Towards robust network synchronization with IEEE 802.1AS

Quentin Bailleul
IRT Saint Exupéry
Toulouse, France
quentin.bailleul@irt-saintexupery.com

Katia Jaffrès-Runser, Jean-Luc Scharbarg
IRIT, Université de Toulouse, Toulouse INP
Toulouse, France
{katia.jaffres-runser, jean-luc.scharbarg}
@irit.fr

Philippe Cuenot
Continental
Toulouse, France
philippe.cuenot@continental-corporation.com

***Abstract*—IEEE 802.1AS is the synchronization protocol associated to the novel real-time switched Ethernet solution called Time Sensitive Networking (TSN). We discuss and provide first insights on the design choices required for a critical embedded network context. More specifically, our aim is to extract the key features that offer both a target synchronization precision and a robust setting to mitigate the loss of synchronization.**

I. Introduction

Fieldbuses cannot cope with the increasing communication needs of current embedded systems that aim at supporting a diverse set of flows, some of them carrying video and necessitating thus a large bandwidth. As such, real-time Ethernet solutions have been designed to bring much higher bandwidth and advanced quality of service policies. The challenge is then to guarantee that time sensitive flows can meet very strong latency constraints in a network composed of multiple Ethernet switches. Moreover, some embedded distributed applications may require precise timestamping. For all these reasons, synchronization has become a central service in real-time switched Ethernet networks. This paper investigates the IEEE 802.1AS protocol [1], proposed by the IEEE TSN working group. Synchronization is required for the implementation of the real-time TSN shapers such as the Time-Aware Shaper (TAS) or the Cyclic Queuing and Forwarding (CQF) one.

The IEEE 802.1AS protocol is a profile of the IEEE 1588 [2] synchronization standard already in use in non-critical systems. IEEE 802.1AS has been designed with the goal of reaching a precision of less than 1 microsecond in a linear network setting where a master clock and an ordinary clock are separated by 7 hops. Simulation and worst case analysis [3] but also experimental measurements [4] have assessed its performance in the absence failure. However, for a standard to be adopted in a critical network, it is necessary to study the behavior of the protocol in this case.

In this paper, we address the problem of designing a robust and precise configuration of 802.1AS. First, we underline how robustness can be enforced with this standard. Second, we analyze what impacts the quality of synchronization using simulations to guide the design of a robust and precise deployment methodology for IEEE 802.1AS.

II. IEEE 802.1AS overview

IEEE 802.1AS [1] is a profile of IEEE 1588 Precision Timing Protocol [2] for Time Sensitive Networking. Sometimes called generalized Precision Time Protocol (gPTP), this protocol is used to synchronize clocks across a network using the master slave paradigm. Each port of a time-aware system has one of the following states:

- Master : sends time synchronization information to the slave port of a time-aware system located at the other end of the physical link.
- Slave : receives time synchronization information from the master port.
- Passive : ignores time synchronization information to avoid loops in the spanning tree.

The time-aware system with all its ports in master state is called Grandmaster and it is the time source of the network. It can be synchronized by an external time source (GPS, NTP, ...) or use its internal clock.

To determine port states, IEEE 802.1AS provides two methods. The first method is the external port state configuration. This method allows to statically define the state of the ports for all the devices involved in the synchronization. The second method is the Best Master Clock Algorithm (BMCA). This distributed algorithm is executed on each time-aware system to eventually determine the state of the local ports and to elect the Grandmaster by comparing the information received from each Grandmaster candidate.

Synchronization itself is based on two core mechanisms: *i)* the distribution of synchronization information and *ii)* the measurement of the link delay using the peer-to-peer delay mechanism.

The distribution mechanism is based on the transmission of `Sync` and `Follow_Up` messages that allow each time-aware system to synchronize to the Grandmaster clock. Every synchronization interval, which is typically of 125ms, the Grandmaster sends a `Sync` message out of its master ports, followed by a `Follow_Up` message containing t_0, the exact sending time of the `Sync` message, as pictured in Fig. 1. These two messages are received via the slave ports of the equipment connected to the Grandmaster. If the receiving device has at least one port in the master state then it will forward the `Sync` message to the next time-aware system. It forwards as well the `Follow_Up` message that carries t_0 and the data $T_{prop}+T_{res}$. Here, T_{prop} is the propagation delay measured with the other peer-to-peer delay mechanism and T_{res} is the residence time of the `Sync` message in the switch. These operations are repeated at each hop, until the complete set of network equipment is reached.

The peer-to-peer delay mechanism defined in IEEE 802.1AS measures the link propagation delay. It consists in an

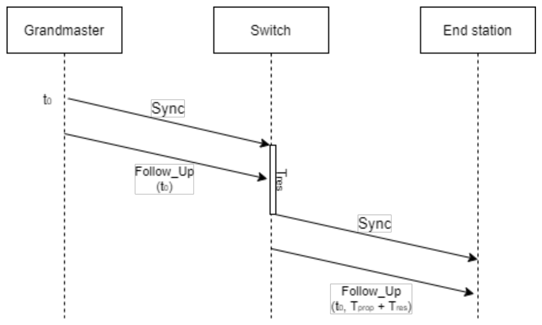

Fig. 1: Distribution mechanism.

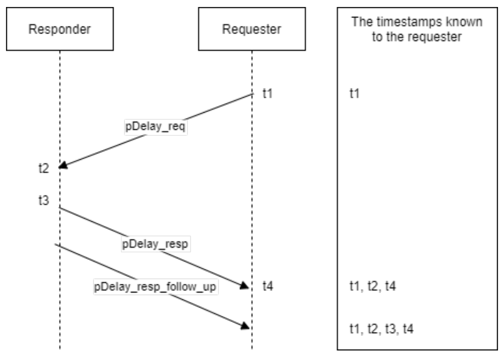

Fig. 2: Peer to peer delay mechanism.

exchange of `Pdelay` messages carrying timestamps between two time-aware systems that are separated by one hop. This mechanism is executed every *Pdelay interval* (1s by default). To measure this propagation delay, the protocol needs four timestamps as depicted in Fig. 2. The first timestamp $t1$ is measured when the `Pdelay_req` is issued. Timestamp $t2$ is obtained upon receipt of this message. Timestamp $t3$ is measured when the `Pdelay_resp` is sent. Finally, $t4$ is measured upon reception of `Pdelay_resp`. With the formula (1), the mechanism can calculate the delay of the link, T_{prop}, from the timestamps. Moreover, with the $t3$ and $t4$ timestamps of two consecutive Pdelay procedures, the requester can extract the neighbor rate ratio nr of Eq. (2) in order to compensate the relative clock drift in Eq. (1).

$$T_{prop} = \frac{(t2 - t3) + nr.(t4 - t1)}{2} \tag{1}$$

Equation (1) makes the hypothesis of the symmetry of the link but the protocol makes it possible to compensate for the existing asymmetries if they can be estimated (typically in a calibration step).

Using the two mechanisms described above, each device can adjust its local clock. Indeed, to deduce the current time, the device just sets its clock to t_0, the original time of transmission of the `Sync`, and adds the previous propagation delay and residence times accumulated in the `Follow_Up` message and the propagation delay of the last hop T_{prop} measured with the peer-to-peer delay mechanism.

$$nr = \frac{f_{req}}{f_{resp}} = \frac{t3_i - t3_{i-1}}{t4_i - t4_{i-1}} \tag{2}$$

In the 2020 version of IEEE 802.1AS [5], a domain mechanism has been added. A domain is made up of several time-aware systems participating in the synchronization including a Grandmaster. A time-aware system can belong to several domains, originating from the same Grandmaster or from different ones. In the latter case, the backup Grandmaster is called the *hot-standby* Grandmaster. It is important to note that all these domains are active together at the same time, with proper `Sync` and `Follow_Up` messages sent in the network. However, the function that determines which domain to use has not yet been standardized. Defining this function is one of the objectives of the upcoming IEEE 802.1ASdm amendment.

III. A CONFIGURATION FOR ROBUST SYNCHRONIZATION

For a critical on-board network, ensuring robustness is compulsory for the deployment of IEEE802.1AS. The BMCA offers robustness in the event of a link, time-aware system or Grandmaster failure by re-configuring the state of the ports to allow synchronization information to be broadcast again. This is a very flexible mechanism, but for which it may be difficult to predict which spanning tree will be used for the dissemination of `Sync` and `Follow_Up` messages at any time. In addition, the reconfiguration time may depend on the order of arrival of the messages describing the quality of a Grandmaster candidate and reconfiguration may take up to several seconds.

Nonetheless, the use of multiple domains combined with the external configuration of the port state makes it possible to obtain robustness under certain conditions. Indeed, it is necessary that the spanning trees used to distribute synchronization information in the different domains offer robustness to time-aware system or link failures. It is also possible to use several Grandmasters to overcome the failure of one of the Grandmasters. This solution is less dynamic than the BMCA but unlike a network which uses the BMCA, a network using the domains with the external configuration of the ports has a very low and deterministic reconfiguration time. Indeed, in the event of a loss of synchronization information, once the loss is detected, the device can decide to set its clock to the synchronization information of another domain that is already available with gPTP. This short reconfiguration time is therefore dominated by the time to detect the failure.

Despite the fact that the fault detection function is not standardized, it is reasonable to assume that the fault detection mechanism works similarly to the one of the legacy BMCA. Thus, for a default configuration of the standard, a time-aware system has to wait for the loss of 3 consecutive `Sync` messages, i.e. 3 intervals of 125ms, to detect a failure. With the *syncLocked* mode activated, the detection of the loss takes place at the same time in all the network up to the residence time. The *syncLocked* mode forces the device to forward a `Sync` via its master ports directly upon the reception of its parent `Sync` message. We can conclude that it is possible to modify the *syncInterval* and the *syncReceiptTimeout* to change the fault detection duration in order to match our robustness constraint.

Given the lower reconfiguration time in the event of a failure and the strong design preference of the industrial world for static configurations for safety reasons, we will therefore focus on the implementation of multiple domains using static port configuration.

With the static configuration, we have control over the state of each port in the network, and thus we can choose at set of domains that offer redundancy. A synchronization domain is mathematically represented by a spanning tree, rooted at the Grandmaster node. Ideally, to resist k failures, we should find a set of $k+1$ independent spanning trees. That is, the paths joining every pair of vertices x and y in any two spanning trees should be vertex disjoint. Finding a set with this property ensures that the loss of k link or time-aware system will have no impact on the distribution of synchronization information, but it is difficult to observe this property on a real topology. Indeed, let us take the example of a TSN topology reminiscent of a standard automotive use case as presented in Fig. 3. In this topology, nodes 0 to 6 and 17 are switches and others are end systems. Device 4 is the Grandmaster. Two domains are defined at most since we can only find at most two independent paths between the Grandmaster and the other time-aware systems. It is possible to observe as well that end stations are not accessible by independent paths.

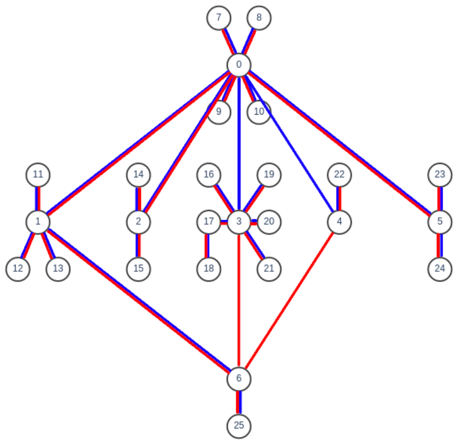

Fig. 3: A robust configuration of two domains for an automotive topology.

Ensuring the complete independence of a set of spanning trees may be intractable in real network. Thus, we are looking for the "most independent set of spanning trees". As such, we look for instance for two spanning trees where the loss of a link or of a node prevents the distribution of synchronization information for as few nodes as possible. The configuration shown in Fig. 3 is one of the possible configurations that reduces the impact of a link or device failure compared to weaker one of Fig. 4. For example, a failure of the link between 0 and 4 has no impact on the synchronisation distribution thanks to the presence of the red domain that doesn't use this link. This is not the case for the configuration of Fig. 4. This weak configuration is also more sensitive to device failures because the distribution of `Sync` and `Follow_Up` messages is centralized by a few device such as device 0.

IV. A configuration for precise synchronization

On top of creating a robust set of synchronisation domains, another challenge is to ensure the synchronisation service these

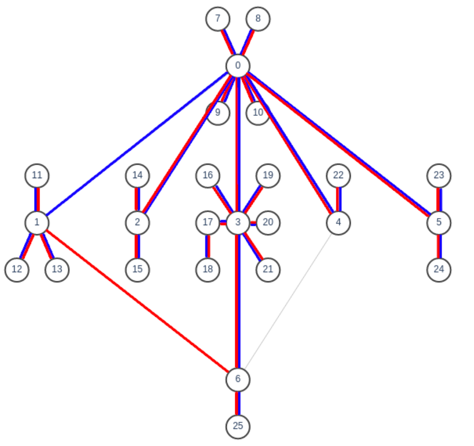

Fig. 4: A weak configuration of two domains for an automotive topology

domains offers is, at least, precise as needed by the networked system. First, let's define more explicitly the term precision. Given the reference time of the Grandmaster, we can compute for any time-aware system the difference between its local time and the reference time. This is the time deviation that can be expressed or measured for any time-aware system composing the network and at any reference time. Thus, a precise device has a small time deviation from the Grandmaster. A theoretical definition of the precision of the complete networked system is given by the maximum time deviation experienced by any given time-aware system and at any time.

A. *Source of imprecision*

The first parameter that can impact a device's precision is the quality of its clock. Indeed, any real clock tends to drift because of tiny changes in frequency of the oscillator due to factors like aging or external temperature. If the clock drift is limited to +/- 10ppm (parts per million), the clock drifts at the maximum of +/- 1.25µs compared to an ideal clock within a *syncInterval* of 125ms.

The second parameter which can affect the precision of the synchronization is the accuracy of the time-stamping necessary for the propagation and residence time measurement mechanisms. Despite the hardware time-stamping used in devices supporting IEEE 802.1AS, which eliminates software-related inaccuracies, several phenomena impact the accuracy of timestamps such as oscillator noise, jitter caused by the PHY layer and clock granularity. In the following, we are only interested in the granularity of the clock and the PHY layer jitter. Indeed these two combined phenomena can add up to 16ns to the real timestamp, compared to the noise related to the oscillator which is less than 1ns according to P. Loschmidt et al. in [6].

B. *Simulation study*

In order to study the impact of inacurate timestamps on precision, simulations with an OMNeT ++ library created by H. Puttnies et al. [7] are presented next. We have added new features to this library: *i)* the rate ratio, which allows a logical

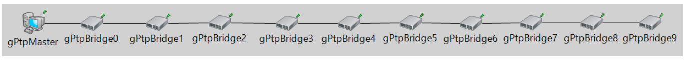

Fig. 5: The simulated daisy chain network.

syntonization with the Grandmaster, *ii)* the PHY jitter and *iii)* the clock granularity.

For our simulations, we consider a daisy chain network, as shown in Fig. 5, where 10 hops separate the Grandmaster from the most distant time-aware system. We use 5m cables between each pair of gPTP devices, enforcing a propagation delay of 25ns. Clocks have a constant -10ppm drift, a PHY jitter evenly distributed between 0 and 8ns and a granularity of 8ns. The default IEEE802.1AS parameters of *syncInterval*= 125ms and the time between `Pdelay_req` of 1 second are set. Results are extracted from 30 simulations of 1000s each. The results obtained with perfect and inaccurate timestamping mechanisms are presented in the Table I.

From these results, we observe that for the perfect timestamping scenario, the worst precision is of -1250ns regardless the distance between the device and the Grandmaster. This value is the drift that a clock of -10ppm undergoes between two `Sync` messages spaced by 125ms. This time deviation can be reduced by using better clocks of lower drift or by reducing the *syncInterval* resulting in an increase in synchronization traffic. Moving to a *syncInterval* of 31.25ms and using clocks with a drift between -5ppm and 5ppm, the time deviation can be reduced to a value ranging between -162.5 and 162.5 ns.

By adding the timestamp inaccuracies in the simulations, we observe that the precision decreases with the number of hops. In fact, at each new hop, the inaccuracies in the measurement of the propagation and residence times are added to the `Follow_Up` message which is forwarded to the next device. The inaccuracies have a reduced impact on the first hop device because only the propagation measurement is needed to adjust its clock. Since we consider the Grandmaster to be perfect, due to the superior quality of its oscillator compared to other time-aware systems, the first hop device is not impacted by the Grandmaster drift or its timestamping inaccuracies. For the last jumps, the inaccuracies that accumulate are lower because the timestamping errors compensate for each other. Approaching a worst case is therefore more difficult. The error caused by the imprecision of timestamps can be reduced by using clocks with a lower granularity, by using a filter to average the propagation delay measurements or by reducing the number of jumps. Thus, the choice of the state of the different ports of the time-aware devices, and therefore of the spanning tree, impacts the precision reachable by the synchronization because of the number of hops.

Hops	Worse precision with perfect timestamp in ns	Worse precision with inaccurate timestamps in ns
1	-1250	-1259
2	-1250	-1274
3	-1250	-1281
4	-1250	-1285
5	-1250	-1288
6	-1250	-1295
7	-1250	-1301
8	-1250	-1302
9	-1250	-1307
10	-1250	-1317

TABLE I: Simulation results

V. Conclusion

This work has investigated the design of a robust and precise synchronisation service for critical on-board networks. In terms of robustness, we advocate for the use of static independent domains. We have shown that it is difficult to ensure strict independence of domains on real topologies. However, it is possible to find spanning tree sets that reduce the number of devices affected by a failure. In terms of precision, we illustrated through simulations the fact that timestamp inaccuracies cause measurement errors which propagate in the rest of the network. It is therefore reasonable to minimize the number of hops between the Grandmaster and the different time-aware system when designing a spanning tree.

Future works will investigate other sources of imprecision of the timestamp, by validating the behavior of the simulator using experimental measurements on device supporting IEEE 802.1AS and finally by proposing a method to find the sets of spanning trees that best meet the constraints of robustness and precision of on-board networks.

Acknowledgment

The authors thank all people and industrial partners involved in the EDEN project. This work is supported by the French Research Agency (ANR) and by the partners of IRT Saint Exupéry Scientific Cooperation Foundation (FCS): Airbus Operation, Airbus Defence and Space, CNES, Continental Automotive, INPT/IRIT, ISAE-SUPAERO, ONERA, Safran Electronics and Defense, Thales Alenia Space and Thales Avionics

References

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

[2] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–269, 2008.

[3] M. Gutiérrez, W. Steiner, R. Dobrin, and S. Punnekkat, "Synchronization quality of IEEE 802.1 AS in large-scale industrial automation networks," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2017, pp. 273–282.

[4] G. M. Garner, A. Gelter, and M. J. Teener, "New simulation and test results for IEEE 802.1 AS timing performance," in *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, 2009, pp. 1–7.

[5] "IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications," *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pp. 1–421, 2020.

[6] P. Loschmidt, R. Exel, and G. Gaderer, "Highly accurate timestamping for ethernet-based clock synchronization," *Journal of Computer Networks and Communications*, vol. 2012, 2012.

[7] H. Puttnies, P. Danielis, E. Janchivnyambuu, and D. Timmermann, "A Simulation Model of IEEE 802.1 AS gPTP for Clock Synchronization in OMNeT++." in *OMNeT++*, 2018, pp. 63–72.