Hochschule Offenburg
offenburg.university

ivESK Institut für verlässliche Embedded Systems und Kommunikationselektronik

Offenburg University of Applied Sciences

Institute of reliable Embedded Systems and Communication Electronics (ivESK)

Time Synchronization Based on IEEE Std 802.1AS – 2020

-

General overview & comparison with
IEEE Std 802.1AS – 2011

Project:	ControlTSN
Version:	0.1
Date:	25.08.2021

Authors

Kamil Alkhouri [kamil.alkhouri@hs-offenburg.de]

Axel Sikora [axel.sikora@hs-offenburg.de]

History

Version	*Author(s)*	*Date*	*Summary of changes*
0.1	K. Alkhouri	25.08.2021	Initial release

Abbreviations

Abbreviation	*Description*
ivESK	Institute of reliable Embedded Systems and Communication Electronics
TSN	Time Sensitive Network
PTP	Precision Time Protocol
gPTP	generalized Precision Time Protocol
GM	Grandmaster
PEI	PTP End Instance
PRI	PTP Relay Instance
EPON	Ethernet Passive Optical Network
WLAN	Wireless Local Area Network
PEI	PTP End Instance
PLL	Phase-Locked Loop
BMCA	Best Master Clock Algorithm
CMLDS	Common Mean Link Delay Service
TLV	Time-Length-Value

PPS	Pulse Per Second
YANG	Yet Another Next Generation
GPS	Global Positioning System

Table of Contents

List of Figures

Executive Summary

This report provides a sufficient explanation of time synchronization protocol based on the IEEE Std 802.1AS standard with special focus on the main differences between the two versions of the standard released in 2011 and 2020. Chapter 1 introduces the need for time synchronization, while chapter 2 briefly explains the concepts of clock synchronization and syntonization. The establishment of synchronization hierarchy is described in chapter 3, followed by the technical details of the synchronization process in chapter 4. Chapter 5 focuses on the new aspects defined in 2020 and makes useful comparisons with the previous version of the year 2011. In the end, the discussion is concluded in chapter 6.

1 Introduction

Since the development of modern industry, human intervention in factory processes has been gradually decreasing while sophisticated control systems took over the mission of administrating and supervising different machineries in the industrial domain. In order to function efficiently, industrial networks pose very strict requirements on their communication systems in terms of determinism and reliability of delivery which can only be achieved using real-time communications.

Aiming to deal with this issue, some proprietary communication protocols were invented. Although many of these protocols are commonly used nowadays (like Profinet, EtherCAT, SERCOS III, etc.), they are often not compatible with each other so they cannot coexist on the same physical network, not to mention the lack of compatibility with IEEE standards. This need led to the development of Time Sensitive Network (TSN) which can be described is simple words as an Ethernet-based technology that promises a standardized solution to extend the existing IEEE Std 802.3 Ethernet to support real-time requirements and make it useful for industrial use-cases [1].

The development of TSN standards is not yet finished but it is safe to say that time synchronization is considered as an essential prerequisite for any TSN implementation. To clarify this statement, the following case is presented.

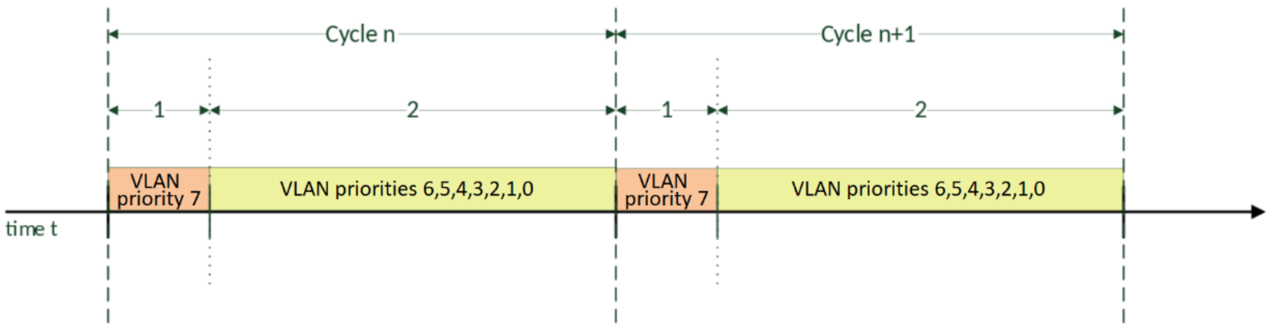

Fig. 1: Example of a time-aware schedule [2]

Fig. 1 shows an example of a time-aware schedule made on the basis of IEEE Std 802.1Qbv standard (one of the main elements of TSN). According to this standard, eight transmission priority gates are defined for each outbound port on each device in the network. These gates open and close according to a pre-calculated schedule for the purpose of creating periodic congestion-free time slots dedicated for the transmission of real-time traffic (VLAN priority 7 in this example). For this to work, the timing of the schedules needs to be the same all over the network. Otherwise, bottlenecks can be formed in some segments leading to delays and/or blockages of critical traffic which is completely unacceptable in TSN. Therefore, the gate operations, and consequently time slots, need to be precisely synchronized on all TSN devices within a network. To accomplish that, time synchronization is a crucial requirement for the underlying clocks on which the gate operations are based [3].

TSN standardization community adopted IEEE Std 802.1AS as the official time synchronization standard. It is called generalized Precision Time Protocol (gPTP), because it is based on the Precision Time Protocol (PTP) defined in the IEEE Std 1588. This document will not discuss PTP, rather it will focus on gPTP as a standalone standard.

2 Clock Synchronization and Syntonization

2.1 Synchronization Model and Devices

gPTP time synchronization model consists of a grandmaster (GM) which has the best clock or connected to the best external clock and acts as the reference of time. It sends its synchronized timing information messages to the directly connected devices. Each of these devices receives the synchronized time, processes it and forwards (or regenerates) it according to some specific working principles explained later in details. As a result, GM reference time travels through the consecutive devices until it reaches the destination endpoints (slaves) in which it is used to adjust their local time to be equal to the time of the grandmaster [4]. The following Figure below (Figure 2) provides an example of a gPTP network which typically consists of two types of devices:

- PTP End Instance (PEI): acts as the source of timing information if it is selected to be the grandmaster, or the destination of timing information if it is selected to be a slave
- PTP Relay Instance (PRI): intermediate bridging device which receives the timing information from the previous device, processes and updates it, then sends it to the following devices in the network.

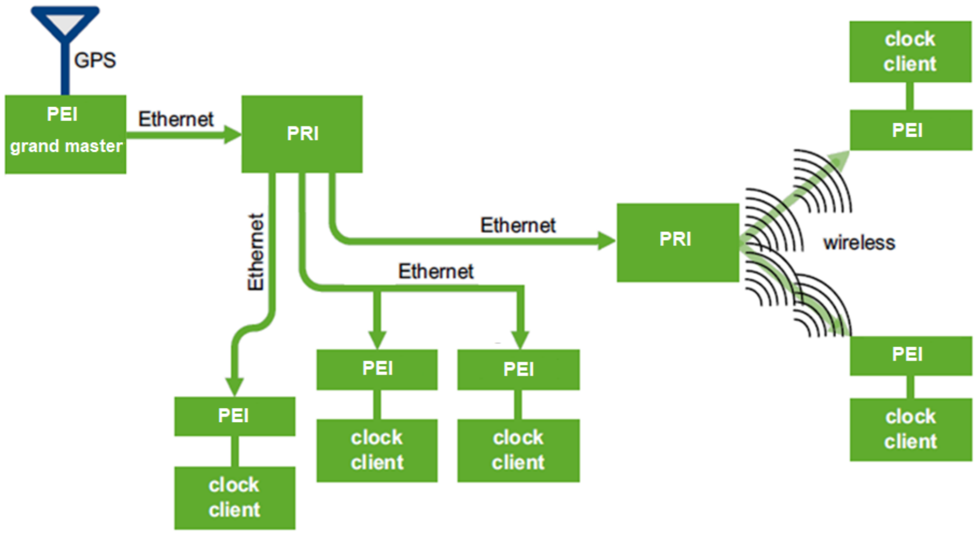

Fig. 2: Example of a gPTP synchronization network model [4]

Fig. 2 also shows that one grandmaster device can be responsible for synchronizing many devices even when the media connecting them is not Ethernet. In fact, the standard defines different ways for synchronization information to travel from Ethernet to other media connections like Ethernet Passive Optical Network (EPON) and Wireless Local Area Network (WLAN) without problems [4]. However, since this document is mainly concerned with Ethernet TSN, only time synchronization over regular Ethernet is covered.

2.2 Clock Syntonization

Since each of the devices along the way from the grandmaster to the endpoints has a specific role in the synchronization process, it makes sense to assume that the concept of time on these devices needs to be similar to the one of the grandmaster. In other words, the duration of one second (consequently, the frequency of the clock) on all devices communicating the synchronized information should be identical. The process of coordinating the frequencies of two or more clocks is called clock syntonization. It can be performed in two different ways [4]:

- Physical syntonization: the actual frequency of the local oscillator is changed. This can be achieved using a Phase-Locked Loop (PLL) for example. However, this method is known to be slow and prone to gain peaking effects.
- Logical syntonization: the frequency of the local oscillator is not adjusted. Rather, the ratio of the frequency of the grandmaster clock to the frequency of the local clock is continuously updated and considered in the time calculations to achieve the same time perspective on all devices.

Although the standard allows physical syntonization, it basically uses logical syntonization in which the local clock of the gPTP device is a free-running clock. Separately, the frequency ratio of the local clock of each device with respect to the grandmaster is maintained in a variable called gmRateRatio and used to calculate the accurate synchronized time which is maintained in variables called masterTime and clockSlaveTime. Later chapters provide more details about this statement [4].

3 Establishment of Synchronization Tree

3.1 Overview

As mentioned earlier, timing information travels from the grandmaster to the rest of the devices in hierarchal manner. In a stable situation, the gPTP domain is logically structured in the form of a time-synchronization spanning tree with only one active grandmaster acting as the root and containing one or more master ports. The grandmaster can be followed by multiport PRI devices each with a single slave port connected to the path toward the grandmaster and one or many master and passive ports connected to the remaining branches of the tree. These branches can also be connected to other multiport PRIs and/or to final PEIs at the leafs of the tree. Each leaf PEI has only one port and it is in the slave state. Overall, each port in the synchronization tree shall have one of the states: MasterPort, SlavePort, PassivePort or it can be disabled either by the management or if it is not capable of fulfilling gPTP standard requirements. Fig. 3 shows a diagram a gPTP synchronization tree, where “M” stands for MasterPort, “S” for SlavePort and “P” for Passive Port [4].

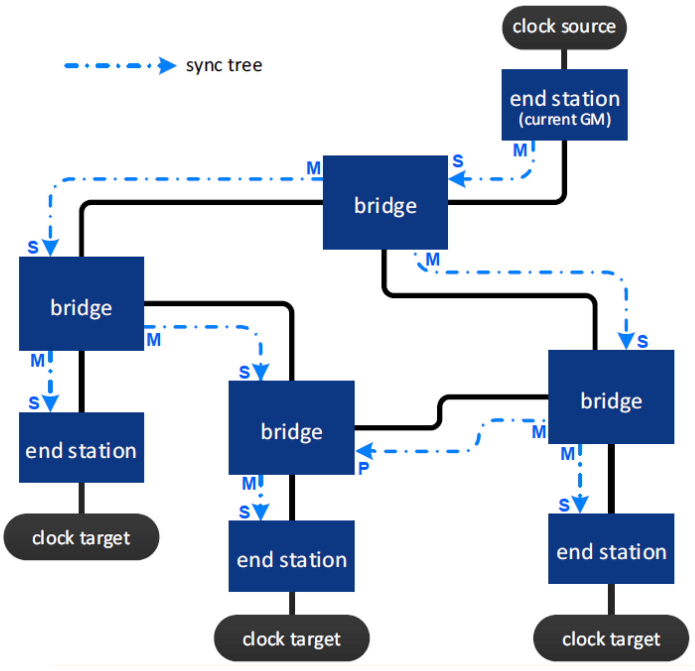

Fig. 3: gPTP synchronization tree [4]

The decision on how the synchronization tree is built and the state of each port in it is taken by an algorithm called Best Master Clock Algorithm (BMCA) which runs locally on every device in the gPTP network. The algorithm decides which port gets which state by comparing different sets of information describing the local and remote ports and choosing a suitable state for each of them based on its characteristics and location in the tree with respect to the grandmaster. Information about remote ports is fed to the BMCA using a special type of messages called **Announce** messages. These messages are periodically originated from the grandmaster then updated and forwarded by each PRI throughout the network [4].

3.2 Best Master Clock Algorithm

Each of the information sets used by BMCA consists of the following elements in the same order from the most significant to the least significant: priority1, clockClass, clockAccuracy, offsetScaled-LogVariance, priority2, grandmaster clockIdentity, steps-Removed, source portIdentity and destination portNumber. The values of these elements are either manually initialized according to standardized rules like priority1 and clockClass, or filled with runtime information like stepsRemoved which represents the number of communication paths between the local clock and the current grandmaster. For all the previous components, the smaller numerical value represents a better attribute and since they are already ordered in a form of priority vectors, comparing them is a straightforward process from a mathematical point of view. The following priority vectors are defined [4]:

- portPriorityVector (pPV) – per port: holds the information of the port in a stable situation (when the reception of Announce messages and any pending update has been completed)
- messagePriorityVector (mPV) – per port: conveyed in the received Announce messages
    - If mPV < pPV then pPV is replaced by mPV
- gmPathPriorityVector (gmPPV) – per port: derived from pPV and holds the information of the grandmaster as seen from the local port
- systemPriorityVector (sPV) – per system: holds information about the local system and used to build mPV in Announce messages transmitted from the system's master ports if the system was selected to be the grandmaster
- gmPriorityVector (gmPV) – per system: the best selected priority vector among the local information (sPV) and all received information (gmPPVs)
- masterPriorityVector (maPV) – per port: derived from gmPV and used to build mPV in the Announce messages transmitted by the port if it is selected to be a master

- if maPV < pPV then the port is selected to be a master port

The BMCA mechanism can be explained by the following simplified example of a synchronization tree establishment scenario. The example is also graphically represented in Fig. 4.

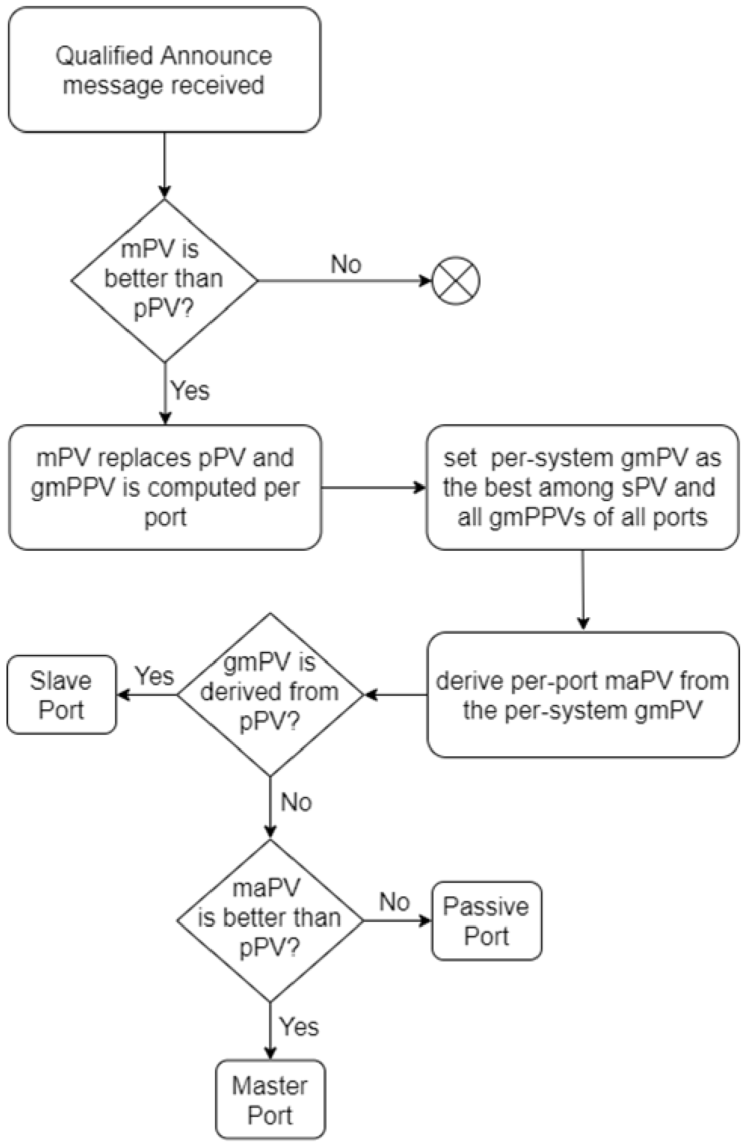
Qualified Announce message received
mPV is better than pPV?
No
Yes
mPV replaces pPV and gmPPV is computed per port
set per-system gmPV as the best among sPV and all gmPPVs of all ports
derive per-port maPV from the per-system gmPV
gmPV is derived from pPV?
Yes
Slave Port
No
maPV is better than pPV?
No
Passive Port
Yes
Master Port

Fig. 4: Working principle of BMCA (simplified)

From an operational point of view, when a gPTP time-aware system is turned on, sPV is initialized and its value is assigned to gmPV, this means, the system initially considers itself as a grandmaster. Consequently, all the ports that are “AS capable” are assigned the MasterPort state. A port is considered “AS capable” if it runs the IEEE Std 802.1AS mechanism for calculating the propagation delay from its neighboring peer port (explained later as peer-to-peer delay mechanism) and successfully calculates a propagation delay period below a predefined threshold [4].

Once an Announce message arrives at any of the gPTP ports, it is qualified. This means, the message is only considered if [4]:

- It is not sent by the local system
- It is less than 255 steps away from the grandmaster
- It is not circulating in a loop

After the Announce message is qualified, its mPV is compared to pPV of the port and if it is better, it replaces it, which leads to the update of gmPPV of the port. Afterwards, gmPV is recomputed on the system level based on the new information received from this port and possibly similar updates on other ports. At this point, one of the following three possible scenarios can take place [4]:

- If gmPV is derived from sPV, the local instance is considered to have the best master clock. Hence, it is selected to be the grandmaster and all its ports are assigned the MasterPort state.
- If gmPV is derived from pPV of the considered port, this indicates that the information received on this port is the best among the remaining information available from the local system and the other local ports. As a result, this port is assigned the SlavePort state.
- If gmPV is not derived from sPV or pPV of the considered port, this means that there is a better slave port in the system. Therefore, the port can either be master or passive. In order to decide, pPV is compared to the maPV of the port and if it is better, the port at the other end of the link is considered as a better master, as a result, the local port is assigned the PassivePort state. Otherwise, the port is given the MasterPort state because the Announce message it can transmit is better than the Announce message it can receive from the peer port.

Finally, if no Announce messages are received until the announce receipt timeout timer expires, the port is directly assigned the MasterPort state [4].

4 Synchronization Process

4.1 Overview

After building the synchronization tree and defining the most suitable paths for the synchronization information to propagate from the grandmaster to the end stations, the process of transporting this information can start. This process consists of two parts running continuously in parallel:

- Calculating the propagation delay from the neighboring peer port (peer delay mechanism)
- Forwarding the synchronization information

Following the approach of chapter 3, this chapter describes these two parts by giving a simplified example of the synchronization process. Prior to that, a brief explanation of the types of used messages is provided.

4.2 Types of Synchronization Messages

In order to transport synchronization information, gPTP standard defines two types of messages [4]:

- Event messages: used for capturing different times and measuring delays in the network by calculating the time differences between their transmission and reception moments. In order to do this, these messages are timestamped once they leave the transmitting port (Egress timestamp) and once they arrive at the receiving port (Ingress timestamp). With adequate hardware support, these timestamps can be captured at a point very close to the physical layer of the port, and therefore, they can be very accurate. These messages are: Sync, Pdelay_Req and Pdelay_Resp.

- General messages: their exact transmission and reception times are not important. Therefore, they are not timestamped. Rather, they are used to convey important synchronization information like timestamps of the event messages appended to them. These messages are: Follow_Up and Pdelay_Resp_Follow_Up.

  Note: general messages can also be used to communicate other gPTP relevant information like clock characteristics. Thus, the Announce messages described earlier are considered as general messages as well.

4.3 Propagation Delay Measurement

Each non-disabled or non-blocked port in the gPTP domain needs to measure the propagation delay encountered by the gPTP messages arriving from its neighboring peer port. This can be done using the peer-to-peer delay mechanism [4].

Fig. 5 describes the process of peer-to-peer delay mechanism where the initiator port starts by issuing a ***Pdelay_Req*** message and generating its egress timestamp t_1. The responder port receives the message and generates the ingress timestamp t_2. Directly afterwards, the responder sends a ***Pdelay_Resp*** message, which carries t_2, and generates timestamp t_3. Later on, once this message is received by the initiator, timestamp t_4 is generated and the initiator waits for the ***Pdelay_Resp_Follow_Up*** message which finally arrives from the responder carrying t_3 [4].

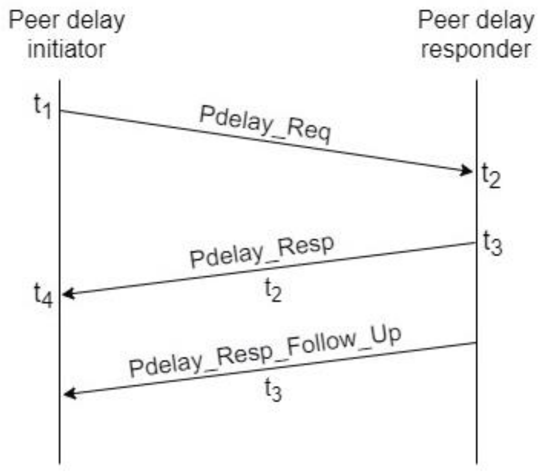

Fig. 5: Peer-to-peer delay mechanism [4]

At this point, the initiator becomes aware of all the captured timestamps, so it uses them to calculate the mean propagation peer delay (D) based on Eq. 1 [4].

$$D = \frac{r\,.\,(t_4 - t_1) - (t_3 - t_2)}{2} \qquad \textbf{Eq. 1}$$

Where r represents the neighborRateRatio which is the ratio of the frequency of the clock used by the responder to the frequency of the clock used by the initiator. It is estimated by observing successive values of t_3 and t_4 taken from successive Pdelay_Resp and Pdelay_Resp_Follow_Up messages. The fact that the initiator multiplies r by the timestamps captured on its side (t_1 and t_4)

indicates that D is actually calculated relative to the time base of the responder, or in other words, to the clock of the peer port [4].

By looking at Eq. 1, the reader can easily see that D represents a mean value because it assumes that delay is the same in both directions of propagation. However, in reality, this might not be the case all the time. If the propagation link is asymmetric the value of D will include an amount of error. In order to deal with such a situation, the standard introduces an optional delayAsymmetry factor that can be considered to improve the precision of D. Although it is left to the implementation to decide how to obtain delayAsymmetry, the 2020 version of the standard explicitly mentions that it cannot be calculated during the live operation of the system. Therefore, gPTP operations are held if delayAsymmetry measurement process is taking place (i.e. asymmetry measurement mode is enabled). Furthermore, the standard advises to measure delayAsymmetry before running the system or to obtain it separately from the supplier [4].

Another factor to be considered here, that is not shown in Eq. 1, is the time measurement granularity which completely depends on the used clock and can be in the range of 40 ns. This effect can occur while calculating any of the four timestamps and every time it occurs it causes 20 ns change in the calculated value of D. In order to reduce this effect and improve accuracy, the standard proposes to apply an averaging filter on the measured successive values of D [4].

4.4 Transport of Synchronization Information

The source of time is usually an external entity like Global Positioning System (GPS) connected to the grandmaster via an application interface. The slaves on the other hand can be connected to external applications that use the synchronized time via an application interface as well. The standard does not discuss the external entities (clock source and slave applications) but it provides some examples for the application interfaces connecting them to allow better modelling and clearer abstraction for the gPTP process itself [4].

Fig. 6 shows a simplified example of a gPTP time synchronization scenario where the tree built using BMCA consists of one grandmaster PEI and one slave PEI communicating through one PRI in the middle. At the beginning, the synchronization process is initiated on the grandmaster PEI which continuously updates its local time (masterTime) to the reference time obtained from the external source. Another important variable also maintained here is the gmRateRatio which –at this point– represents the frequency ratio of the external clock source to the local clock of the grandmaster PEI.

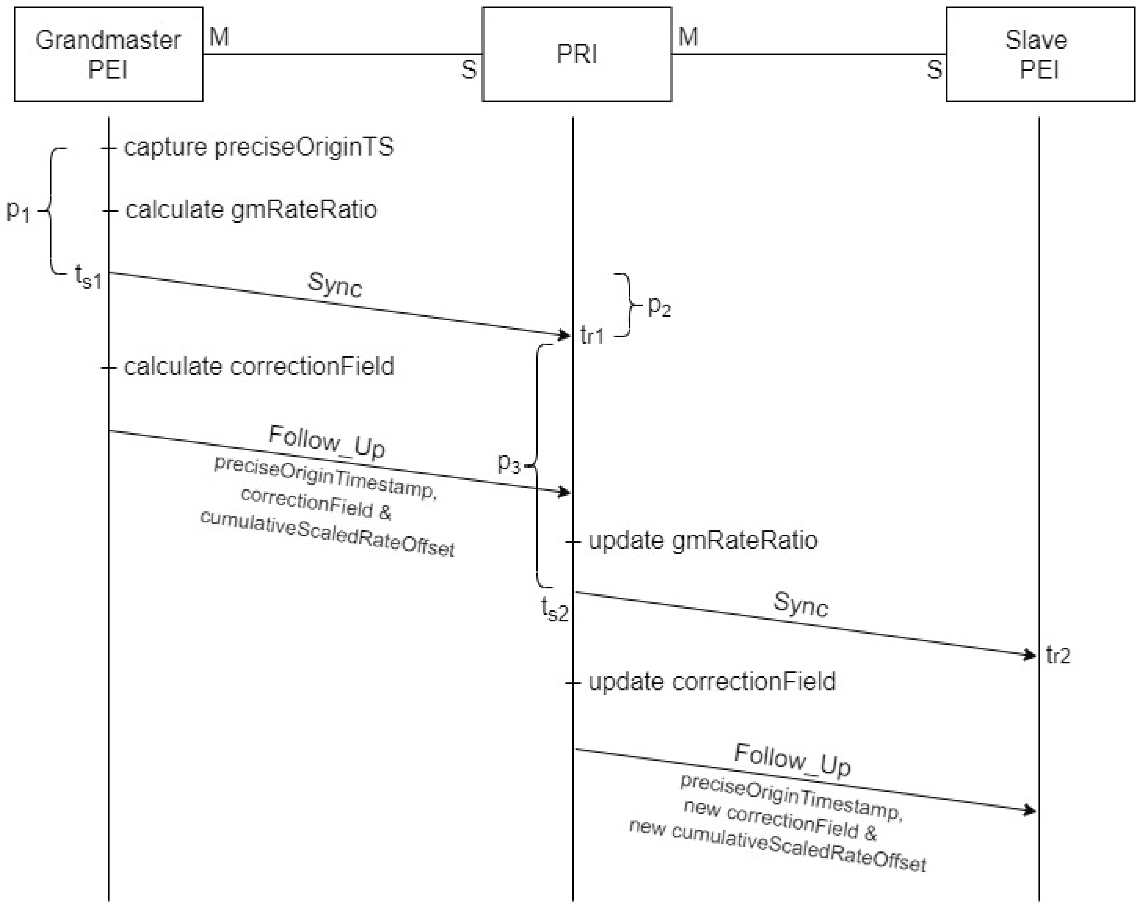
Grandmaster PEI
M
S
PRI
M
S
Slave PEI
capture preciseOriginTS
p1
calculate gmRateRatio
ts1
Sync
p2
tr1
calculate correctionField
Follow_Up
preciseOriginTimestamp, correctionField & cumulativeScaledRateOffset
p3
update gmRateRatio
ts2
Sync
tr2
update correctionField
Follow_Up
preciseOriginTimestamp, new correctionField & new cumulativeScaledRateOffset

Fig. 6: Transport of synchronization information

The grandmaster PEI periodically generates synchronization information and forwards them to the PRI in the form of ***Sync*** and possibly ***Follow_Up*** messages (see 5.10). The purpose of the Sync message is to capture highly accurate t_{s1} and t_{r1} timestamps, while the Follow_Up message is used to carry the following information:

- preciseOriginTimestamp: the accurate masterTime at which the Sync message is internally originated
- correctionField: contains the fractional nanoseconds value of the preciseOriginTimestamp, in addition to the period of time the Sync message spent within the grandmaster PEI until it was transmitted at t_{s1} (appears in the figure as p_1). This period is multiplied by gmRateRatio to represent it relative to the clock of grandmaster's external source (see 2.2).
- cummulativeScaledRateOffset: a scaled value of the gmRateRatio
- Information about the last phase and frequency changes of the external clock source

Upon the reception of the Sync/Follow_Up pair and capturing t_{r1}, PRI calculates the upstreamTxTime which represents the timestamp t_{s1} relative to the local clock of PRI using Eq. 2 [4].

$$upstreamTxTime = t_{r1} - \frac{D}{r} \left\{ - \frac{delayAsymmetry}{gmRateRatio} \right\} \qquad \textbf{Eq. 2}$$

Where D and r are obtained from Eq. 1, and the part within the curly brackets only applies when the peer-to-peer delay is calculated based on the instance-specific method rather than CMLDS (see 5.4). gmRateRatio here is obtained from the cummulativeScaledRateOffset of the received Follow_Up message.

<u>Note:</u> Since the mean propagation peer delay is actually calculated relative to the time base of the neighbor, dividing it by r here converts it to the time base of the current local clock. In the same way, the value of delayAsymmetry is also converted from time base of the grandmaster to the time base of the current local clock by dividing it by gmRateRatio [4].

The PRI then updates the value of gmRateRatio by accumulating the neighborRateRatio (r) to it. As a result, it becomes equal to the frequency ratio of the grandmaster clock to the local clock of the PRI. Afterwards, the following two tasks are performed in parallel:

- Task1: updating and forwarding the Sync/Follow_Up pair via the master ports
- Task2: using the received information for internal synchronization

In task1, the PRI updates the following fields of the Follow_Up message, then forwards it [4]:

- correctionField: contains the sum of the received correctionField and the time period the Sync message spent between t_{s1} and t_{s2} (appears in the figure as p_2 and p_3) expressed relative to the grandmaster clock (see 2.2). The mathematical formula for calculating the new correctionField is shown in Eq. 3.

$$\begin{aligned} new\ correctionField = \ & correctionField + \\ & gmRateRatio * (t_{s2} - upstreamTxTime) \end{aligned} \qquad \textbf{Eq. 3}$$

- cummulativeScaledRateOffset: recalculated based on the new gmRateRatio

In task2, the PRI uses the received preciseOriginTimestamp and correctionField along with the new gmRateRatio to calculate the reception time of the Sync message relative to the grandmaster clock (Eq. 4) and relative to its own local clock (Eq. 5).

$$syncReceiptTime = preciseOriginTimestamp + correctionField + D * \frac{gmRateRatio}{r} + delayAsymmetry \qquad \textbf{\textit{Eq. 4}}$$

$$syncReceiptLocalTime = upstreamTxTime + \frac{D}{r} + \frac{delayAsymmetry}{gmRateRatio} \qquad \textbf{Eq. 5}$$

Based on these two variables, the PRI can calculate the offset of its local clock from the clock of the grandmaster and update its synchronized time maintained in a variable called clockSlaveTime. The update mechanism of this variable is not strictly defined in the standard. However, a valid example is to set it to SyncReceiptTime whenever this value changes and increment it at every local clock tick by the interval of the local clock multiplied by gmRateRatio (see 2.2) [4].

Eventually, the slave PEI receives the forwarded Sync/Follow_Up pair from the PRI and rather than forwarding these messages any further, it preforms the same synchronization procedures performed by the PRI in task2 where the ClockSlaveTime is continuously updated to follow the grandmaster clock [4].

4.5 Signaling Messages

Signaling messages are general messages used to carry information, requests and/or commands using the Time-Length-Value (TLV) structure. The ***message interval request TLV*** is an example of these messages. It is used by a port to request its peer port to change the desired transmission interval for sending synchronization information, propagation delay measurements or Announce messages. This message is also used to indicate whether the peer port should compute neighborRateRatio and mean propagation delay, or whether the local port can handle one-step Sync messages (see 5.10) [4].

5 New Concepts Introduced in 2020

5.1 External Port Configuration

The standard allows an external entity to determine the grandmaster and build the synchronization tree instead of using BMCA. The algorithm used by the external entity is not defined in the standard. However, the same Announce messages used by BMCA are utilized by the external entity to transport synchronization tree information and grandmaster time properties from one PTP instance to the other [4].

5.2 Signaling gPTP Capability

The standard defines the signaling ***gPTP-capable TLV*** which is continuously transmitted from each port as an indicator that it is capable of running gPTP [4].

5.3 Multi-Domain Support

The standard defines the outlines for running multiple gPTP instances on multiple domains. The use-cases behind this feature are mainly industrial where some applications can have special requirement like multiple timescales and/or extra redundancy features [4]. Fig. 7 is an updated version of Fig. 3 where two gPTP domains are configured in the network.

According to the figure, the physical network is divided into two logical networks that overlap in some devices and links. Some ports can be masters in one domain and slaves in the other. Based on this assumption, a separate independent instance of BMCA must be invoked in each domain, so each domain must have its own grandmaster device and must build its own synchronization tree. Additionally, synchronization information travels through different paths in different domains. This indicates that the mechanism used to transport synchronization information is per domain.

In Fig. 7, the same grandmaster is selected by both domains and the redundancy is on the path level, while Fig. 8 shows an example of a multi-domain gPTP network with two grandmasters. It is clearly seen from this figure that the red (standby) grandmaster is at the same time a slave to the blue (primary) grandmaster. This scenario gives the advantage of having a hot standby solution where, if the primary grandmaster failed, the standby grandmaster can take over the synchronization process immediately without having to wait for the BMCA in the primary tree to exchange the updated Announce messages and elect a new primary grandmaster [4].

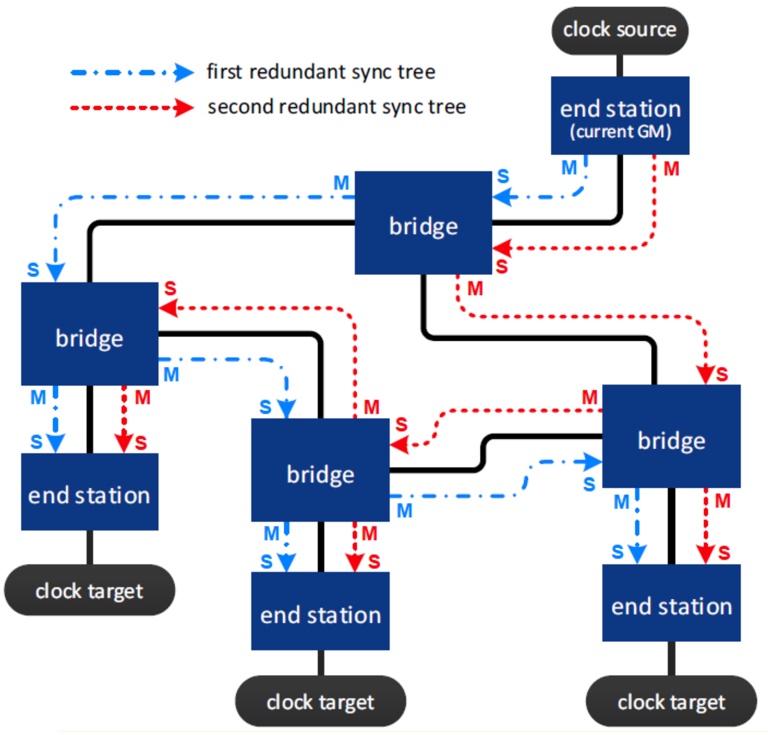

Fig. 7: Multi-domain gPTP synchronization tree [4]

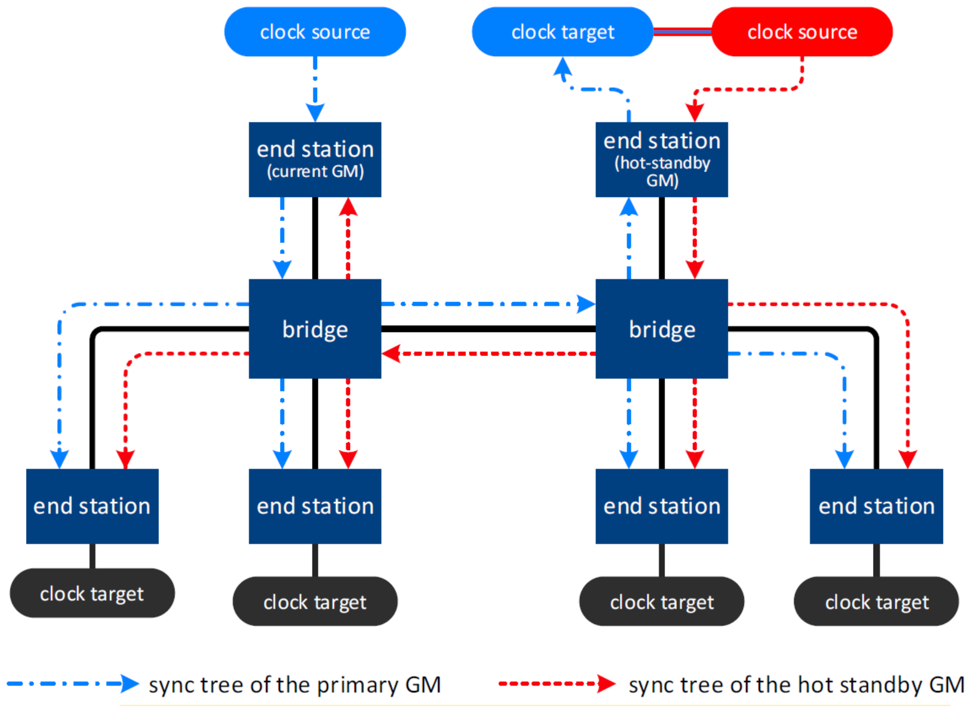

Fig. 8: Multi-domain gPTP synchronization tree with two GMs [4]

Note: some gPTP ports can be disabled on one domain and enabled on another.

Note: every gPTP system shall have support for domain 0 to maintain a backward compatibility with the 2011 version of the standard which allows only one domain (domain 0) to be running. Nevertheless, this domain does not have to be the active one in a multi-domain scenario [4].

5.4 Common Mean Link Delay Service (CMLDS)

The introduction of multi-domain approach requires special support for the peer-to-peer delay mechanism. Since the mean delay on a PTP link is the same in all domains, there is no need to invoke this mechanism for each domain. A single instance of it is enough to do the required measurements and make them available for all active domains. The way to do this is through the CMLDS support which is defined in 2020 version of the standard in addition to the instance-specific peer-to-peer delay mechanism of 2011 version (to maintain backward compatibility). Although both methods follow the same approach described in 4.3, there are some structural differences between them. For example, in CMLDS the delayAsymmetry is considered within the correctionField of the Pdelay_Resp_Follow_Up message used to correct t_3 in Eq. 1, so it is modelled in the equation, while this is not the case with instance-specific peer-to-peer delay where delayAsymmetry is not stored in any correctionField. Hence, it needs to be considered separately when upstreamTxTime is calculated later (see Eq. 2). Furthermore, when neighborRateRatio is calculated using CMLDS, its value needs to be corrected for delayAsymmetry (for the same reason), while this correction is not required when instance-specific peer-to-peer delay is used [4].

5.5 asCapableAcrossDomains

The capability of the port to operate the IEEE 802.1AS protocol is modeled in both versions of the standard as asCapable. This variable is set to true if the port is able to measure a peak-to-peak synchronization jitter within 1 microsecond from a port that is separated by six or fewer PRIs. Additionally, 2011 standard defined the following conditions [5]:

- Port is able to perform peer delay mechanism
- Port measures a peer delay below a pre-defined threshold (800 ns)
- Port does not receive multiple responses to a single request
- Port does not receive a response from the itself or from other ports of the same device

In 2020 version, with the introduction of multi-domain concept, the fulfillment of the previous conditions in one domain sets asCapableAcrossDomains to true (because the links connecting peer ports are identical in all domains), while asCapable of a port is set to true when [4]:

- asCapableAcrossDomains is set to true
- gPTP-capable TLV is received from the peer port, indicating that it is able to operate the IEEE 802.1AS protocol

5.6 gPTP Domain Identification

A domain is identified by two attributes: domain number and sdoId. Domain number can be any value in the range 0 through 127, while sdoId of a domain is a 12-bit structure that consists of [4]:

- majorSdoId (4 bits): set to 0x1 for the gPTP domain and instance-specific peer-to-peer delay mechanism, and set to 0x2 for the CMLDS
- minorSdoId (8 bits): always set to 0x0

majorSdoId was named transportSpecific field of 2011 standard messages while minorSdoId did not exist. Domain number was always set to 0 because only one gPTP domain was allowed [5].

5.7 Announce Slow Down

Although the old version of gPTP also introduced the message interval request TLV to allow a port to request a change in the rate at which Announce messages are transmitted from its peer master port (announce interval), it did not specify how the ports handle the switch-over from one interval to another [5]. However, this mechanism is strictly defined in the new version of the standard. According to the new definition, the transmitting port informs the receiving port about the expected new rate while it continues to send Announce messages at the same old rate for a period of time equivalent to the old announce receipt timeout. This ensures a smooth timeout-free transition from a high rate to a slower one [4].

5.8 Sync Slow Down

This process is completely identical to the announce slow down process (see 5.7) where the message interval request TLV is used to change the rate at which Sync messages are transmitted from master port (sync interval). Again, the new aspect here is the way of handling the switch-over from one interval to another. According to the new definition, the transmitting port informs the receiving port about the expected new rate while it continues to send Sync messages at the same old rate for a period of time equivalent to the old sync receipt timeout. This ensures a smooth timeout-free transition from a high rate to a slower one [4].

5.9 SyncLocked Feature

As mentioned earlier, the PRI receives the Sync/Follow_Up messages through its slave port and forwards them through its master ports. The new SyncLocked variable decides when these messages shall be forwarded. If SyncLocked is TRUE, the PRI forwards the messages as soon as possible after receiving them, while if SyncLocked is FALSE, each master port in the PRI forwards the messages based on its own sync interval which is completely independent from the time of reception [4]. In the old version of the standard, the master port of the PRI forwards the messages as soon as possible after receiving them, provided that they do not arrive sooner than a period of time equals to half of its sync interval. Otherwise, the messages are not forwarded [5].

Note: readers with background knowledge in IEEE Std 1588 can identify that the case where SyncLocked is TRUE reflects the behavior of a PTP transparent clock, while a FALSE SyncLocked reflects the behavior of a PTP boundary clock.

5.10 One-Step Processing

Both versions of the standard define the two-step processing method on which an event message is followed by a follow_up message that carries its Tx timestamp and other useful information (Follow_Up and Pdelay_Resp_Follow_Up messages). However, the 2020 version allows the optional usage of one-step processing for Sync message. This means, Tx timestamp of a Sync message can be embedded “on the fly” in the message itself instead of using a Follow_Up message to deliver it to the receiver. This option nevertheless is not available for peer-to-peer delay mechanism [4].

6 Conclusion

This document is intended to cover the main aspects of IEEE Std 802.1AS and to provide the reader with a simplified entry point into the official time synchronization standards as a part of the ongoing efforts to implement different solutions for TSN networks. It is noteworthy to mention that at the time of writing this document, different implementations of PTP have been already available in the market. Some of these implementations introduce additional settings to activate the gPTP profile (like “linuxptp-3.0” [6]), but a complete gPTP solution where the PRI functionality and multi-domain features are supported is still missing.

References

[1] Alkhouri, K. (2018). *Architectural Design, Implementation and Verification of a Bare-metal TSN Ethernet Protocol* (thesis).

[2] *Time-Sensitive Networking*. Wikipedia. (last update: 23 June 2021). https://en.wikipedia.org/wiki/Time-Sensitive_Networking.

[3] "IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic," in *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)* , vol., no., pp.1-57, 18 March 2016, doi: 10.1109/IEEESTD.2016.8613095.

[4] "IEEE Standard for Local and Metropolitan Area Networks--Timing and Synchronization for Time-Sensitive Applications," in IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011) , vol., no., pp.1-421, 19 June 2020, doi: 10.1109/IEEESTD.2020.9121845.

[5] "IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks," in IEEE Std 802.1AS-2011 , vol., no., pp.1-292, 30 March 2011, doi: 10.1109/IEEESTD.2011.5741898.

[6] Cochran, R. linuxptp-3.1.1. *The Linux PTP Project*. http://linuxptp.sourceforge.net/.