Analysis and Evaluation of End-to-End PTP Synchronization for Ethernet-based Fronthaul

Igor Freire*, Ilan Sousa*, Igor Almeida†, Chenguang Lu†, Miguel Berg† and Aldebaro Klautau*
*Signal Processing Laboratory, Federal University of Pará, Brazil
{igorfreire, ilan, aldebaro}@ufpa.br
†Ericsson Research, Kista, Sweden
{igor.almeida, chenguang.lu, miguel.berg}@ericsson.com

***Abstract*—Provisioning of cost-effective Ethernet-based fronthaul by reusing the LAN infrastructure available in most commercial buildings is challenging predominantly in terms of the required bandwidth and synchronization. In contrast to a synchronous fronthaul, a PTP-based Ethernet network must cope with estimation noise introduced by packet delay variation (PDV) for synchronization recovery. The SYNC packet used for PTP on such networks is expected to suffer from significant PDV due to the fronthaul traffic and other background traffic. Further challenge is when the involved network switches do not support PTP and therefore synchronization can only be done by end-devices. Focusing on this scenario, this paper analyzes the problems that may affect the time-offset estimation accuracy and presents schemes to mitigate these problems. The performance is evaluated through a self-developed FPGA-based testbed and the results suggest that the end-to-end PTP approach can fulfill the less strict time alignment requirements of 3GPP standards if PDV is handled properly.**

***Index Terms*—PTP, IEEE 1588, Fronthaul, CPRI, Ethernet.**

I. Introduction

Cloud radio access networks (C-RAN) provide key solutions for efficient allocation and management of baseband processing resources [1], which are essential to forthcoming *ultra-dense* [2] deployments. However, its emergence poses demands for increased flexibility in the *fronthaul*, either in terms of the infrastructure required for new installations or in terms of baseband traffic routing. Thus, Ethernet has been investigated by standardization task forces such as IEEE 1904.3, IEEE 802.1CM and IEEE1914.1, aiming at further evolving the current fronthaul protocols such as CPRI [3] and OBSAI [4] to support Ethernet.

An obstacle to the replacement of synchronous fronthaul networks is the fact that they inherently enable the delivery of synchronization through line timing paths formed at the physical layer of cascaded nodes, while conventional Ethernet deliberately uses free-running clocks and only provide synchronization through ad hoc solutions such as the Synchronous Ethernet (SyncE) and the Precision Time Protocol (PTP). Since fronthaul networks are relatively more recent, a current problem is to ensure the accuracy required by 3GPP [5] is achievable through such solutions.

Particularly for PTP, PDV represents the main limitation to accuracy. For example, [6] concludes that the fronthaul requirements for jitter can't be satisfied unless schemes such as frame preemption, traffic scheduling or de-jitter buffering are used to alleviate the PDV. Such strategies effectively reduce the PDV in the network, but generally require equipment upgrades. In this work, we live with the PDVs caused by network and investigate the PTP estimation process, such as packet selection and filtering [7], [8], to mitigate the PDV effects and improve the synchronization accuracy.

Packet selection has been thoroughly investigated in several use cases for more than a decade. In general, wise use of the technique must take the PTP traffic statistics into account, either through off-line observation or dynamically [9]. Most of the literature considers the selection of packets experiencing minimum or maximum [10] delays, but depending on the network load and the background traffic pattern, other metrics such as sample-mean [7] and sample-mode are applicable. Delay profiles that do not match a filter available in the system can benefit from the sample-mode strategy, as shown in [11].

Filtering algorithms, in turn, are applied on the estimations whose variations are slow relative to the PTP periods, with the assumption that instantaneous variations in the estimations are due to noise. Many filtering schemes have been proposed, in some cases applied to the time offset estimations and in others to the frequency offset. For example, [8] evaluates exponential-smoothing, linear programming and Kalman filtering strategies. Similarly, [12] evaluates Kalman filtering applied to frequency offset estimations.

This work concentrates on practical difficulties related to packet selection and filtering that arise in legacy Ethernet-based fronthaul networks with PTP solely implemented at the endpoints (BBU and RRU). The scenario is illustrative of fronthaul operation over third party networks without synchronization time service, which is being considered for the ITU-T G.8275.2 profile, known as Partial Timing Support. This work presents a combination of solutions that contribute to the the accuracy achieved in this scenario and evaluates them using an FPGA-based hardware and its corresponding software.

This paper is organized as follows: Section II describes the PTP estimations and their corresponding impairments; Section III presents practical difficulties experienced in an endpoint-based PTP scheme; Section IV presents the solutions adopted for the challenges in the previous section; and Section V presents the results obtained through the testbed. Finally, Section VI summarizes the conclusions.

II. System Model

The PTP system considered in this work employs both the peer-delay mechanism for delay estimations and one-way transmissions of the so-called SYNC packet by the *clock master* toward *slaves* for time and frequency offset corrections. Furthermore, the peer-delay mechanism is assumed to be adopted in a non-standard manner, where peers can communicate over hops, with the implicit assumption that intermediate switches do not filter the corresponding packets.

The slave's clock is assumed to present both a time offset $x(t) = T_s(t) - t$, where $T_s(t)$ corresponds to its local time at true time t, and a frequency offset $y(t)$ relative to the master. The master is assumed as an ideal reference clock, whose local time $T_M(t)$ is identical to the true time t. Thus, whenever the slave initiates a peer-delay request and the PTP master acts as a responder, the true (or master) times $(t'_1, \cdots, t'_4)$ corresponding to the timestamps $(t_1, \cdots, t_4)$ are defined as:

$$\begin{cases} t'_1 = t_1 - x(t_1), \\ t'_2 = t_2, \\ t'_3 = t_3, \\ t'_4 = t_4 - x(t_4), \end{cases} \tag{1}$$

where $t_1 = T_s(t'_1)$ marks the departure of the PDELAY_REQ; t_2 its arrival at the link peer (delay responder); t_3 the departure of the $PDELAY\_RESP$; and $t_4 = T_s(t'_4)$ marks the arrival of the response back to the delay requestor. Note that t_2 and t_3 are taken at the master, while t_1 and t_4 are taken at the slave, therefore are corrupted by the time offset.

The slave-to-master delay d_{sm} and the master-to-slave delay d_{ms} are given by:

$$\begin{cases} d_{sm} &= t'_2 - t'_1 = t_2 - t_1 + x(t_1) \\ d_{ms} &= t'_4 - t'_3 = t_4 - t_3 - x(t_4). \end{cases} \tag{2}$$

At this point, in order for the slave to estimate the link delay, two important conditions must be satisfied. The first is that the slave's time-offsets $x(t_1)$ and $x(t_4)$ are approximately equal, which is practically true over the short period. The second is that the forward and backward transit times are equal, which allows the equal one-way delay to be solved from the system of equations by using the timestamps $(t_1, \cdots, t_4)$ collected after the j-th peer-delay mechanism cycle:

$$\hat{\tau}_j = \frac{d_{ms} + d_{sm}}{2} = \frac{(t_{4,j} - t_{1,j}) - (t_{3,j} - t_{2,j})}{2}. \tag{3}$$

However, this is only approximately satisfied with certain probability.

Once delay estimations are available, a separate mechanism (other than the peer-delay) allows the time-offsets to be computed based on the departure t_1 (from master) and arrival t_2 (at the slave) timestamps of the k-th SYNC packets[1]:

$$\hat{x}_k = t_{2,k} - \left(t_{1,k} + \hat{d}_k + \gamma_k \right), \tag{4}$$

where the master time by the time its SYNC message arrives at the slave is obtained by adding the current link delay estimation $\hat{d}_k$ to t_1. Importantly, $\hat{d}_k$ is a filtered version of the estimations obtained in (3), which can come from a sliding window of L estimations $\hat{\tau}_j$. Note also that index k in $\hat{d}_k$ indicates that it is the output of the delay filter when the k-th SYNC is received. Hence, for example, if the peer-delay mechanism rate (injecting new delay estimations into the delay filter) is four times lower than the SYNC rate, $\hat{d}_k$ will be the same for every four consecutive time-offset estimations. Finally, note normally there is a correction γ_k accounting for the residence times within switches, but without PTP support in the network it can be neglected.

Based on the sequence of time error estimations, it is possible to estimate the instantaneous clock frequency offset as the discrete-time derivative of the time error function:

$$\hat{y_k} = \frac{\hat{x}_k - \hat{x}_{k-1}}{T_M(t'_{2,k}) - T_M(t'_{2,k-1})}, \tag{5}$$

where the denominator is the interval from one offset estimation to the other, measured in the reference's (master) timescale[2]. From (4), and according to [13], $T_M(t'_{2,k})$ is the *corrected master event timestamp*, which is a projection of the master time when timestamp $t_{2,k}$ is taken at the slave (i.e. at true time $t'_{2,k}$), defined as $t_{1,k} + \hat{d}_k + \gamma_k$.

Therefore, using (4), the estimation can be re-stated as:

$$\hat{y_k} = \frac{\Delta T_S - \Delta T_M}{\Delta T_M}, \quad \text{where} \tag{6}$$

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

These offsets are, then, filtered by a moving-average filter and used to update the increment value for the RTC that is applicable to provide its syntonization (frequency correction).

Naturally, the problem is that PDV is always present, so that the estimations in (3) are erroneous and, consequently, (4) and (6) too. In this context, one pre-requisite for establishing strategies to accurately detect the time-offsets is to understand the statistics of the delays. The one-way delays d_k are realizations of a biased random variable that can be modeled[3] as:

$$d_k = d_{\text{prop}} + d_{\text{trans}} + d_{\text{process}} + D^k_{\text{queuing}} \tag{7}$$

where d_{prop} is the propagation delay, d_{trans} is the transmission delay, d_{process} is the processing delay and D^k_{queuing} is the k-th realization of the random queuing delay.

Assuming fixed network topologies, equipments and routing paths, transmission and propagation delay can be assumed constant. In contrast, processing delay can be modeled as a Gaussian [14] random variable, but with negligible variance relative to queuing, so it is assumed constant for simplicity in the ensuing derivation. Finally, queuing delay is a random

[1]Note t_1 and t_2 in (4) are timestamps from a mechanism different to the peer-delay mechanism of (3). Note also j and k are indexes of the iterations of two separate processes, which can be configured with different periods.

[2]Again, the t_2 in (5) is the timestamp marking SYNC reception at slave used in (4), but not the same t_2 from (1).

[3]Note the same model is valid for the one-way transit time of SYNCs or peer-delay packets. Thus, both d_k and d_j are used in the text.

variable, with mean μ_q, variance σ_q^2, and distribution that can be modeled as Erlang for *cross-traffic* patterns and mirrored Erlang for *in-line* traffic [9], [11], due to a sum of independent exponentially distributed queuing delays in each hop.

The essence of such considerations is that the delay estimations to be used in offset computations must be decided and learned by the slave. The first question is what delay the system is looking for, the minimum delay, the average, the maximum or any other? The answer must consider the probability that SYNCs transmitted to slaves are subject to a delay close enough to the choice. Then, a corresponding packet selection strategy must be used, as clarified in the sequel.

By using the model in (7), consider the actual time-offset x_k that should be computed by the estimation $\hat{x}_k$ from (4):

$$x_k = t_{2,k} - \left(t_{1,k} + d_{\text{prop}} + d_{\text{trans}} + d_{\text{process}} + D^k_{\text{queuing}}\right), \quad (8)$$

This reveals that the delay choice determines the pattern in the time offset estimation error, which can be stated as:

$$\begin{aligned}\epsilon_x &= \hat{x}_k - x_k \\ &= \left(d_{\text{prop}} + d_{\text{trans}} + d_{\text{process}} + D^k_{\text{queuing}}\right) - \hat{d}_k. \end{aligned} \quad (9)$$

For example, a moving minimum delay estimation can be chosen, in which case the delay filter output is determined by $\hat{d}_j = \min\{\hat{\tau}_j, \cdots, \hat{\tau}_{j-L+1}\}$. Such a filter tends to select the delays of (7) whose realizations of D^j_{queuing} are minimum. Then, further assuming the minimum queuing delay is zero, after sufficient observation a moving-minimum delay estimation should approach $\hat{d}_j = d_{\text{prop}} + d_{\text{trans}} + d_{\text{process}}$, yielding:

$$\epsilon_x = D^k_{\text{queuing}} - \epsilon_{d,\text{min}}, \quad (10)$$

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

Finally, the rationale of packet selection can be stated. When a non-overlapping window of time-offset estimations is accumulated, each individual estimation is subject to a distinct error. Then, the objective is to *select* only the single estimation in the window that is interpreted as the least erroneous. For each window, 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. 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 (10). Then, the time offset selection that minimizes the error (estimation noise) is the minimum estimation within the window. If the minimum queuing delay is effectively zero and the window is sufficiently long to contain such a realization, the error in the selected time-offset approaches $\epsilon_x = -\epsilon_{d,\text{min}}$.

By a similar argument, it can be shown that the fluctuations associated with the windowed time-offset estimations when the mean delay is chosen are given by:

$$\epsilon_x = n^k_q - \epsilon_{d,\text{mean}}, \quad (11)$$

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

$$\epsilon_{d,\text{mean}} = \hat{d}_k - \left(d_{\text{prop}} + d_{\text{trans}} + d_{\text{process}} + \mu_q\right). \quad (12)$$

In this case, since n^k_q is by definition zero-mean, the optimal selection from the window is the mean of all windowed time-offsets, which ideally should leave $\epsilon_x = -\epsilon_{d,\text{mean}}$.

III. Problem Analysis

The transport of CPRI or radio data over Ethernet has to satisfy several requirements specified for radio transmissions. Specifically regarding time alignment error (TAE), several different requirements are defined by 3GPP for different applications. The tightest requirement is for MIMO or Tx diversity transmissions, in which TAE must not exceed 65 ns [15] peak-to-peak (or ± 32.55 ns, the shortest LTE period). The latter is a problem for joint transmissions through distinct RRUs (i.e. different IEEE 1588 slaves), which is the case of coordinated multi-point (CoMP) and eventually of MIMO.

This paper focus on the practical limitations of the algorithms used on top of a standard PTP implementation to achieve these strict RAN requirements, considering PTP is not supported in the network. More specifically, the difficulties presented in this section are inherent to the tradeoffs governing choices for packet selection and filtering algorithms.

A. Packet Selection

The first problem with packet selection is that, while the selection window is being filled, the true time-offset of the RTC continues to accumulate. For example, for a window of 16 samples and a SYNC rate of 128 packets-per-second, if the instantaneous RTC increment leaves a residual frequency offset of $+120$ ppb, during the acquisition of the 16 samples the time-offset increases by 15 ns. Then, even if $\epsilon_{d,\text{min}} = 0$ or $\epsilon_{d,\text{mean}} = 0$ can be guaranteed in (10) or (11), respectively, the offset accumulated within the window is likely to be missed when the "best" estimation is effectively selected.

The second problem is the fact that nothing guarantees one of the packets within the selection window will be subject to the chosen delay. Generally, the two major features to enhance the probability of this event refer to the packet selection strategy and the selection window length. The former can be reasonably chosen statically or dynamically (see [9]) by considering the queuing delay distribution in the particular network. Contrarily, the window length choice is involved with tradeoffs. In essence, a longer window introduces a slower response to instantaneous offset estimations, enhances the first problem (of missing the time-offset accumulated over the window) and leads to more abrupt step corrections. In contrast, a shorter selection window diminishes the probability of acquiring a SYNC subject to the chosen delay and having a perfect cancellation of delays in (9).

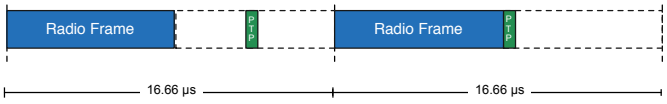

Fig. 1. Examples of PTP message locations within the inter-departure interval of the radio frames for 64 CPRI BFs at line rate option #1.

B. Estimation Filtering

As detailed in Section II, in addition to packet selection, the system considered in this work employs two separate filters, one for the RTC increment value and another for the delay estimations. One problem is that any change in the increment value or in the chosen delay alters the error patterns in time offset estimations. For example, in the case of (10), the error $\epsilon_{d,\min}$ would change if the chosen delay $\hat{d}_k$ was changed or the time offset increase rate would change with an alteration in the RTC increment. The problem is that packet selection requires time offset estimations accumulated in a window to be compared, which implies for fairness the estimation errors of each individual time-offset in the window must be subject to the same conditions. Thus any change in the error patterns while the window is being filled is undesirable.

C. Concurrent Fronthaul Traffic

PDV is mostly a consequence of queuing delays in store-and-forward switches. Therefore, the strongest limitation to the accuracy of PTP being deployed in the fronthaul is the own concurrent radio traffic. For example, if a PTP packet arrives at the switch right after (piggybacking) an radio frame, the PTP message has to wait until the entire radio frame transverses the switch. Furthermore, even if a preemption scheme is adopted, when a PTP packet arrives while an radio frame is already being transmitted, queuing is non-preemptive [7] and, therefore, larger queuing delays are still exhibited.

The problem is also partly from the fact that the Ethernet transmission bit rate is higher than the fronthaul IQ sample bit rate. For example, consider the case in Fig. 1, where 64 CPRI basic frames (BFs) of line rate option #1 (128 bits per BF) carrying 2x2 LTE 5 MHz are encapsulated in each Ethernet frame. Considering the sampling frequency for this bandwidth is 7.68 Mhz, and 2 samples are carried per AxC in each BF, 64 BFs are accumulated over 128 sampling periods. As a result, the amount of data for a single Ethernet frame is accumulated in approximately 16.66 µs. In contrast, it takes approximately 8.4 µs to transmit the frame of 1050 bytes (including the Ethernet header) with a 1000BASE-T transceiver. Thus, a long "idle" interval is left for the PTP packets to be located, and this interval is only reduced by paying the price of shorter frames and the corresponding higher overhead.

D. Over-correction

Finally, one problem with PTP implementations in general is that of over-corrections. Due to PDV noise introduced by the concurrent fronthaul radio traffic, offset estimations may exceed the actual values, which can lead to divergence.

TABLE I
SUMMARY OF DIFFICULTIES IN THE CONSIDERED PTP IMPLEMENTATION.

Problem	Short Description
i	Missed time offset due to long selection window
ii	Probability of SYNC delays matching the chosen delay
iii	Abrupt time-offset corrections with packet selection
iv	Selection unfairness due to delay and RTC increment changes
v	Overcorrection of RTC time offset
vi	PDV due to fronthaul radio traffic

IV. Proposed Solution

Table I summarizes the practical difficulties detailed in the previous section. This section presents approaches to improve the time synchronization by addressing each of these problems.

Problems i and *ii* are contradicting, *ii* requires a long window and *i* is caused by a long window. Also *Problem iii* is enhanced for longer windows. Thus, one proposition is to start the system with a relatively short packet selection window. The rationale is that, during initialization, the error between the SYNC delay and the chosen delay is not the worst problem, but rather the frequency offset. Once the RTC increment value approaches a reasonable value (which occurs more easily, due to its coarse granularity), *Problem i* becomes less critical, so the selection window can be enlarged on-the-fly.

With respect to *Problem iv*, in the beginning of each window, the current minimum, mean or maximum filtered delay is sampled and used within the entire window. This guarantees that at least within the window the estimation error patterns are consistent in terms of delay. This is helpful during the startup phase, but nonetheless high error is expected, because the delay estimation may not be sufficiently trained. Once the system achieves a more stable state, in which time-offset estimations are relatively lower, the proposed strategy is to lock the delay estimations and the RTC increment for as long as the "locked" state is preserved. This provides a more stable solution than sampling the delay at the beginning of each window, because the two elements that alter time-offset error patterns do not change even between different windows.

The system infers the "locked" state based on the trend in time-offset estimations. In each selection window, a difference is computed between the time offset computed using the selection and the time offset previously registered in the RTC hardware (updated after the previous selection). The result is, then, divided by the window length L_w and the quotient is regarded as the time-offset "step" δ_i, computed as:

$$\delta_i = \frac{\hat{x}_i - \hat{x}_{i-1}}{L_w}, \tag{13}$$

where i is the packet selection window index. This "step" could be applied while each subsequent sample is acquired to fill window $i+1$. In the end, L_w corrections of δ_i complete the correction up to $\hat{x}_i$, then a new value δ_{k+1} is computed.

However, before applying the so-called step-by-step time-offset corrections, the system first verifies the magnitude of each δ_i. If the magnitude is under a threshold (e.g. fractional nanoseconds) for a number of iterations, the internal "locked"

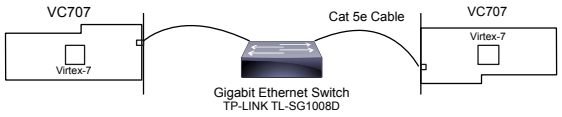

Fig. 2. Testbed setup.

state is entered because the system infers the frequency correction has been finely adjusted. In this case, the step-by-step corrections are enabled to solve *Problem iii*. Otherwise, the system continues to apply abrupt time-offset corrections only after every window becomes full.

Problem v is addressed by attenuating the "steps" δ_i by a coefficient $\alpha < 1$. Once an estimation is obtained from packet selection, L_w corrections of $\alpha\delta_i$ are applied, totalizing $\alpha\delta_i L_w$ instead of the full estimated difference $\hat{x}_i - \hat{x}_{i-1}$ in (13). This approach is similar to [16] and essentially takes longer convergence time as the cost for avoiding over-corrections.

Finally, controlled packet departure is used to address *Problem vi*. The goal is to control the departure of SYNCs relative to the radio packets such that they all experience an almost constant delay. More specifically, a hardware-assisted mechanism was developed to force SYNC departures only right before a radio frame, a simplification of [14] that works for the ensuing evaluated scenario.

V. Results

A testbed was developed using the Xilinx VC707 board and its 7 Series FPGA. A PTP-capable Ethernet MAC with hardware timestamping is instantiated in the FPGA fabric together with an associated RTC fed by a free-running clock of 125 MHz. The RTC counts nanoseconds using Q32.20 fixed-point numbers (20 sub-nanosecond bits) and uses a Q6.20 increment value that provides fine resolution (120 ppb for 125 MHz). The driver periodically updates the RTC time offset registers that are summed with the syntonized (adjusted in terms of increment value) RTC to produce the so-called synchronized RTC (time aligned). Then, the latter is used to derive an 8 kHz clock, whose jitter is evaluated in this section.

Tests individually timed to 5 minutes were carried in the 1-hop setup illustrated in Fig. 2, where one FPGA represents the BBU (PTP master) and the other represents the RRU (PTP slave). A low-cost non PTP-capable switch (TP-LINK TP-SG1008D) is used in the network and fronthaul traffic with 256 bytes of radio data per frame carrying 2x2 LTE 5 MHz goes along the same path as synchronization packets, characterizing in-line background traffic. Measurements were collected in the Keysight Infiniium MSO 9104A oscilloscope at 10 GSample/s. Table II summarizes the adopted default PTP parameters.

Fig. 3 presents the one-way delay profile in the network. It resembles a mirrored-Erlang distribution, as anticipated for in-line background traffic and specially given the link utilization of roughly 50%. Importantly, note the mean delay is 7.278 µs, but the probability distribution is more heavily concentrated near the mode of 7.644 µs. In this case, among minimum, maximum and mean selection strategies, mean is expected to

TABLE II
Default PTP parameters adopted in the tests.

Parameter	Value
Peer-delay Rate	8 packets-per-second
SYNC Rate	128 packets-per-second
RTC Increment Filter Length	128
Delay Filter Length	256
Selection Strategy	Sample-mean
Selection Window Length	16 (initialization) and 64 (locked)
Attenuation Factor	0.5
Controlled SYNC Departure	Always right before a radio frame

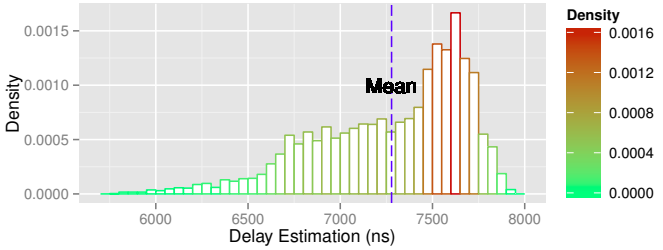

Fig. 3. One-way delay profile considering concurrent in-line fronthaul stream carrying 16 CPRI BFs of 128 bits per frame.

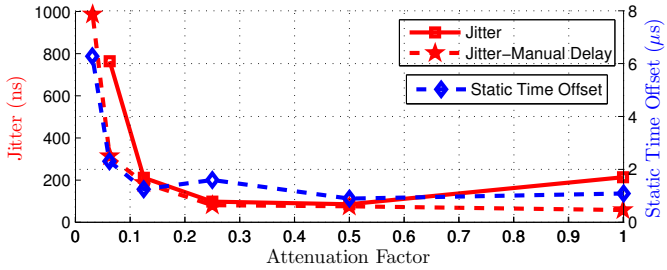

Fig. 4. Jitter and static time offsets for varying attenuation factors.

yield the best performance, but the constant error ϵ_d in (11) will introduce a static timing error in the resulting clock.

A first investigation is with respect to the results in Fig. 4 for varying attenuation factors. Two axis are shown: one for the so-called static time offset, here defined as the constant phase error measured between the center of the jittered clock in the slave and the rising edge (trigger source) of the master clock; and another for the peak-to-peak jitter, here defined as the variation in the slave's rising edge over the measurement period. The latter, in particular, is presented for two cases: when the estimated mean delay is used for the time-offset computations and when a delay is manually inserted through a debug interface such that the static time offset is zeroed. Note the attenuation factor influences both static error and jitter. Also, note that attenuation cannot be too strong. Instead, values close to unit yield better performance. Yet more importantly, note the mean delay estimation leads to inferior performance than the manually inserted delay values close to the mode.

A second investigation concerns the selection window length and its increase when the system enters the "locked" state, which is shown in Fig. 5. In accordance to what

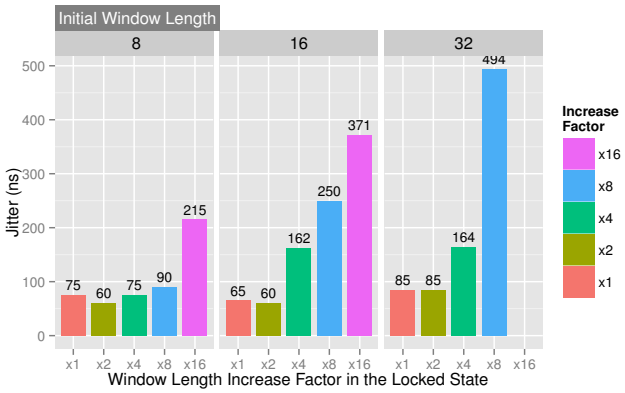

Fig. 5. Combination of selection window lengths and increase factors.

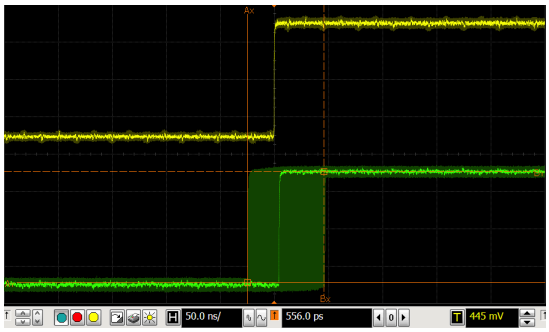

Fig. 6. Time alignment using the optimal parameters over 1 hop.

was discussed regarding *Problems i* and *ii*, short selection windows of 8 or 16 combined with a doubling in window size yielded reasonable compromises between the two conflicting problems. The individual results confirm the importance of starting with a shorter selection window and the gain provided by window enlargement after entering the stable state.

Finally, Fig. 6 presents the infinite persistence plot of the master and slave clock signals using the optimal configuration from the above experiments, which were given in Table II, and a delay that reduces the static error to zero. A jitter of approximately ± 70 ns is observed. Nonetheless, assuming a PLL with sufficiently narrow loop bandwidth can further attenuate the jitter, an average TAE below the most strict 3GPP requirement of 65 ns can be attained.

VI. Conclusions and Future Works

Synchronization is a well-developed area with a large body of algorithms and architectures. In the context of fronthaul transmission, the contribution of this work was to highlight practical difficulties and potential solutions of selection and filtering techniques applied on endpoint-based PTP systems. The error pattern introduced by the delay estimation error was modeled and it was shown that once the most probable delay in the network is effectively searched by the selection strategy, a time error under 3GPP specification can be achieved. Furthermore, it was analyzed and demonstrated that jitter can be constrained by a combination of strategies, such as adopting on-the-fly increase in the packet selection window lengths, attenuating time-offset estimations prior to correction and, primarily, by locking the delay and RTC estimations once the system converges to a stable state.

Future extensions of this work shall investigate the performance over more practical network topologies, including *cross-traffic* scenarios that were not considered in this work; model-based filtering approaches such as Kalman filtering; improved delay search strategies; further jitter attenuation through extra PLL layers; and effects due to improved sub-nanoseconds resolution in the RTC increment.

Acknowledgment

This work was supported in part by the Innovation Center, Ericsson Telecomunicações S.A., Brazil, the Capes Foundation, Brazil, and by the European Union through the 5G-Crosshaul project (H2020-ICT-2014/671598).

References

[1] C.-L. I, J. Huang, R. Duan *et al.*, "Recent progress on C-RAN centralization and cloudification," *Access, IEEE*, vol. 2, pp. 1030–1039, 2014.
[2] J. G. Andrews, S. Buzzi, W. Choi *et al.*, "What will 5g be?" *Arxiv preprint*, pp. 1–17, 2014.
[3] "Common Public Radio Interface (CPRI) specification v6.1," http://www.cpri.info, Jul. 2014.
[4] "Open base station architecture initiative (OBSAI) specification v2.0," http://www.obsai.com, Apr. 2006.
[5] D. Bladsjo, M. Hogan, and S. Ruffini, "Synchronization aspects in LTE small cells," *Communications Magazine, IEEE*, vol. 51, no. 9, pp. 70–77, Sep. 2013.
[6] T. Wan and P. Ashwood, "A performance study of CPRI over ethernet," *IEEE 1904.3 Task Force*, 2015.
[7] 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.
[8] 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, Sep. 2005.
[9] 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*, Sep. 2010, pp. 42–47.
[10] D. T. Bui, A. Dupas, and M. L. Pallec, "Packet delay variation management for a better IEEE1588V2 performance," in *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, Oct. 2009, pp. 1–6.
[11] M. Anyaegbu, C. X. Wang, and W. Berrie, "A sample-mode packet delay variation filter for IEEE 1588 synchronization," in *ITS Telecommunications (ITST), 2012 12th International Conference on*, Nov. 2012, pp. 1–6.
[12] 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.
[13] I. Instrumentation and M. Society, "IEEE 1588-2008: Standard for a precision clock synchronization protocol for networked measurement and control systems," Jul. 2008.
[14] B. Mochizuki and I. Hadzic, "Improving IEEE 1588v2 clock performance through controlled packet departures," *IEEE Communications Letters*, vol. 14, no. 5, pp. 459–461, May 2010.
[15] 3GPP TS 36.104, "Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) radio transmission and reception," 2014. [Online]. Available: http://www.3gpp.org/dynareport/36104.htm
[16] 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.