Asymmetry Mitigation through Line Swapping in IEEE 802.3 Ethernet

Reinhard Exel[1], Thomas Bigler[1], and Nikolaus Kerö [2]

[1]Austrian Academy of Sciences, IISS, Viktor Kaplan Straße 2, 2700 Wiener Neustadt, Austria

[2]Oregano Systems, Franzosengraben 8, 1030 Vienna, Austria

reinhard.exel@oeaw.ac.at, thomas.bigler@oeaw.ac.at, nikolaus.keroe@oregano.at

Abstract—One of the basic services in a distributed network is clock synchronization as it enables a palette of services, such as synchronized measurements, coordinated actions, or time-based access to a shared communication medium. The IEEE 1588 standard defines the Precision Time Protocol (PTP) and provides a framework to synchronize multiple slave clocks to a master by means of synchronization event messages. While PTP is capable for synchronization accuracies below 1 ns, practical synchronization approaches are hitting a new barrier due to asymmetric line delays. Although compensation fields for the asymmetry are present in PTP version 2008, no specific measures to estimate the asymmetry are defined in the standard.

In this paper we present a solution to estimate the line asymmetry in 100Base-TX networks based on line swapping. This approach seems appealing for existing installations as most Ethernet PHYs have the line swapping feature built in, and it only delays the network startup, but does not alter the operation of the network. We show by an FPGA-based prototype system that our approach is able to improve the synchronization offset from more than 10 ns down to below 200 ps.

Key words: clock synchronization, IEEE 1588, Precision Time Protocol, asymmetry mitigation

I. INTRODUCTION

Having a common notion of time between the nodes of a network is vital, as many applications depend on tight clock synchronization. Collaborative tasks such as distributed measurements of a physical quantity at different locations, scheduling of algorithms, the timely-ordered detection of events, or even time-based access to a shared communication medium rely on clock synchronization as a basic service of the network. As a result, a significant amount of research has been carried out in the field of clock synchronization aiming at various improvements in terms of precision, accuracy, reliability, and dependability of synchronization. Apart from these enhancements in terms of synchronization quality, other equally important aspects such as security, message overhead, and – connected to it – the energy demand for synchronization in wireless sensor networks have been addressed in the past. Nevertheless, many aspects such as the protection of time by encryption and fault-tolerant synchronization are still open topics.

A. New accuracy demands for clock synchronization

One of the enabling factors for clock synchronization in packet-based networks was the introduction of IEEE 802.3 Ethernet, which became a standard in the mid-1980s. A few years later, the Network Time Protocol (NTP) [1] was among the first synchronization protocols widely adopted. In NTP, multiple clients synchronize their clocks to an NTP server by bilateral message exchange. NTP servers are hierarchically organized in the form of strata generating a highly-scalable solution. NTP clients, at the other end, use statistical measures, such as filtering and outlier detection, to achieve synchronization accuracies in the low millisecond range.

The introduction of the IEEE 1588 standard in 2002 can be considered as a landmark for accurate clock synchronization as it enabled sub-microsecond synchronization in local area networks. The IEEE 1588 standard defines the Precision Time Protocol (PTP), which is used to synchronize devices in packet-based

networks. PTP is based on timestamps, which are taken at the ingress and egress of event messages exchanged between one master and multiple slaves. These timestamps and the assumption of symmetric propagation delays in both communication directions enable the slaves to steer their clocks and follow the master's clock, thus synchronizing all devices. In the updated version of IEEE 1588, version 2008, the constraint of symmetric propagation delay was removed and asymmetry fields and compensation calculations were introduced into PTP. However, no specific procedures to measure the asymmetry were defined within the standard.

Similar to the evolution of the transmission speed of Ethernet, the accuracy of clock synchronization improved over time from about 1 ms to below 1 ns. The accuracy enhancement of about 6 orders of magnitude within the last 25 years has enabled new services or replaced legacy timing networks. It can be accounted to algorithmic improvements, such as improved oscillator characterization, improved clock servo design and filtering, and to a large amount to the quality of the timestamps. Prior to IEEE 1588, timestamps were taken by the synchronization application before sending and after receiving a synchronization packet. Compared to the true ingress or egress time at the network interface, taking the timestamps within the application adds a significant amount of variable delays due to the unpredictable processing time of the network stack, the scheduling of applications within the operating system, interrupt latencies and so on. To mitigate the delay variability, PTP introduced hardware timestamps, which are taken by dedicated timestamping logic at the network interface, thus removing all the software-induced jitter. While IEEE 1588 synchronization using software timestamps is bound to about 10 µs accuracy [3], commercial IEEE 1588 implementations using hardware timestamps can reach accuracies in the 10 ns range [4]. This kind of synchronization accuracy enables the replacement of traditional synchronization cabling. For instance, for test and measurement applications, the LAN eXtensions for Instrumentation (LXI) standard [2] based on PTP was introduced, which guarantees clock offsets below 40 ns.

Nevertheless, there exist a number of applications, where PTP is in practice not accurate enough. One such application is time-based wireless locating in indoor environments. Most state-of-the-art indoor locating systems are based on the received signal strength (RSS), as signal strength measurement is implemented in virtually all wireless chipsets. Time-based locating is based on the assessment of the propagation time between a transmitter and receiver. A particular advantage compared to RSS is that time-based methods can be used over very large distances. Thus, this method is used for satellite ranging systems, such as the Global Positioning System (GPS). The concept of GPS can be used for indoor locating as suggested in [5]. In this paper, multiple fixed base stations capture the signal of a WLAN device and estimate the position based on the Time Difference of Arrival (TDoA) method. The TDoA method does not require synchronizing the WLAN target to be located, but it requires synchronization of the base stations to calculate the TDoA based on a common notion of time. As one nanosecond synchronization offset equates to a ranging error of 0.3 m (and possibly a location error of more than 1 m depending on the current geometry), the synchronization accuracy should be well below 1 ns.

With the completion of the Large Hadron Collider (LHC) in CERN a renovation of the legacy timing network was initiated. The aim of the White Rabbit project was to build a data and timing network based on PTP, which should be able to achieve sub-100 ps phase jitter and sub-nanosecond accuracy enabling distributed measurements [6]. First measurements of the system show that White Rabbit is able to fulfill these ambitious goals with a mean offset of 517 ps and a clock standard deviation of 117 ps [7]. However, the synchronization quality does not come for free. It demands special hardware (switches and nodes), White Rabbit extensions to PTP, and the use of single-mode fiber communication according to 1000Base-BX10. The 1000Base-BX10 standard has been selected as it uses only one fiber in both directions with two different wavelengths resulting in a nearly symmetric link delay. In addition, White Rabbit compensates for the dispersion-induced asymmetry.

It is a fundamental limitation that the asymmetry cannot be measured by round-trip measurements as performed by PTP [8]. The determination of asymmetric propagation paths demands the use of the same physical wire in both directions using the same signal in both directions. Unfortunately this is not foreseen

in most communication media as unidirectional links are used. For White Rabbit the bidirectional 1000Base-BX10 standard was selected, yet due to the dispersive nature of the cable, a small residual offset still remains after the compensation. White Rabbit has been proven to achieve sub-nanosecond accuracy; however the accuracy demands a fiber-only network with fiber optic switches and network cards.

In this paper we investigate the measurement and asymmetry compensation for the 100Base-TX Ethernet standard, which is widely used and allows for cheap copper cabling. Our approach is based on line swapping, where the transmission line pairs are switched within the PHY IC on software-request. As the swapping enables measurements of the same physical line pair in both directions, the asymmetry can be resolved. This approach seems appealing for existing installations as most Ethernet PHYs have the line swapping feature built in. The asymmetry measurement, when performed at link establishment, only delays the network startup but does not alter the operation of the network.

B. Organization of the paper

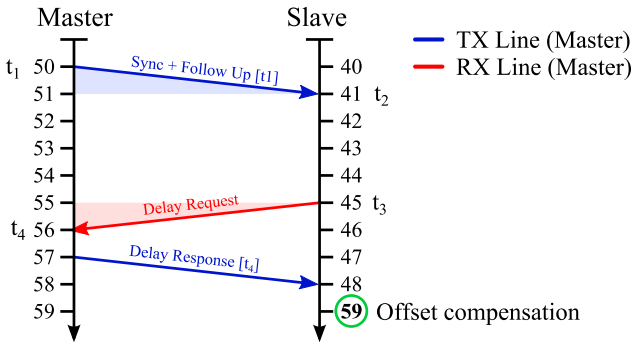
The paper is organized as follows. Section II describes the generic synchronization model and measures dealing with the asymmetry issue. In Section III we describe our proposed line swapping approach to measure the asymmetry. The following section covers the implementation of our approach and necessary prerequisites. Finally, results for the setup are shown in Section V and an outlook is given.

II. MOTIVATION AND RELATED WORK

A. Clock synchronization in a nutshell

Clock synchronization between two entities, commonly a master and a slave, can be classified into one-way and two-way synchronization. In one-way synchronization, the master transmits its current time to the slave and the slave compares the arrival time of the synchronization message to its local clock and adjusts it. The link delay between master and slave is not taken into consideration.

Two-way synchronization also measures the round-trip time and compensates for the link delays. This approach is taken by the IEEE 1588 standard as depicted in Figure 1. First, the master sends a Sync message to the slave. As the master is not aware of the actual egress time of the frame while assembling it, it uses a Follow-Up message containing the precise egress time t_1. For the sake of simplicity, the Sync and Follow-Up messages are combined in one message, because it is only essential that the slave receives t_1. Masters, which are able to transmit t_1 already within the Sync message by inserting t_1 into the message while sending it (so-called one-step mode), do not require Follow-Up messages. The slave timestamps the Sync message and generates timestamp t_2. A similar procedure is performed in the opposite direction. The slave sends a Delay-Request message at t_3, which the master timestamps at t_4 and re-transmits this information back using a Delay-Response message. As soon as the slave knows all four timestamps, it calculates offset and delay (see equations (1)) and adjusts the clock offset as indicated by the green circle at 59 in Figure 1.



$$Offset = \frac{(t_2 - t_1) - (t_4 - t_3)}{2}$$

$$Delay = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \quad (1)$$

Figure 1. Principle of two-way synchronization.

According to the equations (1) it can be seen that this procedure assumes symmetric delays on the transmit and receive lines. Yet, the assumption of symmetric propagation delays is in many cases not fulfilled and neglecting the presence of asymmetry leads to a residual offset between master and slaves.

Figure 2 outlines the situation with asymmetric link delays. In this specific example, the receive line delay is two time units larger than the transmit line delay. Applying the same procedure as described above leads to an offset of one time unit although the slave believes it is perfectly synchronized, as depicted with the green circle at time 60 in Figure 2. The reason for this can be seen on the right part of Figure 2, where the real delays and the calculated delays (dotted grey lines) are compared. It is clearly visible that setting the one-way delay to half the round trip time (RTT) leads to an asymmetry error – in this specific example one time unit. Although the IEEE 1588 version 2008 defines fields for asymmetry correction, it does not specify a particular method for measuring the asymmetry.

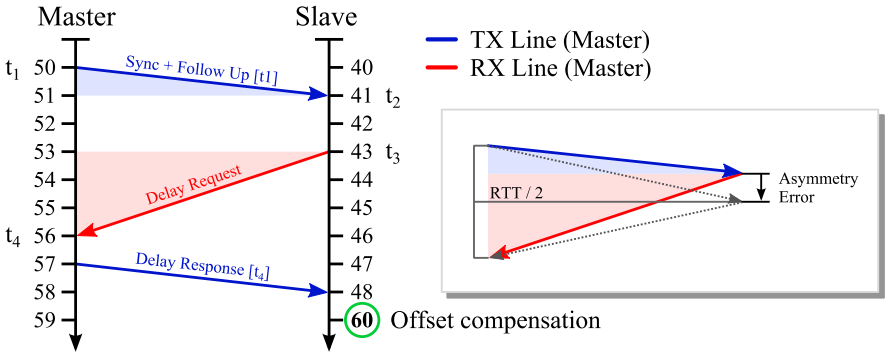


Figure 2. Two-way synchronization with asymmetry.

B. *Sources of asymmetry*

In order to mitigate the impact on the synchronization successfully, it is essential to analyze the reasons for asymmetry. One source is the channel induced asymmetry which depends on the transmission media. For 100Base-TX Ethernet, for instance, a twisted pair cable is used, which may have different twist rates of each cable pair. This obviously alters the length and consequently induces a cable pair dependent delay. The delay variation between different pairs is specified in the ANSI/TIA/EIA 568-B standard to be in a range of up to 50 ns. Even channels which seem to be symmetric at a first glance can show a certain asymmetry. For example, wireless communication is symmetric as long as the same channel is used for transmitting and receiving. Selecting different transmission frequencies also changes the propagation characteristics and consequently induces asymmetry. This wavelength dependent behavior is also present in optical fiber communication where the dispersion causes different transmission latencies.

Focusing on wired Ethernet 802.3 communication, several additional sources for asymmetry can be found. First, the Ethernet PHY itself may generate the so-called lock-in issue. The received serial data stream is converted from 125 MBaud to parallel 5-bit data with a 25 MHz data rate. The subsequent 5B4B decoder has five possible positions to lock onto the data which leads to processing delays of up to 32 ns. Although this delay stays constant after the link between the nodes has been established, it is not possible to know it in advance. Another quite similar source for asymmetry is the digital processing delay inside the PHY which differs for the transmit and receive sides. This can be explained easily, as the receive path demands more complex signal processing than the transmit path, e.g. clock and data recovery or equalization. As the actual implementation (and consequently the number of clock cycles for data processing) may be device and vendor dependent, using two different PHYs for master and slave may create a significant asymmetry. Additionally, constant delays caused by the digital domain and also the analog front end may induce asymmetry as the amplification may change the phase of the signal.

C. State-of-the-art solutions for asymmetry compensation

As shown in Section II.A, it is not possible to eliminate asymmetry within the regular PTP protocol as long as unidirectional communication is used. A real-world example for the delay in large systems is the backhaul network of China Mobile where the clock offset error is up to 6 µs [12].

An algorithmic approach in [13] suggests solving the asymmetry by additional messages. In this particular method the master is assumed to (indirectly) know the asymmetry and to adjust its inter-packet gap accordingly. Consequently, this approach is not feasible as a general solution for any given topology and size. One possible solution is to apply an observer that compares common events (e.g. PPS pulses) among the considered nodes. As a prerequisite for this method the observer itself has to have unbiased monitoring capabilities, which actually claims perfect synchronization. A practical implementation could contain an oscilloscope with two probes connected to the PPS output of the nodes under test. Although this is a feasible measurement setup, it cannot be applied to larger networks easily.

Another method is manual calibration of all cables used in the network. For that, either a delay measuring tool has to be mounted on both ends of the cable or reflectometric metering has to be applied. The latter variant either requires an open or shorted termination. Besides the obvious fact that this method does not work during regular mode of operation, it is rather error-prone as small changes in the network topology may induce delay variations, which remain undetected.

III. ASYMMETRY COMPENSATION IN ETHERNET

In the following section we show how the asymmetry can be measured and mitigated in a 100Base-TX peer-to-peer communication network. To this end, the line swapping approach is introduced which uses a common functionality of standard Ethernet PHYs.

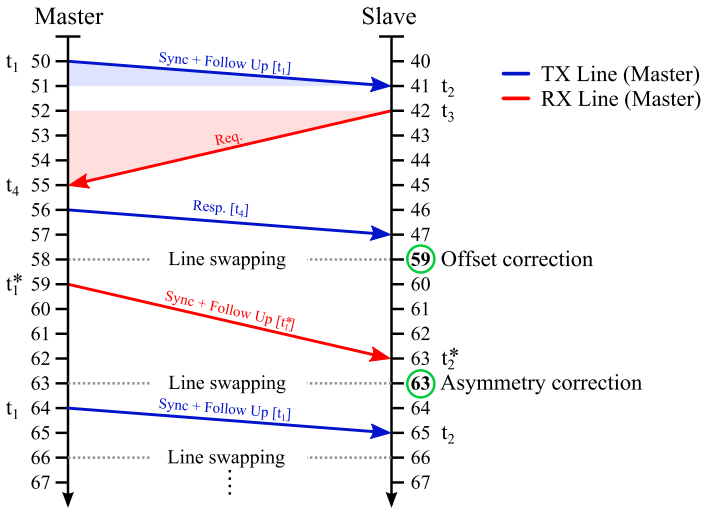
A. The line-swapping approach for 100Base-TX

For a 100Base-TX network either full-duplex or half-duplex modes are applicable for line swapping. Theoretically, the half-duplex mode uses only one line for communication between two nodes. Due to the fact, that one single line has the same delay in both directions, the delay difference can be easily measured when exchanging the transmission line pair.

On the other hand, the full duplex communication uses one transmission line pair and the other one for reception. When using round trip measurements the protocol is able to calculate the offset and delay values for the current configuration. Exchanging the line pairs requires the proposed asymmetry measurement protocol to map specific messages (e.g. Sync, Delay-Request, ...) and timestamps to the associated line in order to calculate the asymmetry. Both methods (half and the full-duplex) have the drawback of line swapping interrupting data communication. As autonegotiation during link establishment may take up to about two seconds, line swapping is only applicable as an extended link start-up procedure.

B. Asymmetry Measurement Protocol

Figure 3 shows the principle for the asymmetry measurement protocol. In a first step, the system performs the normal synchronization procedure where the master sends Sync and Follow-Up messages, followed by Delay-Request and Response messages by slave and master, respectively. Using the collected timestamps t_1 to t_4 the slave calculates offset and delay (see equations (1)) and sets its clock accordingly. As shown in Section II.A, the offset for the slave is reduced to zero although there is obviously an offset of one time unit.



$$Offset = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} = -11$$

$$Delay = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} = 2$$

$$Error = t_2^* - t_1^* - Delay = 2$$

$$Asym. = \frac{Error}{2} = 1 \qquad (2)$$

Figure 3. Asymmetry measurement protocol.

In the next step, the master forces a line swap by altering the PHY's registers via the serial Management Data Input/Output (MDIO) interface. As soon as the link is reestablished the master sends a Sync message to the slave. The slave now calculates the difference between ingress and egress timestamps which should result in the already known delay value for a symmetric system. If there is a delay difference, which is the case in this specific example, this symmetric line assumption does not hold any longer and the measured value represents double the asymmetry value. Equations (2) show the calculation of offset, delay and asymmetry correction values.

The asymmetry correction value has to be either added or subtracted depending on the line swapping state with respect to the current delay value. In the example shown above, the receive delay is increased by the asymmetry correction value of one time unit, whereas the transmit line delay is decreased by one time unit. This leads to precisely the delays shown in Figure 3 – the system is now synchronized and the asymmetry is compensated. For the ideal system it would be sufficient to just do one line swap after the system has been synchronized. In reality, several line exchanges have to be executed to improve the accuracy of the asymmetry estimate.

IV. IMPLEMENTATION

As the cable-induced asymmetry is typically in the range of a few nanoseconds, the measurement of time spans in the nanosecond range demands a carefully designed hardware implementation and the selection of a physical layer IC, which is able to support these accuracy demands. In the following we describe the key points for a system, which is able to measure asymmetry and finally synchronize with accuracies far below the nanosecond range.

A. Prerequisites and limitations

Apparently, several prerequisites and limitations have to be considered when designing a system that should be able to measure delay errors in the range of sub-nanoseconds. As outlined in Section II.B, a commercial off-the-shelf PHY may have different lock-in positions on every link establishment. Consequently, it is crucial to use a PHY that does not show this shortcomings – this is fulfilled for the Texas Instruments DP83848 which was used in our hardware platform. Regarding the PHY, the support for the auto-cross (auto MDI-X) feature is mandatory as well as the slave has to follow the line swap forced by the master. Another important point for the implementation is a highly accurate timestamping unit that connects to the Media Independent Interface (MII) interface between PHY and MAC. Basically, the timestamp resolution is limited by the clock frequency which in our case is set to 102 MHz and leads to a resolution of 9.8 ns. Due to some mathematical optimizations, we were able to improve this resolution down to 76.6 ps, cf. Section IV.C. Finally, the oscillator stability and skew are crucial points as the system has to be syntonized to perform the asymmetry measurements. When using oscillators with insufficient stability or without compensating for the clock skews, the notion of time would drift apart from master to slave degrading the asymmetry measurements.

B. Hardware and software architecture

For the hardware platform a custom made FPGA board was selected. This so-called SMiLE3 board has been used previously for wireless synchronization applications [14] and also fulfills the requirements for synchronization measurements mentioned above. One important feature is the possibility to interconnect two (or even more) SMiLE3 boards in order to share a common clock between the boards. This way, a potential oscillator drift has equal impact on both boards and a clock control servo can therefore be omitted. This enables us to focus on the asymmetry measurement without the requirement to deal with oscillator stability issues.

Figure 4 illustrates the most relevant parts of the hardware implementation. The FPGA System-on-Chip contains an Altera NIOS II CPU and an Opencores 10/100 Mbps MAC both running at 100 MHz clock frequency. The MAC is connected to the Texas Instrument Ethernet PHY DP83848 via MII. The MII is also connected to the ClockCore module to timestamp egress and ingress Ethernet frames. The timestamping units (TX/RX TS) are able to create a timestamp for every nibble on the MII rather than operating in single-shot mode at one specific position within the Ethernet frame. This method is further explained in Section IV.C, where it is also outlined why the Adder-based Clock (ABC) within the ClockCore needs to run at a slightly higher clock frequency than an integer multiple of the 25 MHz MII clock. Because of this, FIFO (TX/RX FIFO) buffers are used both for stable clock domain crossings and to store several timestamps in case the CPU is unable to fetch the timestamps in time. The ABC provides the time used by the timestampers and is also directly connected to the PPS generator (PPS Gen.).

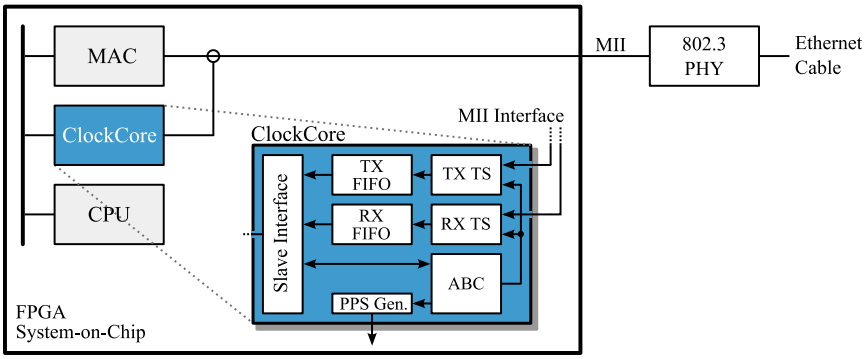


Figure 4. FPGA System-on-Chip implementation.

From the software side an Altera NIOS II CPU is used as it allows for convenient programming and debugging on the development platform. The current implementation works on an Ethernet frame basis, which makes it possible to run the system without TCP/IP stack and also without an operating system.

C. Timestamp and clock optimization

Key for the assessment of delays in the nanosecond range and even below is the quality of the timestamps taken at ingress and egress of synchronization messages. In fact, the increase of the synchronization accuracy within the last few years can be accounted to improved timestamping approaches and the use of highly stable oscillators or networks, which distribute a common carrier signals, such as SyncE [9]. Timestamping methods can be classified into single-shot methods and phase-estimating methods [10]. Single-shot takes a single timestamp per frame, when a specific position of the frame has passed the timestamping unit. This approach is intuitive, when considering that IEEE 1588 version 2008 defines in Section 7.3.4.2 that timestamps need to be taken, when the event message timestamp point passes the reference plane marking the boundary of the PTP node and the network [15].

The 100Base-TX standard uses unidirectional transmission lines with one line pair in each direction. The transmitter modulates the data on the line, the receiving physical layer device on the opposite side recovers the data and embedded clock signal and outputs the data on the MII. Thus, the receive clock is phase locked to the respective transmitting PHY, i.e., its clock source. When taking timestamps with the local clock, the phase relationship between the local clock and receive clock is arbitrary. Thus, the timestamping induces a uniformly distributed jitter with the width of the local clock period. Consequently, the local clock period should be as short as possible, or equivalently, the clock frequency should be very high. Even in recent FPGA hardware, adder-based clock designs are limited to a few hundred MHz, thus resulting in timestamping errors of 1–20 ns.

Nevertheless, it is not prohibited to use any other timestamp point, when the appropriate latencies to the reference point are taken into account. This fact provides the basis for the second class of timestamping methods. Phase-estimating methods determine the event message arrival time by estimation, and the timestamp is not simply a copy of the time register. This approach has several advantages in terms of accuracy compared to single-shot methods, because frame timestamps can be generated by taking multiple timestamps per frame and averaging can be used to reduce the timestamping error. It is well-known from estimation theory that averaging multiple realizations of a random function may lead to improved statistics, e.g., the mean value. Yet, high-quality timestamps can only be calculated if the timestamps realizations have a high statistical independence.

Let us illustrate this with an example. Consider that the MII receive clock is 25 MHz with a specified frequency skew of 50 ppm. A timestamper with 100 MHz will lead to a uniformly distributed error of 10 ns. When taking multiple timestamps per frame, the 100 MHz clock samples the 25 MHz clock at the same phase, as 100 is an integer multiple of 25. In reality, due to clock skews we observe the beating frequency between these clocks, and the captured timestamps are highly correlated. Thus, it is apparent that the local clock frequency needs to have a frequency offset to increase the beating frequency and improve the statistical independence of the timestamps taken within each frame. For this reason, our implementation uses a 102 MHz local clock and a linear regression model for averaging. Let $\boldsymbol{y}$ be the true timestamp vector, $\boldsymbol{x}$ the measured timestamp vector, a the clock offset, b the clock skew, and $\boldsymbol{e}$ the residual error vector and the system follows the simple linear regression model as

$$\mathbf{y} = a + b\mathbf{x} + e \tag{3}$$

Then we can estimate the clock offset a and skew b by (with $E[]$ the expectation operator)

$$b = \frac{\mathrm{cov}(\mathbf{x}, \mathbf{y})}{\mathrm{var}(\mathbf{x}, \mathbf{y})} \text{ and } a = E[\mathbf{y}] - bE[\mathbf{x}]. \tag{4}$$

Graphically, this approach performs a least-squares line fitting for the noisy measurement data ***x***. With the knowledge of *a* and *b* it is possible to calculate a timestamp for any arbitrary position within the frame. A particularly appealing approach for the hardware implementation is to set the timestamp point at the center of the frame, because the regression model degrades and the timestamp estimate is simply $E[\boldsymbol{x}]$. This approach improves the timestamp precision from 9.8 ns to about 400 ps for a frame with 600 timestamps per frame. This matches with the assumption of independent timestamps as the statistics improve by sqrt(600) = 24.5. Internally, we use the same resolution as for the ABC, 76.6 ps.

Similar to the quantization of the single-shot timestamps is the output quantization of the system. When using Adder-based clocks (ABCs) for synchronization applications, the local clock frequency is fixed and the slave clocks follow the master clock by adjusting the clock increment, which represents the amount of time added per clock cycle. To compare the synchronization quality of two synchronized devices a Pulse Per Second (PPS) signal is generated, whenever a certain time span elapses, commonly one second. When using ABCs this pulse is generated at the clock edge of the local clock when the full second overflows. Thus, when using a 102 MHz local clock, the PPS is quantized to 9.8 ns. One solution to this measurement issue is to delay the PPS pulse with a variable delay circuit depending on the fractional error after the overflow (from 0 to 9.8 ns). We selected an even simpler approach by modulating the fractional error as additive pulse width to the PPS. For each unit of the fractional PPS error (76.6 ps), the pulse width is lengthened by one clock tick (9.8 ns), resulting in a zoom factor of 128. Thus, the measured clock offset between two devices is the difference of the PPS arrival times and the difference of the pulse widths of master and slaves by

$$Offset = (t_M - t_S) + (PW_M - PW_S)/128. \tag{5}$$

Figure 5 depicts this principle. The advantage of this approach is that it permits measurement of the clock offset with very high accuracy without requiring extremely high local clock frequencies or additional analog phase shift circuits.

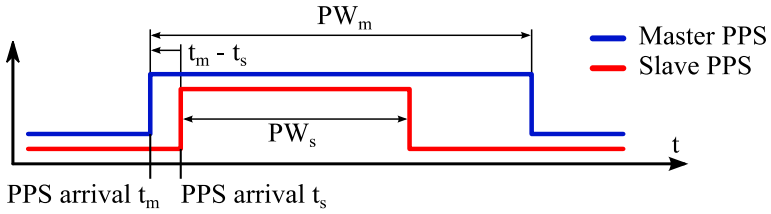


Figure 5. Pulse width modulated offset output.

V. PERFORMANCE ASSESSMENT

In order to assess the performance of the line swapping method and to get comparable results, this section proposes a special measurement setup which allows the creation of artificial asymmetry. Finally, several results are compared and it is shown that the system works equally well for small and large asymmetries.

A. Measurement Setup

The measurement system is a special setup that allows the connection of two separate cables, in order to use one for the transmit path and the other for the receive path (as seen from the master). Figure 6 shows that the blue cable/wire is much longer than the red one, introducing an artificial delay and therefore asymmetry.

This adapter is connected to the SMiLE3 hardware platform that is syntonized via an additional cable. This way, the ClockCore modules in both nodes run at the same frequency and the otherwise necessary clock servo can be omitted. The decision to use direct board-to-board syntonization instead of free-running oscillators was taken to keep the work focused on the asymmetry mitigation. Otherwise, the oscillator quality and servo control algorithm would influence the results and outcome of the measurements. Finally,

the PPS pulse output of both nodes is connected to an oscilloscope where the low to high transition of the PPS signal is compared. The time difference then displays the residual offset between the two nodes.

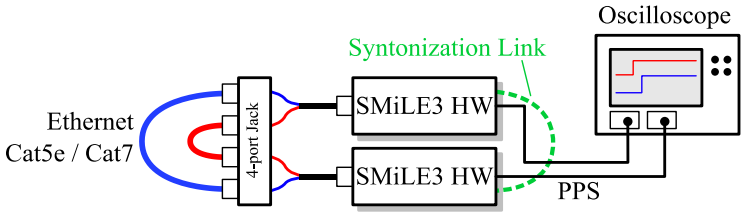


Figure 6. Measurement setup.

B. *Performance with syntonized nodes*

Before using the measurement setup illustrated in Figure 6, all of the cables were directly connected to the SMiLE3 hardware to measure the asymmetry of each single cable. Table 1 shows the results of these measurements, whereas the entries listed in the table correspond to the asymmetry values as seen inside the software implementation. One time unit equals the LSB of the ClockCore, which is 76.6 ps. Multiplying the units by this resolution gives the asymmetry of the cable. As it can be seen in table 1, for the 1 m cable, asymmetries below 76.6 ps cannot be measured, resulting in an asymmetry value of zero.

Table 1. Asymmetries of standard CAT5 and CAT7 cables.

Cable Type	Length	Asym. [Units]	Asym.
Cat5E	1 m	0	0.0 ps
Cat5E	10 m	3	229.7 ps
Cat7	20 m	8	612.7 ps
Cat5E	60 m	26	1991.6 ps

From the measurement point of view there exist two possibilities to show the asymmetry value for a specific cable. In the first approach, line swapping without asymmetry compensation is performed, and one can observe the delay asymmetry directly by the PPS pulse. An example for such a measurement is shown in Figure 7 for the 60 m cable. First, the slave synchronizes to the master until it believes that the synchronization offset is zero. Afterwards, the lines are swapped which results in a jump from -2 ns to approximately +2 ns. By dividing this value by two the approximated asymmetry of 2 ns can be calculated for this specific cable (compare with Table 1, last entry).

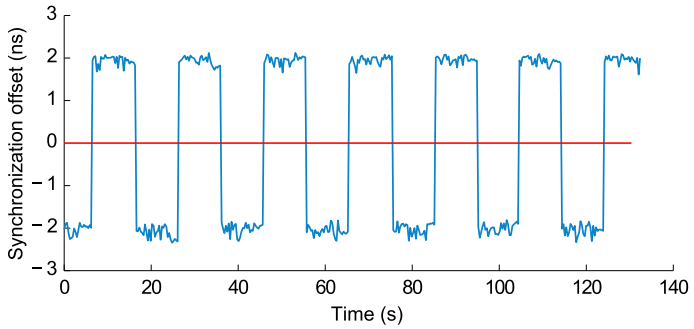


Figure 7. Measurement 60 m cable with asymmetry compensation off.

This second approach basically corresponds to the observer approach described in Section II.A. Enabling the asymmetry compensation removes these jumps and exhibits the residual asymmetry on the oscilloscope. Figures 8 and 9 depict a measurement with a combination of a 50 cm and 1 m cable (see the measurement setup in Figure 6). Consequently, as the difference in length is 50 cm this creates a delay difference of about 2.5 ns (assuming a propagation speed of 2/3 the speed of light). The measurement in

Figure 8 shows the result with asymmetry compensation turned off, whereas it is enabled in Figure 9, depicting the residual asymmetry. It can be seen that the resulting error is inside a boundary of ±200 ps with a mean value of -76 ps. Also visible is that the asymmetry compensation conforms to the expected delay of 2.5 ns.

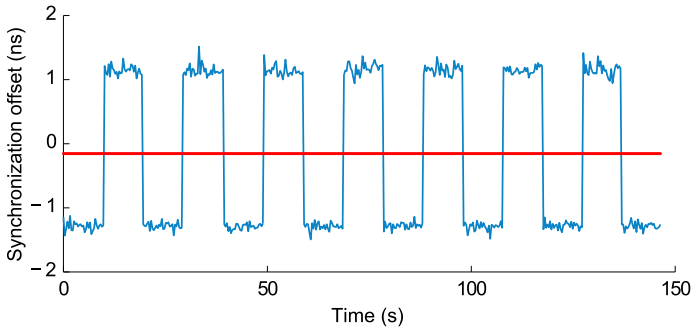


Figure 8. Measurement 50 cm / 1 m cables with asymmetry compensation off.

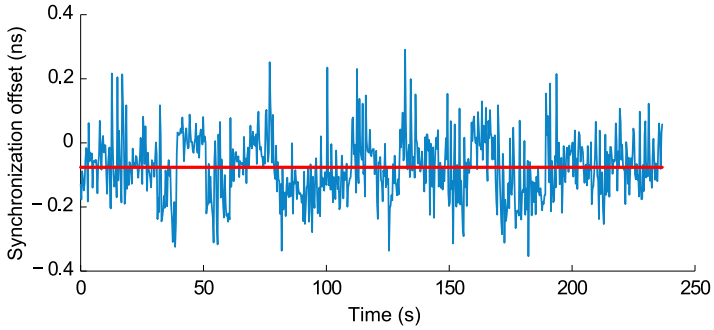


Figure 9. Residual asymmetry for 50 cm / 1m cables.

The same measurement was also executed for a cable combination of 50 cm and 60 m, where the expected propagation delay difference is 297 ns. Figure 10 shows the residual asymmetry which remains within a timing window of ±200 ps, with a mean offset of 22 ps.

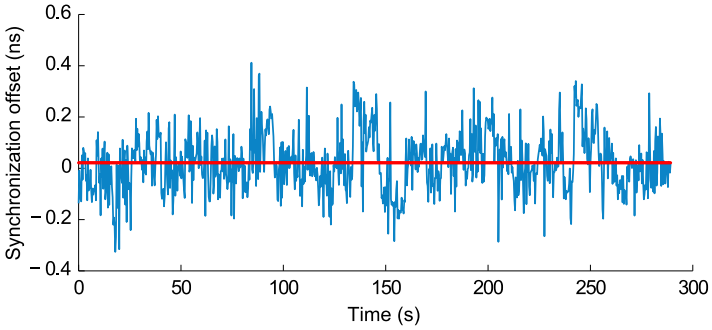


Figure 10. Residual asymmetry for 50 cm / 60 m cables.

These two measurements (Figures 9 and 10, respectively) clearly show that the residual asymmetry is independent of the actual asymmetry of the cable pair and that the method performs equally well for different amounts of asymmetry.

VI. CONCLUSION

Clock synchronization using PTP in Ethernet networks is limited, in terms of accuracy, by the inability of PTP to determine the asymmetry created by the cabling and the PHY devices. In this work we presented the line swapping method, which is able solve this issue for 100Base-TX Ethernet. Using a highly accurate timestamping method, the asymmetry can be almost compensated and an unbiased synchronization with residual offsets below 200 ps can be achieved.

It has to be emphasized that this system requires no changes on the hardware level, such as external PLLs, or other ICs as most commercial off-the-shelf PHYs have the line swapping feature built-in. As the line swapping causes a link interruption, we suggest using the asymmetry mitigation approach as start-up procedure right after the link is established. The measured asymmetry can then be used to fill the corresponding correction fields within the PTP messages without impairing the operation of the link.

Nevertheless, two open topics for future work have been identified. The work presented in this paper assumes syntonized nodes, which represents the special case of having highly stable oscillators without significant frequency skews. Further investigations need to be carried out to determine the degradation when using two different clock sources for the master and slave. The second potential issue is the asymmetry variability. The line swapping induces a link interruption of more than one second. Measures to speed up the autonegotiation are under consideration to enable periodic fast line swapping. Another solution, which is currently under investigation in the AEtas project, is to use out-of-band communication to determine the asymmetry during normal operation.

ACKNOWLEDGMENTS

This work was partly financed by the province of Lower Austria, the European Regional Development Fund, the FIT-IT project AEtas under contract 825904 and Oregano Systems Design and Consulting.

REFERENCES

[1] D. Mills, “Internet time synchronization: The network time protocol,” IEEE Transactions on Communication, vol. 39, no. 10, pp.1482–1493, Oct. 1991.

[2] H. Huckeba and R. Dlugy-Hegwer, “Precise Time Synchronization Using IEEE 1588 for LXI Applications,” in Autotestcon, IEEE, pp. 129–135, Sept. 2006.

[3] J. Ridoux and D. Veitch, “Ten Microseconds Over LAN, for Free (Extended),” IEEE Transactions on Instrumentation and Measurement, vol. 58, no. 6pp. 1841–1848, June 2009.

[4] K. Han and D.-K. Jeong, “Practical considerations in the design and implementation of time synchronization systems using IEEE 1588,” Communications Magazine, IEEE, vol. 47, no. 11, pp. 164–170, Nov. 2009.

[5] R. Exel, “Receiver Design for Time-Based Ranging with IEEE 802.11b Signals,” Int. Journ. of Navigation and Observation, No. 743625, Jul. 2012.

[6] P. Loschmidt, G. Gaderer, N. Simanic, A. Hussain, and P. Moreira, “White Rabbit – Sensor/Actuator Protocol for the CERN LHC Particle Accelerator,” Sensors Conference, pp. 781–786, Oct. 2009.

[7] M. Lipinski, T. Wlostowski, J. Serrano, and P. Alvarez, “Performance results of the first White Rabbit installation for CNGS time transfer,” Int. IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS), pp. 1–6, Sep. 2012.

[8] N.M. Freris, S.R. Graham, and P.R. Kumar, “Fundamental Limits on Synchronizing Clocks Over Networks,” IEEE Transactions on Automatic Control, vol. 56, no. 6, pp. 1352–1364, June 2011.

[9] P. Loschmidt, R. Exel, A. Nagy, and G. Gaderer, “Limits of Synchronization Accuracy Using Hardware Support in IEEE 1588,” Int. IEEE Symposium on Precision Clock Synchronization for Meas., Control and Comm., pp. 1–6, Sep. 2008.

[10] R. Exel, and P. Loschmidt, "High Accurate Timestamping by Phase and Frequency Estimation," Int. IEEE Symposium on Precision Clock Synchronization for Meas., Control and Comm., pp. 1–6, Oct. 2009.

[11] P. Loschmidt, R. Exel, and G. Gaderer, "Highly Accurate Timestamping for Ethernet-Based Clock Synchronization," Journal of Computer Networks and Communications, ID 152071, pp.1–11, 2012.

[12] L. Huang, "Compensation for Asymmetry of Physical Line," March 2011, 802.1 AVB, 201103 IEEE 802 plenary.

[13] Shuai Lv, Yueming Lu, and Yuefeng Ji, "On Enhanced IEEE 1588 Time Synchronization for Asymmetric Communication Link in Packet Transport Network," IEEE Communication Letters, Vol. 14, No. 18, Aug. 2010, pp. 764–766.

[14] R. Exel, "Clock Synchronization in IEEE 802.11 Wireless LANs using Physical Layer Timestamps," Int. IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS), pp. 1–6, Sep. 2012.

[15] IEEE, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, July 2008.