "Adaptive packet selection for clock recovery," in *Precision Clock Synchronization for Measurement Control and Communication (ISPCS), 2010 International IEEE Symposium on*, Sept 2010, pp. 42–47.

[76] A. Bletsas, "Evaluation of Kalman filtering for network time keeping," *Ultrasonics, Ferroelectrics, and Frequency Control, IEEE Transactions on*, vol. 52, no. 9, pp. 1452–1460, Sept 2005.