INVITED PAPER

Performance and Reliability Aspects of Clock Synchronization Techniques for Industrial Automation

This paper deals with synchronization aspects of distributed nodes connected to industrial networks, specifically, real-time Ethernet networks.

By Nikolaus Kerö, *Member IEEE*, Andreas Puhm, Thomas Kernen, *Member IEEE*, and Anton Mroczkowski

ABSTRACT | Modern distributed control systems comprise multiple intelligent devices capable of performing complex time and mission-critical tasks both independently of each other or partly jointly with each other. To do so, they strongly depend on an accurate common notion of time as well as a reliable shared communication medium for timely data exchange. Traditional legacy communication technologies (field bus systems) supported time transport to a certain extent or provided at least a common frequency. Due to its numerous undisputed advantages, Ethernet has become the only viable communication medium effectively replacing such systems. Being inherently asynchronous time and frequency transfer has to be accomplished using a packet-based approach when moving to Ethernet. After explaining the basic principles of packet-based time transfer, the most common standards are explained compared with each other with respect to their intended application domains. Special emphasis will be put on the Precision Time Protocol (PTP) as defined in the underlying IEEE 1588 standard and its variant IEEE 802.1AS used for time-sensitive networks. Maintaining a highly accurate common notion of time under all circumstances is a crucial prerequisite for most distributed systems. Although PTP has proven to provide sub-microsecond accuracies, it can cope only with a limited number of error conditions. This paper describes all major sources that can either deteriorate the accuracy or cause a total loss of synchronization altogether. Selected countermeasures and enhancements are presented, which can greatly improve the resilience of PTP against errors as well as malicious attacks. This paper concludes by presenting the selected measurements' results of a novel proposed method.

KEYWORDS | Precision Time Protocol (PTP), IEEE 1588, redundancy, reliable clock synchronization.

I. INTRODUCTION

A common notion of time or at least a common notion of frequency is a crucial requirement for any distributed embedded system, where intelligent devices have to accomplish a series of subtasks independently of each other, yet in a strictly synchronous manner. In automation systems, this is particularly crucial for diverse tasks such as distributed process monitoring or control of large plants that require coordinated control actions. Ideally, automation systems share a single communication medium for exchanging control and status data as well as distributing time or at least frequency information. In the past, a variety of different legacy technologies have been used for respective application domains. Industrial automation relied on a large variety of legacy field bus systems. In general, these systems are governed by a series of respective standards defining the various communication layers [1], whereas

Manuscript received October 3, 2018; revised March 12, 2019 and April 26, 2019; accepted April 30, 2019. Date of publication May 23, 2019; date of current version May 28, 2019. This work was supported in part by the Austrian Federal Ministry for Digital and Economic Affairs (BM:DW) and in part by the National Foundation for Research, Technology and Development as related to the Josef Ressel Center "Innovative Platforms for Electronic-Based Systems" (INES), managed by the Christian Doppler Research Association. *(Corresponding author: Nikolaus Kerö.)*

N. Kerö and **A. Mroczkowski** are with Oregano Systems, 1030 Vienna, Austria (e-mail: keroe@oregano.at).

A. Puhm is with the University of Applied Sciences, Technikum Wien, 1030 Vienna, Austria.

T. Kernen is with Mellanox Technologies Ltd., Yokneam 2069200, Israel.

Digital Object Identifier 10.1109/JPROC.2019.2915972

time-division multiplex (TDM) infrastructure such as, for example, E1/T1 or E3/T3 systems, which are specified in respective ITU-T recommendations [2] was globally deployed for traditional digital telecom networks. Due to their synchronous or plesiochronous architecture, frequency transfer could be accomplished in a straightforward manner. As the payload data were modulated or encoded onto a carrier frequency, every end node could easily extract this frequency using a suitable phase-locked loop (PLL) embedded in the receiver front-end circuit. Absolute time, Time of Day (ToD), information was transferred usually as part of the payload. Accurate phase information, however, was difficult to obtain and more often than not impossible to convey, because time transfer was unidirectional and, thus, the absolute transmission time remained unknown leading to inaccurate ToD information at the end node. Whenever only a common frequency within a network was required, such technologies were sufficient. If phase information was required as well, the transmission time from the reference node to every distinct device needed to be accounted for. This was done either by measuring it applying suitable out-of-band methods or by estimating it via some sort of cable length measurement or estimation mechanism.

As Ethernet is continuously evolving into the sole shared communication medium for nearly every distributed system, both absolute time and frequency transfer have to be provided as a basic service. In contrast to most of the technologies mentioned earlier, Ethernet is an inherently asynchronous medium if all the nodes within a network are considered. Two adjacent nodes (i.e., nodes connected via a physical cable with each other) have to establish a common frequency prior to commencing data transmission. Which node will act as a source forcing the local PLL of the other node to be synchronized to its frequency is an arbitrary process.

The purpose of this paper is to give an overview of packet-based time transfer technologies over Ethernet networks pointing out their advantages over traditional systems. Special emphasis on reliability aspects is given, which is of fundamental importance when considering distributed systems in the industrial automation domain having to perform complex distributed control tasks.

This paper is structured as follows. After describing the basic principles of packet-based time transfer, the two prevalent protocols are explained and compared with each other. In Section III, PTP is explained in greater detail, focusing on its capability to cope with different error conditions. Several enhancements are discussed to improve the resilience of PTP in that respect. Section VIII covers the specific aspect of highly accurate time transfer via redundant networks.

II. BASIC PRINCIPLES OF PACKET-BASED TIME TRANSFER

Highly accurate clock synchronization can be accomplished over Ethernet networks using a message-based approach. To this end, a two-way time transfer has to be established between a designated reference node and every node in the network requiring this service. Dedicated packets are exchanged periodically both in the upstream and downstream directions, i.e., to and from the reference node. For these, the send as well as the receive time has to be denoted by all nodes using their respective local clocks, as shown in Fig. 1 with the downstream packets denoted in green and the upstream traffic denoted in blue.

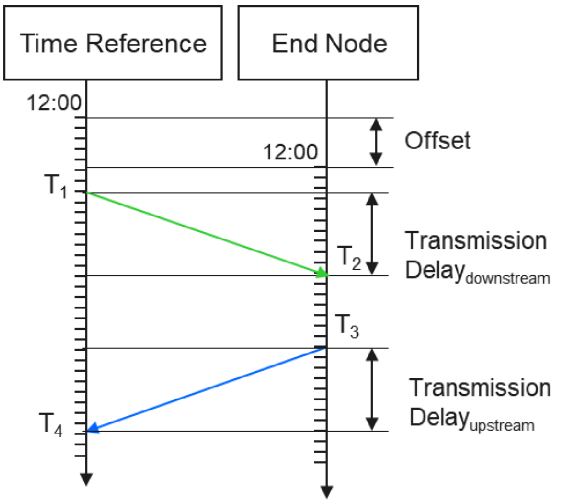

Fig. 1. *Bidirectional packet exchange for time transfer.*

In the downstream direction, such a message would contain its send time (T_1) derived from the local clock of the reference node. Upon receiving it, a node draws a time stamp from its local clock (T_2). The difference of these two time stamps equals the offset of the local clock with respect to the reference plus the transmission time of the message over the network

$$T_2 - T_1 = \text{Transmission\_Delay\_Downstream} + \text{Clock\_Offset}. \quad (1)$$

For the corresponding upstream message, a send (T_3) and receive (T_4) time stamp has to be gathered in a similar fashion from the respective local clocks. The latter, of course, has to be conveyed back to the end node using an appropriate mechanism. The difference between these two time stamps equals the difference between the transmission delay and the offset of the two clocks

$$T_4 - T_3 = \text{Transmission\_Delay\_upstream} - \text{Clock\_Offset}. \quad (2)$$

Using the differences of the send and receive time stamps in the upstream and downstream directions, both the offset and the transmission delay can be easily

calculated as

$$\text{Clock\_Offset} = \frac{(T_2 - T_1) - (T_4 - T_3)}{2} \quad (3)$$

$$\text{Transmission\_Delay} = \frac{(T_2 - T_1) + (T_4 - T_3)}{2}. \quad (4)$$

If purely frequency transfer is required, the upstream path can be omitted: using the data of a consecutive sequence of downstream messages, the average frequency offset can be calculated and accounted for as

$$\overline{\Delta}_f = \frac{\sum_{i=2}^{N} \frac{(T_{1,i} - T_{1,i-1})}{(T_{2,i} - T_{2,i-1})}}{N}. \quad (5)$$

A. Accuracy of Packet-Based Time Transfer

This approach implicitly assumes that the transmission delay is constant and identical in both the directions. Unfortunately, this assumption is not correct for modern switched networks and its violation significantly affects the achievable accuracy of every message-based synchronization technology. There are two principal factors contributing to the synchronization errors (inaccuracies), one being related to the way the time stamps are gathered at the nodes and the other how the messages are transmitted over the interconnecting network.

1) Timestamp Accuracy: Whenever a node sends a packet containing time information, the CPU inserts the time stamp while assembling the packet. Upon completion, it asserts the respective network control unit about new data to be sent. Consequently, it will be moved from the main memory to its local buffer memory and either sent instantaneously or moved to an output queue in case other network traffic is currently being processed. In the latter case, the message is further delayed. The time span between drawing a time stamp from the local clock and sending the corresponding packet via the physical channel may vary significantly, because it encompasses a number of tasks each of which could be interrupted for an unknown amount of time by other processes with higher priority within any nonreal-time operating system.

Every end node needs to denote the receive time as accurately as possible as well. This process is prone to suffer from similar inaccuracies. Message reception is done autonomously by a network subsystem, which will inform the CPU about new data after having copied a new message to an assigned buffer. The CPU will start to process this message at various levels of the operating system eventually, concluding that this is a timing message for which a time stamp has to be drawn upon reception. These processes can and will be interrupted more often than not by other tasks requiring the same hardware resources.

The variations of this time stamping method—often referred to as software time stamping—could well reach ten or even hundreds of milliseconds. It should be mentioned that real-time operating systems, although an obvious solution to this problem, can only reduce the variation to a certain extent, because communication requests by other tasks have to be handled by the same units and transferred via the same shared medium [3].

A number of proposals targeting this problem suggest drawing the time stamp as close as possible to the hardware, i.e., by enhancing the network drivers or even the underlying kernel modules [4], [5]. These methods can mitigate the problem only so much and, thus, are unsuited to synchronize clocks reliably in the submicrosecond range within arbitrary network topologies.

The only way to eliminate this problem is to draw all time stamps in hardware. This can be accomplished by adding special hardware circuits as close as possible to the physical channel. The most common location for such an extension is the media-independent interface (MII) between the physical layer IC module and the media access controller (MAC). For a number of copper- and fiber-based Ethernet technologies, multibit serial interfaces are specified in the respective standards defining the lower layers of Ethernet [6]. This hardware unit will listen to all incoming and outgoing traffic. Whenever a packet is either sent or received, a time stamp is drawn at the start of the packet (i.e., at the precise moment, the packet is actually sent over or received from the physical channel). This module will analyze the packet headers of the various cascaded protocols starting with Layer 2 data followed by the Internet Protocol (IP) and the User Datagram Protocol (UDP) [7], [8]) searching for a timing packet conformant to the time transfer protocol in question. If such a packet is detected, the corresponding time stamp is stored together with additional information about the packet such as a unique sequence number or a hash code. Otherwise, the time stamp is discarded. The time transfer protocol stack is now able to correlate every packet with its precise time stamp. The latter is provided by the hardware module via a suitable interface. The accuracy at which these time stamps can be drawn at the MII is in the nanosecond range as shown (see [9], [10], or [11]). It should be noted that the actual transmission time over the physical layer, i.e., the network cable connecting two nodes is constant and highly stable; it varies typically less than a couple of nanosecond over time and temperature for copper-based transmission systems.

2) Network-Related Inaccuracies: As long as Ethernet was using literally a single shared medium, i.e., one coaxial cable within one physical collision domain, transmission times over such a cable could be considered to be constant. For modern switched networks with intelligent network elements, this is no longer the case. These network devices are optimized for maximum throughput, therefore having to handle different classes of traffic (i.e., high versus low priority) in parallel rather than ensuring constant forwarding times for every packet. Effectively, these two requirements are contradicting each other.

These variations in the packet processing and forwarding time are referred to as packet delay variations (PDVs). If a packet is received at an ingress port, it is forwarded to the switch fabric—a hardware unit that forwards the packet to the corresponding egress port(s) according to the currently active layer-2 or layer-3 routing rules by analyzing the respective header information. It transfers it to the queues(s) of the destination port(s). If these are empty at that time, the residence time of the packet (i.e., the overall time a network devices requires to process/forward the packet from receiving it at a source port to actually transmitting it on a specific destination port) is minimal and will remain constant as long as these queues remain empty, i.e., no other traffic is being processed. In the presence of other network traffic, these queues may very well start to fill up causing the residence time to increase, or to be more precise, to vary on a per packet basis. The amount of PDV a specific device adds to a packet is not only dependent on the network load that the switch has to process at that time but also on its underlying hardware architecture [12]. Modern datacenter grade devices can cope with network loads of up to 90% without a significant increase in the PDV, which would remain in the range of several hundreds of nanoseconds [12], [13], [15]. Simpler devices typically used for industrial automation applications may not be able to cope with even medium network loads causing the PDV to increase by several orders of magnitude. One explanation of this behavior is that they need to run a number of "house-keeping" tasks in parallel to the actual packet processing. These are typically run by the local CPU of the network device and comprise, for example, buffer memory reallocation, updating of destination address tables by discarding outdated (aged) entries.

PDVs will increase even further, if the packet in question has to traverse through a number of network devices in sequence each of which may face a different loading condition at the time. It should be noted that PDVs for packets carrying time information can be reduced forcing the network device to forward such messages immediately after the current packet is sent. These packets would effectively jump the queue for the respective egress port. Expedited forwarding, as this method is referred to, can mitigate the PDV problem to a certain extent [16]. However, it cannot be eliminated altogether. Moreover, for a number of applications, expedited forwarding is already required for other classes of traffic, and thus, it cannot be used efficiently to improve the quality of time transfer.

III. TIME TRANSFER PROTOCOLS

Two different time transfer protocols have evolved in the past and both are being used in current systems. As they are intended for different use cases, they do not compete. However, both rely on the same bidirectional time transfer principle, as described in Section II: the Network Time Protocol (NTP) and the Precision Time Protocol (PTP).

A. NTP—Network Time Protocol

NTP was published by David L. Mills in the mid-1980s [17], [18]. It has been improved continuously over time with version 4.0 being the most recent release [19]. It is intended and thus optimized for transferring time over wide area networks (WANs). Time transfer is always initiated by the end node, i.e., the NTP Client, which, therefore, needs to know the address of one or more NTP Servers. To cope with even excessively high PDVs, an NTP Client can take the time information of more than one NTP Server into consideration. This is accomplished via filters and criteria on how to select or discard messages. The NTP protocol covers message structure and exchange as well as the synchronization algorithms to cope with PDVs. The more recent versions address security issues by being able to detect malicious NTP Servers sending out wrong time information deliberately.

For embedded systems, however, NTP has a number of stringent disadvantages limiting its applicability significantly. First, NTP transmits Universal Time Coordinated (UTC) time, which is perfectly justified, because the system clocks of all NTP Clients use this timescale as well. However, UTC is a discontinuous timescale, because it has to cope with leap seconds that are added (or subtracted) to the timescale at regular intervals to account for the variations of the earth's rotational speed. However, if highly accurate time stamps have to be drawn from the local clock at such a point of discontinuity, the temporal relationship of the current subtask with respect to adjacent nodes can become corrupted.

Furthermore, NTP was intended to operate with software time stamping only, resulting in a target accuracy in the millisecond range at best [20], which is by far sufficient for any desktop computer or server. Consequently, support for hardware time stamping by semiconductor vendors or IP core providers for network modules is very limited. Finally, within NTP, every time the transfer cycle is initiated by the NTP Client, which has to be made aware of the address of the NTP server. This could add an unnecessary burden when (re)configuring or deploying NTP for a distributed embedded system.

B. PTP—Precision Time Protocol

As opposed to NTP, PTP was originally intended to synchronize nodes within a local area network (LAN), although it was subsequently extended to provide support for WAN-based time transfer. PTP was first published as an IEEE standard in 2002 (IEEE 1588-2002) where version 1.0 was defined [21] and in 2008 where version 2.0 was published [22]. Only with the latter version of the standard various industries and application domains adopted PTP as the sole time transfer protocol in favor of either NTP or legacy solutions. Due to its highly generic structure combined with the ability to tailor its performance and resource usage to the requirements of a specific application via so-called PTP profiles, PTP has become the

dominant time transfer technology for highly accurate clock synchronization and was taken up by nearly every application domain, although sometimes gradually. Nowadays, the telecom industry [22], the financial services' providers [24], industrial automation, the broadcasting industry [25], and the power utility [26], [27] industry use PTP exclusively for distributing time information accurately, all having defined their respective profiles.

C. Basic Principles of PTP

In contrast to NTP, time transfer is initiated by the time source rather than the end node. Within a PTP network, one and only one node is selected as the reference, i.e., the PTP Grandmaster (GM), while all other devices become PTP Slaves [22]. The GM continuously sends Sync messages to all nodes. It should be noted that by default, all PTP messages are sent as multicast traffic, although unicast time transfer is supported as well. Depending on the hardware capabilities of the GM, it either inserts the time stamp into the message while it is actually being sent or merely denotes the precise send time providing it to all nodes as part of a corresponding Follow-up message that is sent subsequently to the Sync message. The former is referred to as one-step mode, while the latter is called two-step mode.

Every PTP Slave draws a time stamp whenever it receives a Sync message. To obtain the second pair of time stamps, it sends a Delay_Request message denoting the send time. The Master, in turn, denotes the receive time of every such message and bounces back this information by means of a corresponding Delay_Response message. The Slave is now able to adjust its local clock to the one of the PTP GMs. The PTP event message flow is shown in Fig. 2.

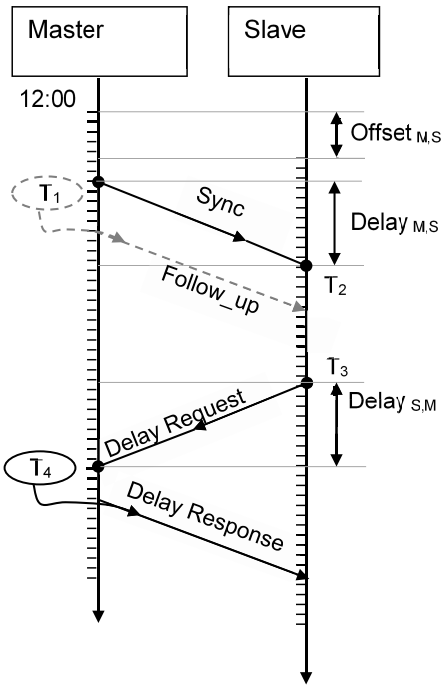

Fig. 2. ***PTP message exchange.***

D. PTP Master Election Process

Until now, we have assumed that the network has already selected a node to become its Master. This selection process is part of the PTP protocol and will be triggered and executed autonomously by all nodes whenever certain conditions occur. In parallel to Sync and Delay_Response messages, every GM sends PTP Announce messages—usually at a lower rate. These messages are used to convey the quality of the local clock of the Master to all the nodes in the network via a set of parameters. Some are user configurable allowing to control the behavior of a network, i.e., allowing only certain nodes within a network to assume PTP Master role, while others would remain in the PTP Slave mode. Other parameters are used to convey whether the PTP Master is connected to an external time reference, thus providing traceable time information.

The Master election process is triggered by the absence of Announce messages for a certain amount of time or, to be more precise, by a number of consecutive Announce messages missing. Consequently, the Announce message rate and the allowed number of missing Announce messages in a row have to match on all nodes within a PTP network. These are the most important parameters that need special attention when configuring PTP-based time transfer. Whenever a new Master is to be elected, all eligible nodes start sending Announce messages, advertising their respective clock quality to the network. All nodes have to evaluate the content of every Announce message they receive, regardless of whether they are capable of becoming a Master. They will compare them pairwise applying the Best Master Clock algorithm (BMCA) and discarding the data of the inferior Master allowing it to select the node with the "best" clock, which eventually will assume Master role. The BMCA defines a fixed precedence for comparing the various parameters with each other; it designed to guarantee that all nodes will always agree on the same node to become the new Master.

The BMCA is triggered during start-up of a PTP node because it has to listen for the presence of Announce messages prior to applying any state changes. Whenever a node with a "better" clock than the currently active Master is attached to the network, it will trigger the BMCA for the complete network and eventually take over as the new Master. To do so, it will compare the data of the Announce messages of the current Master with its own data set, concluding that its clock is better. Subsequently, it will commence sending Announce messages, which will force all nodes, and in particular, the current Master executes the BMCA forcing the currently active Master to back off and all other nodes to accept it as the new Master.

The BMCA is a fully autonomous process, which guarantees that a PTP network remains synchronized in case

one or even multiple nodes fail. This feature makes PTP perfectly well suited for time-aware distributed systems, where a highly accurate and reliable common notion of time is far more important than all nodes being synchronized to a traceable time reference. In a PTP network, such a time reference will always become the GM; however, in case it fails, the remaining network will still maintain a common time, allowing it to continue to perform all mission-critical and time-sensitive tasks. The common time will eventually drift away from the reference time, but for many applications, this is of far less concern than losing synchronicity altogether.

IV. HOW TO COPE WITH PDVs

If a certain level of accuracy has to be guaranteed in the presence of PDVs, it does not suffice to simply readjust the value of the local clock periodically: a multistage clock adjustment process has to be applied. In a typical PTP application, the rates at which both Sync and Delay_Request messages are sent are considerably higher than the rate at which messages are exchanged in NTP. High message rates (i.e., several messages per second) provide additional data for the Slave to better cope with PDVs by applying the digital filter(s) prior to feeding the offset to its control loop which is adjusting the local clock. A number of different approaches have been studied in the past ranging for PI-control loops structures preceded by linear digital filters [28], [29]. An in-depth analysis of the PI controller for clock synchronization is given in [29]. A different approach proposes a split between frequency and phase synchronization reducing the settling effects under certain PDV conditions, thus reducing the overall settling time [31]. A cascade of both nonlinear and linear digital filters is usually implemented in the servo algorithm of a PTP Slave. The former allows to remove outliers, i.e., packets with excessively high delay, which would cause transient errors to occur in a purely linear filter stage. Respective packet selection criteria have been investigated, e.g., in [32]. Other approaches suggest adaptive packet selection algorithms depending on the PDV distribution that can vary with respect to the network load [33]. Feedforward approaches using Kalman filters have been investigated as well, showing a somewhat superior performance to PI controllers with respect to robustness for a number of PDVs' patterns [34], [35].

Many of these diverse proposals for mitigating the influence of PDVs on the synchronization accuracy have proven to be impressively effective solutions, yet they all have been explicitly tailored to network environments prevalent for a dedicated application domain. A generic solution, thus highly desirable from a user perspective, will show limited capabilities to cope with arbitrary levels and distributions of PDVs, let alone with dynamic changes of them. Furthermore, it should be noted that PTP describes only a protocol without defining or even proposing any kind of control loop or suggesting filter architectures. In contrast to NTP, this is considered to be implementation specific; consequently, different PTP stack implementations cope differently well with network impairments, faults, and PDVs.

A. PTP-Aware Network Devices, How to Cope With Packet Delay Variations

PTP has been designed to work best in LAN-based network environments, and thus, it has to cope with significantly smaller PDVs than an NTP Client querying an NTP Server located somewhere on the Internet. Nonetheless, PDVs are a major concern for PTP even more so when high clock accuracy is an issue. The countermeasures described in Section IV mitigate their effect to a certain extent; however, if reliable clock synchronization in the range of submicrosecond or even sub-100 ns is requested, the interconnecting network elements have to be considered with respect to PTP.

PTP does provide two different viable solutions to address the effects caused by PDVs by defining two types of PTP-aware network devices: boundary clocks (BCs) and transparent clocks (TCs), respectively.

B. End2End Transparent Clocks

A PTP End2End TC (E2E TC) is a standard network device, which will process all messages according to the respective layer-2 and layer-3 forwarding rules, regardless whether they are PTP messages or not. PTP event messages (i.e., PTP messages carrying time information), however, are treated slightly differently. Rather than striving to keep the residence time constant thus minimizing PDVs, the residence time is actually measured on a per message basis. This is accomplished by adding time stamping logic similar to the one used in end nodes to every ingress port and egress port. This logic scans for PTP event messages, drawing a time stamp from a local clock of the TC whenever receiving such a message. This time stamp is stored either in a local buffer together with additional information of the message to enable correlating the time stamp with the packet later on or the time information is inserted into or appended to the message itself. The message is then processed by the switch according to the respective forwarding rules and copied to the respective egress port(s), where a second time stamp is drawn. Together with the first time stamp, the TC is able to precisely calculate the residence time on a per packet basis by simply subtracting the two time stamps from each other. This information is conveyed to the PTP Slave via a dedicated field within the message (correction field). Rather than simply copying this value to the field, the residence time is added to the value already stored in the correction field. This allows cascading TCs without sacrificing accuracy. The Slave can account for the overall residence of all TCs that the packet has traversed through by simply adjusting the send time stamp of the message with the data in the correction field. The formulas for calculating the offset (4) are now correct

because the residual transmission delay is caused only by the physical medium itself and thus can be assumed to be constant. The residence time measurement is applied both in the upstream and downstream directions.

C. Peer Delay Mechanism

The message exchange procedure described earlier is referred to as End2End delay mechanism, because all messages traverse through the whole network between Master and all Slaves in both the directions. For daisy chain network topologies that are typical in industrial automation applications, a different approach turned out to be a more efficient solution. Within a complex and long assembly line, the number of network hops can easily exceed 50 or even 100. For such topologies, the End2End delay mechanism has some shortcomings, especially during a Master change, if the new Master happens to reside at the opposite end of such a chain viewed from the Slave's perspective. Such a Slave node would have to cope with a significant increase or decrease of the absolute transmission delay that could cause settling effects to occur in the local control loop of the nodes.

Furthermore, large PTP networks applying the End2End delay mechanism may suffer from another drawback regardless of their topology. The large number of delay-request/response messages can cause a significant burden for every node, because these messages are sent as multicast traffic. Every Slave has to process all PTP messages within its respective protocol stack just to discard all Delay_Request/Response messages except for one single Delay_Response message actually addressed to it. Furthermore, the GM has to respond to all delay-request messages. Embedded systems used for industrial automation applications have limited computing resources to comply with constraints in power dissipation, available space, and costs and, thus, are not very well prepared to handle high PTP message loads.

To overcome these shortcomings, PTP has specified a variation of the upstream part of the message exchange mechanism where the upstream traffic is kept link local, i.e., messages are exchanged only between two adjacent nodes. In the peer-delay mode, every node within a PTP network operating in Peer Delay mode has to continuously measure the physical link delay of every of its ingress ports with respect to the respective adjacent node. This is done using a distinct set of messages. A device operating in the peer delay mode will send a Peer_Delay_Request message periodically to its adjacent node, denoting the send time (T_{01}). Upon receiving such a message, a time stamp (T_{02}) is drawn and a Peer_Delay_Response message is returned, for which in contrast to a Delay_Response message, a send time stamp (T_{03}) is drawn as well. The node having initiated this transfer will draw a time stamp (T_{04}) upon receiving the PDelay_Resp message. In the two-step mode, the missing time stamp (T_{03}) needs to be bounced back to the querying node via a Peer_Delay_Response_Followup message. The complete message exchange is kept link

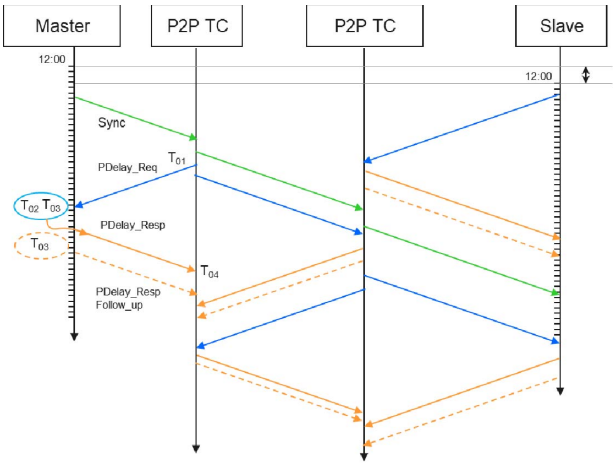

Fig. 3. *Peer-delay message flow.*

local, i.e., none of the messages mentioned earlier is actually forwarded to other devices in the network, they are processed and always returned to the sending node. The corresponding message flow is shown in Fig. 3.

The querying node is now able to calculate the absolute transmission delay (7) via the physical link to its adjacent node using the differences of the respective time stamp pairs [see (6)]

$$T_{02} - T_{01} = T_{21}, \quad T_{04} - T_{03} = T_{43} \tag{6}$$

$$\text{Ingress\_Delay} = \frac{T_{21} + T_{43}}{2}. \tag{7}$$

This messages exchange has to be performed periodically by every node on all of its ports. After the ingress-delay has been calculated using a series of measurements to minimize the influence of quantization errors, the node can use these data to calculate the absolute transmission delay for all downstream traffic (i.e., Sync messages) on a per message basis. An end node will simply account for the ingress delays to the next node. However, switches have to operate as Peer2Peer TCs. These devices follow the same approach of measuring the residence time by drawing ingress and egress time stamps for every Sync message. Instead of updating the correction field with the residence time of the Sync message alone, they consider the ingress delay of the respective port as well

$$\text{Sync\_Corr\_Field} = \sum \text{Residence\_Times} + \sum \text{Ingress\_Delays} \tag{8}$$

$$\text{Offset} = T_1 + \text{Sync\_Corr\_Field} + \text{Ingress\_Delay\_Slave}. \tag{9}$$

An end node is now able to account for the absolute transmission time of every Sync message by evaluating its

correction field using (9) and thus can directly use the data in the Sync message to adjust its local ToD clock.

D. Boundary Clocks

PTP BCs are more complex PTP devices than TCs. Rather than simply being forwarded, PTP messages are processed by every port, because all ports of a BC are acting as PTP nodes or ordinary clocks (OCs). However, the ports on a BC do not act independently of each other with respect to PTP. Furthermore, they all share one single local clock rather than maintaining individual time bases. On a BC, one and only one port will switch to PTP Slave state, gathering the time information provided by an external GM in order to adjust the local clock of the BC accordingly. All other ports will assume PTP Master Role, accessing the time information of the local clock and transferring it downstream to Slaves attached to their ports.

The BC participates actively in the BMCA, taking into account the data of Announce messages received on all ports. As a result, the port connected to the Best Master will switch to the Slave state, while all other ports assume Master role. Needless to mention that BCs can be cascaded as well, whereas a path to the GM is established traversing through the chain of BCs. With a BC, a PTP network can be segmented or, put in other words, BCs support hierarchical time transfer. As certain parameters can be modified on Master ports with respect to the Slave port, a BC can be used to provide timely information with different configuration settings such as message rates to various subnets or segments.

BCs have one further advantage to TCs: as they terminate all PTP traffic rather than forwarding it, the overall (i.e., network wide) network load with respect to PTP messages is greatly reduced. This is especially advantageous for large networks with a high number of PTP nodes.

BCs do have one significant disadvantage as well. Being an active PTP device, the port in the Slave mode needs to adjust the local clock. Depending on the architecture and design of the servo loop, this process may take considerable time during which the time information conveyed to the output ports is inaccurate. During the settling period, the servo loop could oscillate or at least overshoot, and this again is forwarded to all Slaves whose servo loops have to cope with these settling effects causing them to oscillate transiently as well; at least the lock time is extended for all devices. If several BCs are cascaded, this start-up effect could increase the settling time by at least one order on magnitude.

V. IEEE 802.1AS—ANOTHER PTP PROFILE?

Time-sensitive networks (TSNs) are gaining more and more attention for industrial automation and automotive applications. They are governed by a series of IEEE standards that allows building Ethernet networks where the overall transmission latency does not exceed certain upper bounds for specific traffic classes. IEEE 802.1Q defines the respective scheduling and traffic shaping mechanisms [36]. To this end all, nodes within a TSN network need to be synchronized tightly with each other. The IEEE 802.1AS [37] was defined as the underlying synchronization protocol. This standard originates from IEEE 1588 and can be considered a PTP profile; however, it has a number of slight but important differences. First of all, the two standards share the same bidirectional message exchange as well as the definition of all respective messages; 802.1AS assumes the whole network to be "802.1AS capable," i.e., all devices (end nodes and switches alike or bridges as they are referred to in the 802.1AS terminology) have to support the standard. The Master election process is controlled using the same BMCA; however, it is triggered slightly differently because the respective time-outs for Announce messages are defined in a different way. The IEEE 802.1AS is limited to operate as a Layer-2 protocol only using the peer-delay mechanism exclusively.

The most important difference with respect to implementing an end device is that within 802.1AS, all clocks used to draw time stamps for event messages have to be kept free running rather than being permanently readjusted to follow the time of the Master as closely as possible. This requirement sounds rather strange and one would think that it contradicts the primary purpose of clock synchronization within a network. To understand its benefits, one feature of IEEE 802.1AS needs to be highlighted. The PTP peer-delay mechanism is used not only to evaluate the absolute transmission delay between two adjacent nodes but also to measure the frequency difference of their respective local clocks, i.e., the neighbor rate ratio as it is referred to in the standard. This information is transmitted via the type length value (TLV) field attached to the respective event messages. The accumulated frequency offset between the Master and a specific end node together with the respective send and receive time stamp of a Sync message (and of course the data in the correction field) is sufficient to precisely calculate the current time. At every given point in time, every 802.1AS node is able to calculate the offset of its local clock to the Master and thus knows the precise time without having run a complex servo loop at every node. Whenever a node is attached to an 802.1AS network, it is more or less instantaneously synchronized, i.e., with the first valid Sync message (assuming that one Peer_Delay message exchanged has been completed), it can calculate the offset of the local clock.

It should be mentioned further that every node has to evaluate and communicate the neighbor rate ratio before being admitted to an 802.1AS network. Access will be granted only if this rate is less than 100 ppm. This paradigm has one significant drawback on the system level. If a node needs to provide highly accurate time information to other local hardware modules enabling them, for example, to initiate the gathering of sensor data based on a global event, a second hardware clock needs to be implemented in every node. This second clock has to

Table 1 Comparison of Basic PTP Requirements

Application Domain	Basic Requirements		
	Accuracy	*No. of nodes*	*NW-Type*
Telecom	< 1 µs … < 10 ns	>> 1000	WAN (LAN)
Ind. Autom	< 10 µs … < 500 ns	100rds	LAN
Finance	< 100 µs ... < 1µs	> 1000	LAN
Power Industry	< 1 µs	100rds	LAN
Broadcasting	< 1 µs	10 ... > 1000	LAN
TSN	< 1 µs	> 500	LAN (WAN)

Table 2 Comparison of PTP Network Requirements

Network related Information		
PTP NW Support	*Preferred Topology*	*Topology Variations*
none / full / partly	metro area networks	strictly fixed
mandatory	daisy chain (mesh)	infrequent changes
optional	spine / leaf	mostly fixed
mandatory	daisy chain	strictly fixed
optional	spine / leaf	highly variable
mandatory	daisy chain (mesh)	mostly fixed

be actually synchronized to the Master using a suitable servo loop; otherwise, synchronized events could not be generated by all nodes in a network. Both of these clocks have to be coupled in a way that their respective time information can be retrieved simultaneously; otherwise, the servo loop yields inaccurate results.

VI. PTP PROFILE COMPARISON

Being a highly generic protocol, PTP had to provide adequate means to tailor and adapt it to the requirements and, more importantly, the constraints of specific application domains without sacrificing the advantages of packet-based time transfer. Table 1 attempts to give a summary of basic requirements for the six most common PTP application domains. The values for the required accuracy are either derived from the respective profile documents or their complementary standards' documents. The number of nodes is to be understood as a rough estimate highlighting the scale of the target networks.

Table 2 shows the network-related information for the same six application domains (the data are ordered in the same way as in Table 1). The first column designates whether the respective profile requests the interconnecting network to support PTP. The profiles do distinguish even further on which types of network devices (TCs or BCs) are to be deployed or whether End2End or Peer2Peer delay shall be used. Furthermore, they specify the communication methods as well, i.e., whether PTP packets are to be sent as unicast or multicast messages using Layer 2 or Layer 3 as a transport protocol.

Tables 1 and 2, but mainly the latter one, explain the necessity of defining a PTP profile for an application

Table 3 Summary of PTP Profiles

PTP Profiles	
Application Domain	*Profile / Standard Name*
Telecom	G.8265.2, G.8275.1, G.8275.2
Ind. Autom	IEC 62439-3
Finance	Tictoc Working Group (still draft)
Power Industry	C37_238_2011, C37_238_2017
Broadcasting	AES67, ST2059-2
TSN	802.1AS

domain. The requirements and constraints are simply too diverse to be satisfied by a "single" protocol. Unfortunately, most of the PTP profiles are incompatible with each other, and some of them are even mutually exclusive, i.e., cannot exist on the same network infrastructure. This, of course, is a more than acceptable drawback from an end user's perspective. The complexity of compliance testing, on the other hand, has become significant.

Finally, Table 3 shows the profile names for the application domains, most of which are standard documents in their own right published by the respective standard organization, namely, the IEC, the IEEE, the SMPTE, the ITU and the AES.

Apart from fairly similar requirements with respect to the accuracy, all application domains share the need for highly reliable time transfer, because all of them assume PTP to operate as a basic service providing time information capable of coping with arbitrary errors without deterioration or even loss of accuracy.

To better highlight the fact that the PTP has been selected as the time transfer protocol for applications requiring highly accurate yet reliable common notion of time in favor of NTP, the two protocols are compared in Table 4.

VII. ALTERNATE TIME TRANSFER METHODS

Ethernet-based time transfer protocols, such as PTP or NTP, have the indisputable advantage of being able to be deployed on a network shared by other applications rather than requiring a distinct infrastructure solely for time transfer. Prior to PTP-enabled devices being readily available let alone Ethernet-based network technology of sufficient performance, the latter option remained to only viable solution, if accurate time information was to be provided for distributed systems. Back in 1952, the Inter-Range Instrumentation Group (IRIG) was established by the U.S. guided missile test ranges [38] to specify information exchange on range instrumentation. Today, it is responsible, among others, for publishing and maintaining telemetry standards, with the set of IRIG serial time code formats being the most popular. They are

Table 4 Comparison Between Time Transfer Protocols

Comparison of time transfer protocols		
Feature	***PTP***	***NTP***
HW Support	Readily available	Sparsely supported
Networks	Optimized for LANs	Optimized for WANs
Standard	IEEE standard	Industry standard
EPOCH	TAI (continuous)	UTC (discontinuous)
Control Loop	Implementation specific	Defined
Transfer	Initiated by Master	Initiated by Client
Accuracy	< 1μs … < 100 (10) ns	>10 ms ... 100 ms
Fault Tolerance	Implementation specific	Extended support

published in the respective IRIG standards document No. 200 [39]. IRIG-B, for example, defines a 1-Hz synchronization pulse followed by a serial bit stream of binary-coded decimal (BCD) absolute time information.

Besides transmitting the data in baseband, IRIG-B can be modulated onto various carrier frequencies. Absolute time information can be transferred highly accurately (i.e., in the range of fewer than 100 ns) [40]. However, being a strictly one-way method, the absolute transmission delay has to be accounted for via appropriate out-of-band measurement techniques. Using modern user programmable digital devices [field-programmable gate arrays (FPGAs], both encoders and decoders are fairly straightforward to design [41]. Although IRIG-B is gradually superseded by network-based protocols, it still has its merits whenever the network elements do not support PTP sufficiently well.

Global Navigation Satellite Systems (GNSSs), such as the Global Position System (GPS) [43] or Galileo [44], transmit highly accurate time information on a global scale. Using special GNSS timing receivers, every node within a distributed system can be supplied with absolute time information in the range of sub-50 ns. Traditionally, GNSS-based time information was merely used to synchronize PTP GMs to an external time reference. Nowadays, the performance of modern semiconductor technologies facilitates the design of small, low-power devices well suited even for mobile applications. In [45], a fully integrated device is presented with less than 5-mA current consumption. In [45] and [46], the experimental results on GNSS-based synchronization for automotive applications are given.

Yet, GNSS-based synchronization will not become an adequate replacement for PTP for the majority of distributed systems in the foreseeable future. Although the costs for timing receivers have dropped significantly, they will remain more expensive than PTP-enabled network devices not least because an RF antenna is a mandatory requirement.

The far more stringent limitation, however, is the fact that every device, or to be more precise the antenna of every device, requires a direct line of sight to the GNSS satellites, i.e., a partially unobstructed view to the open sky. This limits the use of GNSS for in-door applications to a large extent. Several attempts have been made to overcome this obstacle, for example, by improving the sensitivity of the analog front end and the digital signal processing capabilities of the RF receiver (see [48], [49]) or by involving additional out-of-band information using signals of opportunity (SoOP) such as RF signals transmitted by mobile telecom network operators [50].

Finally, it should be mentioned that GNSS-based time transfer is prone to a variety of error conditions, ranging from weak satellite signals to wrong time information altogether. While the former typically is related to an obstructed view to the open sky, bad (solar) weather conditions, or jamming devices, the latter could be caused by RF-interference of malfunctioning devices adjacent to the antenna or even by spoofing attempts. Thus, relying solely on GNSS as a time source for synchronizing distributed systems with multiple nodes performing mission-critical tasks should be considered too risky.

VIII. RELIABLE CLOCK SYNCHRONIZATION

Providing and maintaining a highly accurate common notion of time reliably even under harsh or adverse operating conditions are mandatory for distributed systems. Therefore, the selected time transfer technology has to be able to cope with a variety of error conditions without the system-wide accuracy being deteriorated or losing synchronicity altogether.

For message-based time transfer protocols such as PTP, the effects of different types of error conditions have to investigated independently of each other. The most obvious error to occur is transient or permanent loss of the communication path from reference node to one or more end nodes. It has to be noted that partial disruptions have to be considered as well, i.e., only a subset of the messages reaches the end node, while others are lost. Especially for PTP, this is by no means a hypothetical case, because PTP messages may be processed differently by network devices depending on their content.

The quality of the time information an end node receives, or the lack thereof, has to be considered as an equally critical error condition as well. This could originate from either a malfunctioning reference node, the interconnecting network can cause excessive PDVs, or the time information could be tampered with deliberately. How to detect these errors and more importantly how to react to them or even prevent any adverse effects by applying suitable countermeasures will be described in the remaining sections. A summary of possible error conditions and monitoring methods is given in [51].

A. PTP Message Loss

The basic principle of master–slave-based clock synchronization is that one single master should be active in the system at one time. As the BMCA is based on time-outs

of Announce messages, a PTP network can tolerate the complete loss of the Master. Although both the detection and the reselection process take a finite amount of time, i.e., a multiple of the time span between two Announce messages. During that period, the local clocks of all nodes will fly-wheel and thus drift apart. As a prerequisite, at least two nodes being configured to assume Master role have to be present in the network. For many application domains, this requirement is fulfilled by default, because the majority or even all nodes will be capable of both roles. This is especially useful for deployments where maintaining a common notion of time at all costs is more important than being synchronized to an external traceable time source. The effects of a Master failure on the overall accuracy of a PTP network can be estimated fairly accurately, taking the respective message time-outs as well as stability of the local frequency reference into account.

The BMCA, however, covers only a subset of possible error conditions. If event messages get partially or completely lost or corrupted, the Slaves will remain in their current state without receiving any or insufficient time information. This can happen, for example, if the hardware for drawing (and inserting) time stamps at the Master is malfunctioning. As these messages are modified within TCs, they can become likewise corrupted by transient or permanent faults within the respective hardware units. By default, PTP Slaves cannot remedy this error condition by selecting a different Master they are limited to detecting and reporting it.

Finally, a node or a group of end devices can be subjected to a faulty network connection, resulting in unexpectedly high packet loss. As a consequence, the synchronization accuracy can deteriorate and even more critical that the affected nodes can falsely detect the loss of the Master and initiate the BMCA with the node attached to the affected network segment selecting a different Master for a transient period of time.

B. Loss of Accuracy

When operated in non-PTP-Aware network environments, modern PTP implementations can cope with a certain amount of PDV. To this end, they are optimized for specific network conditions and underlying message rates to provide resiliency tolerating a certain level of PDV while keeping the settling time of the servo loop reasonably low. If the PDVs exceed the expected range, the servo may not be able to lock to the Master any longer with the requested accuracy. This condition can be induced either by failures within the network devices or by errors in the end user application itself causing some or all nodes to generate excessively high network traffic.

Although using a fully PTP enabled network is the most efficient way to eliminate the effects of PDVs once and for all it cannot be applied in every environment, especially if PTP has to be deployed on the existing network infrastructure. Furthermore, malicious attacks will remain undetected.

C. Countermeasures

Providing a reasonable number of Master devices that are usually equipped with high-grade local oscillator and traceable time information that they obtain via GNSS timing receivers is a mandatory first step to provide redundant time transfer capabilities for a network. To cope with most of the other error condition mentioned earlier, a series of countermeasures is called for.

The most promising approach is to provide time information on multiple paths ideally originating from different time sources (PTP GMs) at the same time to all nodes. This can be accomplished in various different ways while remaining within the bounds of the IEEE 1588 standard. To cope with PDVs especially in WANs, the benefits of providing time information via multiple paths has been investigated in detail [52]–[54]. Different approaches have been proposed to make the best use of this additional time information ranging from applying simple linear filters to more complex designs using nonlinear methods on top.

This redundancy concept can be pushed even further by deploying multiple active GMs within a network. The basic principle was discussed already in [55]. Rather than colocating these devices, they should be positioned at opposite ends of a network, thus providing time transfer via distinctly separate network paths. This concept does not contradict or violate the strict Master–Slave principle defined in PTP. If unicast rather than multicast is used for time transfer, a node can receive data from more than one GM at the same time. To this end, PTP has specified a specific negotiation mechanism. It is initiated by every PTP Slave to establish the bidirectional time transfer with a GM.

PTP introduces the concept of a PTP domain, designated by an 8-bit number within the header of every PTP message. According to the IEEE 1588 standard, a PTP node (port) shall operate only in one PTP domain, discarding messages originating from other domains. Redundant time transfer can utilize this concept by configuring GMs to operate in different PTP domains. The Slaves have two separate instances of the PTP stack for every domain to process all messages according to the rules within the PTP protocol. The time information gathered by the respective instances, i.e., the offsets (and their variation) to the local clock has to be combined to joint time information.

If the data are combined without specifically considering network impairments, several fault-tolerant weighing functions and techniques have been proposed in the past, which are capable of discarding excessively wrong time information, i.e., GMs (deliberately) distributing false time (see [56], [57]). To better cope with PDVs, the dynamic behavior of the interconnecting network has to be taken into account, because the variation in the network load will affect the time information received from various sources differently. Preprocessing the data of different time feeds independently of each other, and combining these streams via a suitable weighing function that can be adjusted dynamically taking statistical data gathered during the

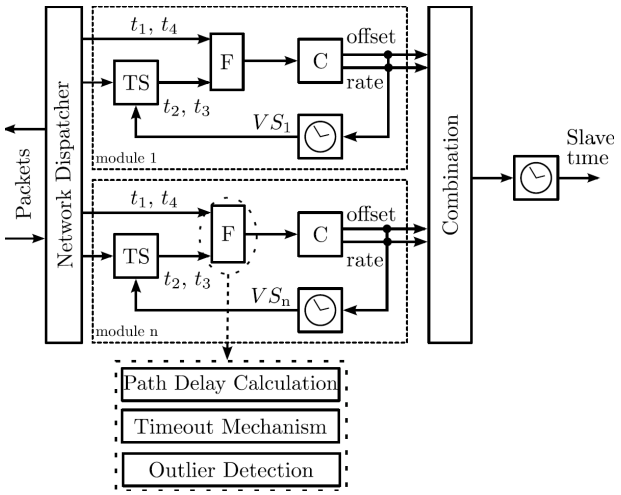

Fig. 4. ***Block diagram of multiple-timing-feed processing architecture.***

preprocessing stage into account has been proposed in [58]. A corresponding block of such a system is shown in Fig. 4.

The modules are processing the time information independently of each other using a series of low-pass input filter stages completed by nonlinear stages. Using the statistical data, these blocks discard packets with excessive PDVs. Finally, each PTP instance adjusts its respective local clock, referred to as a virtual clock using Kalman filters within its control loops.

The quality of the outputs of the respective servo loops is assessed by a special combination function by means of predictive filters. The final output is then used to adjust the offset of the physical hardware clock. This approach requires a hardware structure with two hardware clocks similar to an IEEE 802.1AS capable device. One clock is left free running and will be used for gathering time stamps and as a reference for the respective virtual clocks, while the other is adjusted by the combination function. Both the clocks have to be interlinked with each other, allowing time stamps to be drawing from both clocks simultaneously.

Fig. 5 shows a typical simulation result of this technique where a Slave receives time information via PTP messages from two different Masters in parallel both of which are subjected to PDVs varying over time. The event message rates were set to 8/s in both the upstream and downstream directions. A nonlinear prefilter was applied removing outliers using decision methods based on statistical information, i.e., both mean value and standard deviation being calculated using a window's size of 30 s. All clocks were adjusted at a rate of 1 Hz. The results were obtained by performing a time discrete simulation using an enhanced model for the crystal oscillator typically driving such devices which was applied to account for various impurities affecting its frequency [59]. The transmission time between two end points was subjected to noise using a standard Gaussian distribution, thus simulating network PDVs in the microsecond range. The level of the PDVs was modulated between two Master nodes over time. Each of the virtual clock modules could cope with the PDVs to a certain extend yet was not able to eliminate its effects completely. The subsequent combination function modulated the weights of the two inputs according to their respective residual noise level prior to combining the data.

During the simulation, the amplitude of the PDVs was varied between the two GMs over time and the Slave adjusted the weighing function accordingly synchronizing to the best time feed at the time. The green and red graphs, respectively, show the output of the independent clock modules, i.e., the virtual Slaves, while the blue graph depicts the resulting output after having applied the weighing function to the outputs of the two virtual clocks and feeding the result to the control loop of the actual hardware clock. It can be seen clearly that this approach yields the results far superior to a simple averaging function.

It should be noted that using multiple PTP domains within one network excludes the usage of standard BCs, unless they support this extended redundancy technique. TCs, on the other hand, will most likely support this approach. The same caveats have to be considered when using unicast as a transport mechanism. Again, standard BCs would have to be ruled out, because their PTP support is limited to multicast. Depending on their respective internal architecture, TCs should cope with unicast traffic originating from different GMs.

IX. PTP IN REDUNDANT NETWORKS

The violation of constraints of industrial applications (e.g., packet loss and, therefore, missing packet delivery deadlines) can cause malfunctions of the production line or even have serious impacts on the safety of human lives. Therefore, it is necessary to solve the inherent problems of an event-driven, packet-based, single-channel communication system, i.e., IEEE 802.3 in a more general sense not limited to time transfer to comply with such stringent requirements, for example, the communication medium as the single point of failure and nondeterministic packet

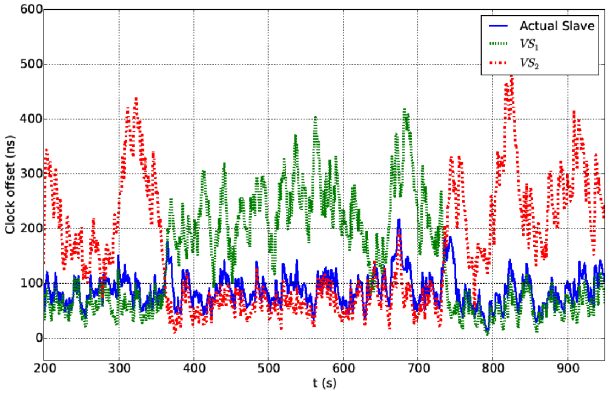

Fig. 5. ***Simulation of dynamic weighing of two different time feeds.***

delays. The provision of a common notion of time is a basis for solving these problems (e.g., TSN) but is also influenced by them, especially by the communication medium as a single point of failure.

The general approach and most effective way to solve the single point of failures in the network infrastructure is the application of redundancy on the physical level. For example, the Spanning Tree Protocol (STP) or Rapid STP (RSTP) is widely in use for Ethernet-based systems. These protocols provision redundant, passive links in the network infrastructure to be automatically used, in case of a fault in the primary, active link. When applying IEEE 802.1w RSTP [58], the time from detection of the fault in the system, to switching to the backup, is above 30 s for STP, above multiple seconds as shown in [61] or down to a few 100 ms [62] for IEEE 802.1D-2004 RSTP [63].

In the context of IEEE 1588 clock synchronization, a switchover time of a 100 ms will have no severe consequences (i.e., loss of a few synchronization messages, in which both the protocol and the local servo have to tolerate by default). A switchover time of more than 30 s can lead to unacceptable inaccuracies jeopardizing the integrity of the data exchange in industrial automation systems. The respective standard, IEC61850-5 [64], specifies different classes of timed traffic with TT6 being the highest transfer time class, requiring that every such message is guaranteed to reach its sink within 3 ms. This transfer time includes the data transfer through the communication stack on both sides, in addition to the actual transit time on the network. To fulfill this requirement, a system with zero switchover time between two independent physical channels is necessary which is often referred to as seamless redundancy.

A. Corresponding Redundancy Standards

Some standards specify (seamless) redundancy mechanisms as system-inherent components. In the context of TSN, IEEE 802.1CB [65] specifies data frame replication and elimination, and of course, IEEE 802.1AS specifies time transfer. These standards do not specify the network topology that has to be used, i.e., it is possible to apply them to various topologies. Avionic communication systems have addressed this topic as well with ARINC 664 part 7 [66], specifying a seamless redundancy mechanism for data frame handling and a network topology for redundant communication paths.

In the context of digital power grids, the IEC 61850 standard suite references IEC 62439-3 [67], namely, Clause 4: Parallel Redundancy Protocol (PRP) and Clause 5: High availability Seamless Redundancy protocol (HSR) that specify redundancy mechanisms for data frame handling, namely, two different network topologies and corresponding data exchange mechanisms for redundant communication paths. To this end, every node has to be equipped with two physical network interfaces.

Accompanying these standards, IEC/IEEE 61850-9-3 [68] specifies an IEEE 1588 profile and IEEE C37.238 [69] specifies an extended IEEE 1588 profile for use in substation automation systems. Reference [70] provides an overview of these standards.

B. Seamless Redundancy

A common goal of these redundancy mechanisms is the transparent handling of the redundancy for the application. A frame is duplicated on the source side (or near to the source) without interaction of the actual application. The duplicates are sent over at least two, ideally distinct, paths to the sink.

These paths are either duplicates of the complete network (e.g., PRP, ARINC 664 Part 7) or clockwise and counterclockwise directions of a closed ring (e.g., HSR). Discarding duplicates is handled at or near the sink. To detect duplicates, additional data are added to the packet during duplication and removed from the packet by the duplicate detection mechanism to provide this as a transparent service to higher layers.

Redundancy mechanisms using a ring topology (HSR) typically prepend a new header to provide this information. This allows faster access to the relevant data to decide, if the packet is to be forwarded on the ring, and/or to the application running on the node, or if it can be removed. In case of HSR, every node has to remove packets already forwarded to eliminate the problem of endlessly looping packets that accumulate and exhaust the network bandwidth.

Redundancy solutions that use a duplicate network infrastructure typically append this information at the end of the packet before the frame check sequence. This allows the operation of nonredundancy-aware nodes in the redundant networks. If this redundancy principle is used for time transfer, a simple selection packet-based selection criterion will yield inaccurate results, because the transmission delay will most likely be different along the two paths and, thus, will likely introduce significant PDVs.

C. Implications of Seamless Redundancy for Clock Synchronization

Having access over two (or more), physical paths to the same Master (or to two different Masters) provide new opportunities for clock synchronization, extending the simple mechanisms described in the standard mentioned earlier. For example, the redundancy principle described previously can be applied by comparing the accuracy of the clock synchronization or the PDV, selecting the better path as the primary source, switching whenever the quality of the paths change, or even combining the time information of the redundant paths. In [58], such an approach is described.

The clock synchronization can utilize this only if the redundancy mechanism differentiates between application

data and clock synchronization packets, i.e., clock synchronization packets are not duplicated. This is necessary, as a two-way clock synchronization method (IEEE 1588) assumes that packets of a single time exchange (i.e., synchronization and delay request) travel over the same path. If this cannot be satisfied, it would dynamically add asymmetry to the clock synchronization. Dynamic asymmetry is a major concern for IEEE 1588, as it can neither be detected nor eliminated within the synchronization system. If the redundancy mechanism properly handles clock synchronization, this can be used to implement new monitoring mechanisms [71].

TSNs (IEC 802.1AS) and standards for the power grid (IEC 61850-90-3 and IEC 62439-3) have taken clock synchronization into account when specifying redundancy mechanisms. Hence, the clock synchronization can freely utilize the network redundancy without being impacted by it. The annex of IEC 62439-3 includes a set of examples and use cases for different scenarios for implementation (and operation) of PRP and HSR redundancy protocols together with the IEEE 1588 clock synchronization.

D. Outlook

The referenced redundancy mechanisms (especially PRP and HSR) and IEEE 1588 clock synchronization have been written for wired communication in the lower layers of the protocol hierarchy (OSI layer 2 IEEE 802.3 Ethernet, in case of PRP and HSR, layers 2 and 3 in case of IEEE 1588).

If an industrial automation system needs to utilize a more sophisticated network hierarchy, i.e., IP subnet segregation via layer 3 routers, it would break the redundancy. This is especially true for systems using PRP. Reference [73] describes this problem and proposes an extension to PRP that would allow the connection of IP subnets via single routers without breaking the PRP redundancy. Reference [74] proposes a solution similar to PRP that can support a wide range of network hierarchies, e.g., the connection of two power substations via a telecom backbone network.

Beside optimization or extension of redundancy mechanisms for wired systems, the application of redundancy mechanisms is explored for wireless networks. For example, [75] proposes the basic steps to improve the reliability of wireless local area network (WLAN). Reference [76] proposes PRP over IEEE 802.11 and extends this to a distinct WLAN redundancy solution in [77]. The impact of redundancy in WLAN on the reliability has been given in [78], and more current results have been presented in [79].

A side effect of applying seamless redundancy to wireless communication is the improvement of communication latency, as the first packet to arrive is forwarded to an application. If the reliability and latency of wireless communication can be improved far enough to satisfy the requirements of industrial automation systems, its utilization as network medium complementary to wired Ethernet becomes feasible. Hence, wireless clock synchronization will have to be added too. An overview of different wireless clock synchronization strategies and problems is given in [80].

X. FUTURE TRENDS

As PTP has become more or less the golden standard for highly accurate time transfer, research has been focusing on improving its resilience to numerous error conditions as well as deliberate attacks. In contrast to NTP, PTP defines only a protocol for time transfer without any implementation specific details or even guidelines on how to best recover the time information provided by the GM. Using multiple time sources in parallel turns out to be the method of choice and is investigated by several groups. It is expected to gain even more interest as the necessary hardware support (i.e., multiple virtual clocks capable of drawing accurate time stamps) is becoming more and more available. It should be mentioned that security of time transfer is currently investigated as a complementary measure to improve its resilience against deliberate attacks.

XI. CONCLUSION

Clock synchronization via inherently asynchronous communication channels can be accomplished by applying a bidirectional message-based time transfer. Submicrosecond or even sub-100-ns accuracy requirement can be met using hardware-assisted time stamping techniques. PTP has evolved as the protocol of choice for numerous application domains owing to the fact that it can be customized easily to specific requirements via PTP profiles. Furthermore, hardware time stamping capabilities are supported both by network and end device manufacturers as well as semiconductor vendors and IP core providers.

If synchronization is considered mission critical, the basic redundancy mechanisms provided natively by PTP are insufficient to cope with all possible error conditions such as intermittent network failures or sudden and unexpected variations in PDVs, let alone malicious attacks. The most versatile and resilient way to counteract these errors is to provide accurate time information by multiple independent sources traversing the network via different paths or using different networks altogether. If the implementation of the control loop at the Slave is enhanced accordingly, it can make the best use of this additional time information and, thus, cope with nearly all transient and permanent error conditions.

Finally, it should be mentioned that several legacy solutions or custom extensions to PTP have been proposed with the aim to mitigate certain shortcomings of standardized time transfer for specific applications (see [81]). Such approaches have been considered out of the scope of this paper, because the authors believe it is mandatory to adhere strictly to the respective standards to guarantee utmost interoperability when deploying time transfer solutions for large multivendor systems. ■

REFERENCES

[1] T. Sauter, "The three generations of field-level networks—Evolution and compatibility issues," *IEEE Trans. Ind. Electron.*, vol. 57, no. 11, pp. 3585–3595, Nov. 2010.

[2] *Physical/Electrical Characteristics of Hierarchical Digital Interfaces*, document ITU-T G.703, 2016.

[3] B. Villain, M. Davisy, J. Ridouxy, D. Veitchy, and N. Normand, "Probing the latencies of software timestamping," in *Proc. ISPCS*, San Francisco, CA, USA, 2012, pp. 1–6.

[4] A. Machizawa T. Iwawma, and H. Toriyama, "Software-only implementations of slave clocks with sub-microsecond accuracy," in *Proc. ISPCS*, An Arbor, MI, USA, Sep. 2008, pp. 16–22.

[5] J. Ridoux and D. Veitch, "The cost of variability," in *Proc. ISPCS*, Ann Arbor, MI, USA, 2008, pp. 29–32.

[6] *IEEE Standard for Ethernet*, IEEE Standard 802.3-2015 (Revision of IEEE Standard 802.3-2012), Mar. 2016, pp. 1–4017.

[7] *User Datagram Protocol*, document RFC 768, Aug. 1980. Accessed: Mar. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc768.txt

[8] *Internet Protocol*, document RFC 791, Sep. 1981. Accessed: Mar. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc791.txt

[9] R. Greenstreet and A. Zepeda, "Improving IEEE 1588 synchronization accuracy in 1000BASE-T systems," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun. (ISPCS)*, Beijing, China, Oct. 2015, pp. 58–63.

[10] Y. Li, B. Noseworthy, J. Laird, T. Winters, and T. Carlin, "A study of precision of hardware time stamping packet traces," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun. (ISPCS)*, Austin, TX, USA, Sep. 2014, pp. 102–107.

[11] B. Noseworthy, "Direct measurement of ingress and egress latency on 1000Base-T Gigabit Ethernet links," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun. (ISPCS)*, Austin, TX, USA, Sep. 2014, pp. 53–58.

[12] J. Han and D. Jeong, "Practical considerations in the design and implementation of time synchronization systems using IEEE 1588," *IEEE Commun. Mag.*, vol. 47, no. 11, pp. 164–170, Nov. 2009.

[13] N. Kerö, T. Kernen, T. Müller, and M. Schimandl, "Optimizing large media networks for highly accurate time transfer via PTP," *SMPTE Motion Imag. J.*, vol. 125, no. 2, pp. 30–44, Mar. 2016.

[14] N. Kerö, T. Müller, T. Kernen, and M. Deniaud, "Analysis of precision time protocol (PTP) locking time on non-PTP networks for generator locking over IP," *SMPTE Motion Imag. J.*, vol. 123, no. 2, pp. 37–47, Mar. 2014.

[15] N. Kerö, T. Kernen, and T. Müller, "SMPTE conference—PTP deployment in large networks—Traps and Pitfalls," in *Proc. Annu. Tech. Conf., Exhib. (SMPTE)*, 2014.

[16] N. Kerö, T. Kernen, T. Müller, and M. Deniaud, "Analysis of lock times of PTP slave under varying network load conditions using class based queuing," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control Commun. (ISPCS)*, Lemgo, Germany, Sep. 2013, pp. 18–23.

[17] D. L. Mills, "Internet time synchronization: The network time protocol," *IEEE Trans. Commun.*, vol. 39, no. 10, pp. 1482–1493, Oct. 1991.

[18] *Network Time Protocol (NTP)*, document RFC 958, Sep. 1985. Mar. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc958.txt

[19] D. Mills, U. Delaware, J. Martin, Ed., J. Burbank, and W. Kasch, *Network Time Protocol Version 4: Protocol and Algorithms Specification*. Accessed: Mar. 2018. [Online]. Available: https://tools.ietf.org/html/rfc5905

[20] X. Jie, X. Liang, D. Lian, and Z. Delin, "Research on network timing system based on NTP," in *Proc. 13th IEEE Int. Conf. Electron. Meas. Instrum. (ICEMI)*, Oct. 2017, pp. 356–360.

[21] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard 1588-2002, 2002, pp. 1–144.

[22] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard 1588-2008 (Rev. IEEE Standard 1588-2002), Jul. 2008, pp. 1–300.

[23] J. Ferrant, *Synchronous Ethernet and IEEE 1588 in Telecoms: Next Generation Synchronization Networks*. Hoboken, NJ, USA: Wiley, Jun. 2013.

[24] W. Owczarek, "Deploying PTP as an enterprise service: Issues, challenges and design considerations," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control Commun. (ISPCS)*, Lemgo, Germany, Sep. 2013, pp. 77–82.

[25] N. Ciarleglio, T. Edwards, and R. Welch, "Large-scale PTP: How big can it get?" *SMPTE Motion Imag. J.*, vol. 126, no. 4, pp. 1–7, May/Jun. 2017.

[26] H. Liu, J. Liu, T. Bi, J. Li, W. Yang, and D. Zhang, "Performance analysis of time synchronization precision of PTP in smart substations," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun. (ISPCS)*, Beijing, China, Oct. 2015, pp. 37–42.

[27] G. S. Antonova *et al.*, "Standard profile for use of IEEE Standard 1588-2008 precision time protocol (PTP) in power system applications," in *Proc. 66th Annu. Conf. Protective Relay Eng.*, College Station, TX, USA, 2013, pp. 322–336.

[28] D. Macii, D. Fontanelli, and D. Petri, "A master-slave synchronization model for enhanced servo clock design," in *Proc. Int. Symp. Precis. Clock Synchronization Meas., Control Commun.*, 2009, pp. 1–6.

[29] P. Tosato, D. Macii, D. Fontanelli, D. Brunelli, and D. Laverty, "A software-based low-jitter servo clock for inexpensive phasor measurement units," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun. (ISPCS)*, Geneva, Sep./Oct. 2018, pp. 1–6.

[30] R. Exel and F. Ring, "Improved clock synchronization accuracy through optimized servo parametrization," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control Commun. (ISPCS)*, Lemgo, Germany, Sep. 2013, pp. 65–70.

[31] C. Andrich, J. Bauer, P. Große, A. Ihlow, and G. D. Galdo, "A fast and stable time locked loop for network time synchronization with parallel FLL and PLL," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun. (ISPCS)*, Geneva, Switzerland, Sep. 2018, pp. 1–4.

[32] G. Giorgi and C. Narduzzi, "On the accuracy of packet-based measurements in the PTP telecom profile," in *Proc. IEEE Int. Workshop Meas., Netw. (M&N)*, Coimbra, Portugal, Oct. 2015, pp. 1–6.

[33] I. Hadzic and D. Morgan, "Adaptive packet selection for clock recovery," in *Proc. IPSCS*, 2010, pp. 41–47.

[34] F. Ring, T. Bigler, and R. Exel, "Synchronization robustness using Kalman-based clock servos," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun. (ISPCS)*, Beijing, China, Oct. 2015, pp. 64–69.

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

[36] *IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks*, IEEE Standard 802.1Q-2014 (Revision of IEEE Standard 802.1Q-2011), Dec. 2014, pp.1-1832.

[37] *IEEE Standard for Local and Metropolitan Area Networks—Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*, IEEE Standard 802.1AS-2011, pp. 1–292, Mar. 2011.

[38] B. W. Pike, "I R I G, inter-range instrumentation group—History, Functions and Status, 1959," *IRE Trans. Space Electron. Telemetry*, vol. 6, no. 1, pp. 59–61, Mar. 1960.

[39] [Online]. Available: https://apps.dtic.mil/dtic/tr/fulltext/u2/1013738.pdf

[40] Y. Zeng and G. Chen, "Research on methods to improve precise time synchronization for IRIG-B code encoder," in *Proc. Asia–Pacific Power Energy Eng. Conf.*, Shanghai, China, 2012, pp. 1–4.

[41] J. R. Razo-Hernandez, M. Valtierra-Rodriguez, D. G. Lieberman, J. P. Amezquita-Sanchez, L. A. Morales-Hernandez, and A. Dominguez-Gonzalez, "IRIG-B decoder based on FPGA for synchronization in PMUs by considering different input formats," in *Proc. IEEE Int. Autumn Meeting Power, Electron. Comput. (ROPEC)*, Ixtapa, Mexico, Nov. 2016, pp. 1–6.

[42] J. R. Razo-Hernandez, M. Valtierra-Rodriguez, D. G. Lieberman, J. P. Amezquita-Sanchez, L. A. Morales-Hernandez, and A. Dominguez-Gonzalez, "IRIG-B decoder based on FPGA for synchronization in PMUs by considering different input formats," in *Proc. IEEE Int. Autumn Meeting Power, Electron. Comput. (ROPEC)*, Ixtapa, Mexico, Nov. 2016, pp. 1–6.

[43] [Online]. Available: https://www.gps.gov/applications/timing/

[44] [Online]. Available: https://www.gsa.europa.eu/european-gnss/galileo/galileo-european-global-satellite-based-navigation-system

[45] M. Luo, Y. Gong, and J. Wang, "Design of a GNSS signal synchronization chip," in *Proc. 14th IEEE Int. Conf. Solid-State Integr. Circuit Technol. (ICSICT)*, Qingdao, China, Oct. 2018, pp. 1–3.

[46] K. F. Hasan, Y. Feng, and Y.-C. Tian, "GNSS time synchronization in vehicular ad-hoc networks: Benefits and feasibility," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 12, pp. 3915–3924, Mar. 2018.

[47] M. A. Enright and C. N. Kurby, "A signals of opportunity based cooperative navigation network," in *Proc. IEEE Nat. Aerosp., Electron. Conf. (NAECON)*, Dayton, OH, USA, Jul. 2009, pp. 213–218.

[48] T. Ren and M. G. Petovello, "A stand-alone approach for high-sensitivity gnss receivers in signal-challenged environment," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 53, no. 5, pp. 2438–2448, Oct. 2017.

[49] S.-H. Kong, "High sensitivity and fast acquisition signal processing techniques for GNSS receivers: From fundamentals to state-of-the-art GNSS acquisition technologies," *IEEE Signal Process Mag.*, vol. 34, no. 5, pp. 59–71, Sep. 2017.

[50] M. A. Enright and C. N. Kurby, "A signals of opportunity based cooperative navigation network," in *Proc. IEEE Nat. Aerosp., Electron. Conf. (NAECON)*, Dayton, OH, USA, Jul. 2009, pp. 213–218.

[51] N. Kerö, T. Kernen, and T. Müller, "Efficient monitoring of ST2059-2 based time transfer performance," *SMPTE Motion Imag. J.*, vol. 126, no. 4, pp. 1–8, May/Jun. 2017.

[52] A. Shpiner, Y. Revah, and T. Mizrahi, "Multi-path time protocols," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control Commun. (ISPCS)*, Lemgo, Germany, Sep. 2013, pp. 1–6.

[53] P. Ferrari, A. Flammini, S. Rinaldi, and G. Prytz, "High availability IEEE 1588 nodes over IEEE 802.1 AQ shortest path bridging networks," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas.*, Sep. 2013, pp. 35–40.

[54] S. Rinaldi, P. Ferrari, A. Flammini, D. Fontanelli, and D. Macii, "Improving syntonization of synchronous Ethernet nodes using multiple paths," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun. (ISPCS)*, Austin, TX, USA, Sep. 2014, pp. 1–6.

[55] G. Gaderer, R. Höller, T. Sauter, and H. Muhr, "Extending IEEE 1588 to fault tolerant clock synchronization," in *Proc. IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, Wien, Austria, Sep. 2004, pp. 353–357.

[56] U. Schmid, M. Horauer, and N. Ker, "How to distribute GPS-time over COTS-based LANs," in *Proc. 31th IEEE Precise Time Time Interval Syst. Appl. Meeting (PTTI)*, Dana Point, CA, USA, 1999, pp. 545–560.

[57] U. Schmid and K. Schossmaier, "Interval-based clock synchronization with optimal precision," *Inf. Comput.*, vol. 186, no. 1, pp. 36–77, Oct. 2003.

[58] A. Puhm, A. Mahmood, T. Bigler, and N. Kerö, "Synchronizing an IEEE 1588 slave clock over both paths of a redundant Ethernet system," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun. (ISPCS)*, Stockholm, Sweden, Sep. 2016, pp. 1–6.

[59] G. Gaderer, A. Nagy, P. Loschmidt, and N. Kerö,

"A novel, high resolution oscillator model for DES systems," in *Proc. IEEE Int. Freq. Control Symp.*, Honolulu, HI, USA, May 2008, pp. 178–183.
[60] *IEEE Standard for Local and Metropolitan Area Networks—Common Specification. Part 3: Media Access Control (MAC) Bridges—Amendment 2: Rapid Reconfiguration*, document IEEE Standard 802.1w-2001, 2001.
[61] G. Prytz, "Redundancy in industrial Ethernet networks," in *Proc. IEEE Int. Workshop Factory Commun. Syst.*, Torino, Italy, Jun. 2006, pp. 380–385.
[62] F. von Tüllenburg and T. Pfeiffenberger, "Layer-2 failure recovery methods in critical communication networks," presented at the 12fth Int. Conf. Netw. Services, (ICNS), Lisbon, Portugal, 2016, pp. 66–71.
[63] *IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges*, IEEE Standard 802.1D-2004 (Revision of IEEE Standard 802.1D-1998), Jun. 2004, pp. 1–281.
[64] *Communication Networks and Systems for Power Utility Automation—Part 5: Communication Requirements for Functions and Device Models*. Standard IEC 61850-5, 2013.
[65] *IEEE Standard for Local and Metropolitan Area Networks-Frame Replication and Elimination for Reliability*, IEEE Standard 802.1CB-2017, 2017.
[66] *664P7-1 Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network*, Standard ARINC 664p7, 2009.
[67] *Industrial Communication Networks—High Availability Automation Networks—Part 3: Parallel Redundancy Protocol (PRP) and High-Availability Seamless Redundancy (HSR)*, Standard IEC 62439-3, 2016.
[68] *IEC/IEEE International Standard—Communication Networks and Systems for Power Utility Automation Part 9-3: Precision Time Protocol Profile for Power Utility Automation*, IEC/IEEE Standard 61850-9-3, 2016.
[69] *IEEE Standard Profile for Use of IEEE 1588 Precision Time Protocol in Power System Applications*, IEEE Standard C37.238, 2017.
[70] M. Adamiak *et al.*, "Revision of IEEE Std C37.238, power profile for IEEE-1588: Why the big changes?" in *Proc. Georgia Tech Protective Relaying Conf.*, College Station, TX, USA, 2018.
[71] T. Koskiahde and J. Kujala, "PTP monitoring in redundant network," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun. (ISPCS)*, Stockholm, Sweden, Sep. 2016, pp. 1–5.
[72] T. Murakami, Y. Horiuchi, and K. Nishimura, "A packet filtering mechanism with a packet delay distribution estimation function for IEEE 1588 time synchronization in a congested network," in *Proc. ISPCS*, Sep. 2011, pp. 114–119.
[73] M. Rentschler and H. Heine, "The parallel redundancy protocol for industrial IP networks," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Cape Town, South Africa, Feb. 2013, pp. 1404–1409.
[74] M. Popovic, M. Mohiuddin, D.-C. Tomozei, and J.-Y. L. Boudec, "iPRP—The parallel redundancy protocol for IP networks: Protocol design and operation," *IEEE Trans. Ind. Informat.*, vol. 12, no. 5, pp. 1842–1854, Oct. 2016.
[75] M. Rentschler and P. Laukemann, "Towards a reliable parallel redundant WLAN black channel," in *Proc. 9th IEEE Int. Workshop Factory Commun. Syst.*, Lemgo, Germany, May 2012, pp. 255–264.
[76] G. Cena, S. Scanzio, and A. Valenzano, "Seamless link-level redundancy to improve reliability of industrial Wi-Fi networks," *IEEE Trans. Ind. Informat.*, vol. 12, no. 2, pp. 608–620, Apr. 2016.
[77] G. Cena, S. Scanzio, and A. Valenzano, "Improving effectiveness of seamless redundancy in real industrial Wi-Fi networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2095–2107, May 2018.
[78] M. Rentschler and P. Laukemann, "Performance analysis of parallel redundant WLAN," in *Proc. IEEE 17th Int. Conf. Emerg. Technol., Factory Autom. (ETFA)*, Krakow, Poland, Sep. 2012, pp. 1–8.
[79] G. Cena, S. Scanzio, and A. Valenzano, "Experimental evaluation of seamless redundancy applied to industrial Wi-Fi networks," *IEEE Trans. Ind. Informat.*, vol. 13, no. 2, pp. 856–865, Apr. 2017.
[80] A. Mahmood, R. Exel, H. Trsek, and T. Sauter, "Clock synchronization over IEEE 802.11—A survey of methodologies and protocols," *IEEE Trans. Ind. Informat.*, vol. 13, no. 2, pp. 907–922, Apr. 2017.
[81] T. Murakami, Y. Horiuchi, and K. Nishimura, "A packet filtering mechanism with a packet delay distribution estimation function for IEEE 1588 time synchronization in a congested network," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun.*, Munich, Germany, Sep. 2011, pp. 114–119.

ABOUT THE AUTHORS

Nikolaus Kerö received the Diploma degree (Hons.) in communication engineering from the Vienna University of Technology, Vienna, Austria.

He then led the ASIC Design Division, Institute of Industrial Electronics, Vienna University of Technology, successfully managing numerous research projects and industry collaborations. In 2001, he cofounded Oregano Systems Design & Consulting Ltd., Vienna, as a university spin-off. He has coauthored more than 100 publications in various fields of application-specific integrated circuit and embedded system design as well as clock synchronization aspects. He holds eight patents. His current research interests include distributed systems design, especially highly accurate and fault-tolerant clock synchronization.

Mr. Kerö is an active member of the IEEE1588 Standardization Committee and the respective SMPTE 32NF standards group and conducts frequent seminars on clock synchronization for both industry and academia.

Thomas Kernen is currently a Staff Architect with Mellanox Technologies Ltd., Yokneam, Israel. Prior to joining Mellanox, he spent over 20 years in the IP industry, including driving Cisco's entry into live media production, co-founding Internet Service Providers, Telecom carriers, and architecting Fiber to the Home networks. He has authored over 20 publications in leading journals. He holds six patents that cover both network and video coding optimizations for media transport and delivery. His current research interests include defining architectures for transforming the broadcast industry to an All-IP infrastructure.

Dr. Kernen is also a member of the IEEE Communications and Broadcast Societies. He currently serves as the Co-Chair of the Society of Motion Pictures and Television Engineers (SMPTE) 32NF Committee. He has served for seven years as an Editor of the Digital Video Broadcasting (DVB) for the TS 101 154 Specification for the Use of Video and Audio Coding in Broadcasting Applications Based on the MPEG-2 Transport Stream supported by millions of digital receivers worldwide. He is also a frequent speaker at leading events, such as the SMPTE Annual Technical Conference, the National Association of Broadcasters, IBC, and the European Broadcasting Union Network Technology Seminar.

Andreas Puhm received the Diploma degree in electrical engineering and the M.S. degree in embedded systems from the University of Applied Sciences Technikum Wien, Vienna, Austria, in 2006 and 2008, respectively.

He has been lecturing on digital system design and distributed systems at the University of Applied Sciences Technikum Wien, where he has been a member of the Research Group Embedded Systems, Department for Electronic Engineering, since 2006. His current research interests include embedded computing, dependable VLSI systems, system-on-chip design, real-time networks, and redundant clock synchronization.

Anton Mroczkowski received the B.Sc. and M.Sc. degrees in embedded systems design from the University of Applied Sciences Technikum Wien, Vienna, Austria, in 2014 and 2018, respectively.

In 2014, he joined Oregano Systems, Vienna, Austria, as an Embedded Systems Designer. Since 2017, he has been heading the Software Team, Oregano Systems, overseeing the continuous development of the PTP software stack as well as various research activities in the area of fault-tolerant clock synchronization. His current research interests include highly reliable clock synchronization with a special emphasis on embedded systems with limited resources for automotive and telecom applications.