Hosting functional safety applications in factory networks through Time-Sensitive Networking

Sascha Gent*, Pablo Gutiérrez Peón†, Thomas Frühwirth*‡, and Dieter Etz*‡
*Austrian Competence Center for Digital Production (CDP), Seestadtstrasse 27 / 16, A-1220 Vienna, Austria
Email: sascha.gent@acdp.at
†TTTech Computertechnik AG, Schoenbrunner Strasse 7, A-1040 Vienna, Austria
Email: pablo.gutierrez-peon@tttech.com
‡Institute of Computer Engineering, TU Wien, Treitlstrasse 1-3 / 4. Floor / E191-03, A-1040 Vienna, Austria
Email: thomas.fruehwirth@tuwien.ac.at and dieter.etz@tuwien.ac.at

***Abstract*—Transmitting time-critical data like safety applications over the traditionally not real-time capable Ethernet standard (IEEE 802.1) requires modifications. Compared to devices, individually connected by signal wires, "network-attached" safety components offer more flexibility in their usage by making their data available to all factory entities instead of just one counter partner.**

openSAFETY – open-source standard for safety-related applications – enables functional safety, that can be integrated in any network with deterministic behavior. Time-Sensitive Networking (TSN) – a set of IEEE standards that can be applied to IEEE 802 networks – offers the possibility to run Ethernet traffic with real-time requirements without the traditional problems of Ethernet like frame dropping or timing issues.

This paper proofs the concept of these two technologies being a suitable combination to transmit time-critical safety data and non-priority traffic within a single Ethernet-based network. Several tests were made to show how TSN can host an openSAFETY-application in different scenarios including background traffic. The results show adequate performance of this system in terms of reliability under worst possible conditions like external broadcast traffic blocking the entire bandwidth.

***Index Terms*—TSN, openSAFETY, real-time factory network, safety application, scheduling, priority classes.**

I. Introduction

Modern industrial applications usually contain several independent machines carrying out different tasks in a collaborating manner to achieve a final goal (see Fig. 1). A typical example would be the production of a specific product or the packaging process of such including – in a simple and illustrating factory – two conveyor belts (line 1 and 2) and a robot. Line 1 transports final products to the robot while line 2 provides the packaging. These "Pick-and-Place"-applications can be found in most modern factories, no matter what kind of industry they are related to. They all have in common the need for coordination between collaborating machines, especially regarding safety functions protecting human staff. Focusing on a collaborating production line like scenario 1 shows that if one of these emergency mechanisms is activated the whole production line has to react, which therefore requires hard-wiring between all machines.

This is inflexible in case of factory re-design, re-certification and the reason for increased use of Ethernet-based safety networks. Connecting each component to the factory network is enough to enable central monitoring of all safety mechanisms. There are two big problems with this approach:

1) Vendor dependencies for standards and components
2) Timing requirements in safety networks

Using a vendor dependent ecosystem like Siemens´ PROFINET/PROFIsafe or B&R´s POWERLINK solves the problems above, but only to a certain extent.

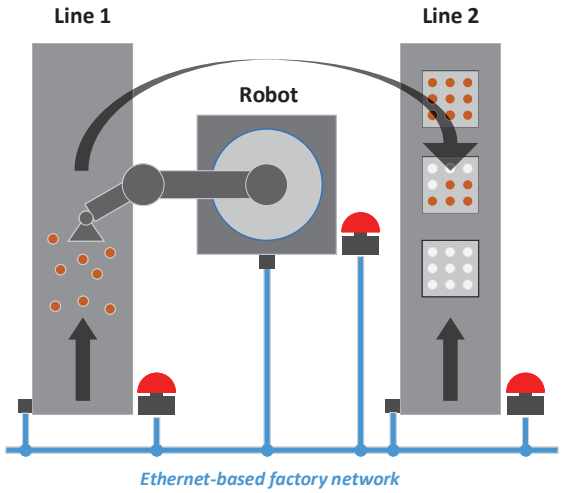

Fig. 1. Scenario 1 with network-attached safety components

Both offer safety implementations based on aged Ethernet standards with the downside of not being compatible to each other or to any other standards. The result is a rather inflexible factory network again, tightly bound to vendor´s equipment and the application it was constructed for.

This paper shows how TSN (Time-Sensitive Networking) represents a vendor independent alternative to mentioned entrenched solutions. It is a set of network standards that offers real-time capabilities for Ethernet networks to host safety applications, like those of the open-source openSAFETY-standard besides other traffic in a busy factory network without losing the real-time requirements.

II. Technical Background

This section deals with detailed information about TSN, openSAFETY and their combination to realize real-time safety-traffic.

A. openSAFETY

Functional safety in industry describes a system that ensures the absence of unacceptable risk concerning physical injury and damage to the health of people either directly or indirectly (through damage to property or the environment) [1]. In order to integrate a variety of safety features (for example an emergency stop button or light-barriers) into an industrial application without individual hard-wiring, a data network has to be established. Real-time capabilities are additionally required in case of parallel usage of this network for non-priority traffic.

openSAFETY is a bus-independent communication protocol [2] used to transmit information that is crucial for safe operation of machinery. It offers the possibility of a flexible industrial system where safety related elements can be added and removed on demand just by connecting/disconnecting them from the network and some changes in its configuration. openSAFETY uses the black channel principle; its characteristics are completely independent from the underlying transport layer. All safety-oriented mechanisms are therefore implemented on application level.

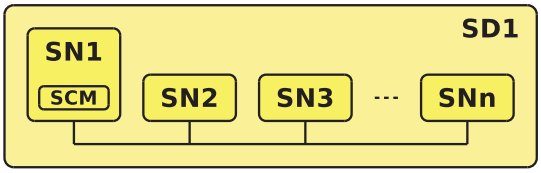

Fig. 2. Example of an openSAFETY-topology

A safety related topology realized with openSAFETY is based upon three basic building blocks:

1) **Safety Node (SN)** A SN is a device or just a single safety feature of such and identified by a unique physical address, the Unique Device Identification (UDID), and a logical address, the openSAFETY Address (SADR) which ranges from 1 to 1023. Device specific firmware, the Unique Device Identification (UDID), and the Device Vendor Information (DVI) are stored permanently on the device.
2) **Safety Domain (SD)** Each SN is part of a openSAFETY-network, referred to as safety domain (SD) [3] which consists of up to 1023 SNs.
3) **Safety Configuration Manager (SCM)** One SN within a SD has to act as SCM which provides a central configuration service. It manages the assignment and verification of the SADRs, the verification of unique UDIDs, verification of the expected parameters and DVI of the SNs, and sends a periodic life guard signal to all SNs within the SD.

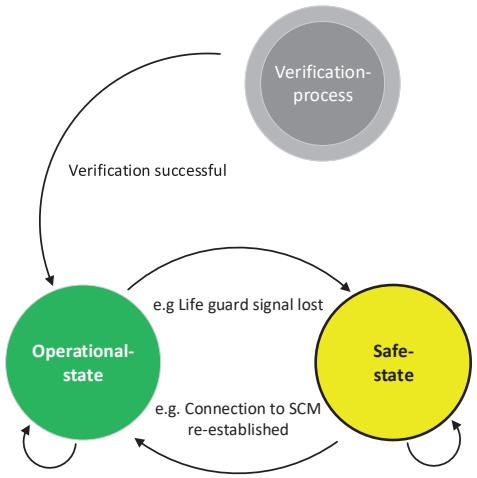

Fig. 3. Simplified state machine of a SN

Fig. 2 shows the topology of a typical openSAFETY-network. Several SNs within one SD where one specific SN acts as the SCM. Simple applications do not require more than one SD which covers the entire network. In more complex situations it is possible to introduce more SDs to split the safety network/application into task related sections – e.g. emergency stop buttons which are actively pressed and for example supervisory components like light barriers.

The application itself and corresponding parameters will be stored on the SCM only or directly in the corresponding device, depending on its capabilities. After power-on, the SCM loads its parameters from memory. Afterwards it switches to operational-state. When switching to operational-state, the SCM starts the UDID/SADR verification of all configured nodes. If the verification of a SN is successful, it will be set to operational-state.

Operational SNs get a cyclic life guard signal to keep them in operational-state (see state machine in Fig. 3, simplified to just life guard signal related state switches). Whenever this signal does not arrive in time the affected SN will switch from operational-state to safe-state. This indicates that the connection is lost – how this safe-state looks like depends on the application itself. It could be safe to cut the power-connection completely to a machine or to just activate the emergency breaks on a robot.

Referring to the setup used later in testing (see section V-A), a very basic topology with just two SNs – one acting as the SCM – transmits, besides some single frames of minor importance, 4 periodic messages between its entities: 2 request-messages from the SCM to the safe-I/Os – IDs 0x001 and 0x004 – and 2 follow-up-messages in the reverse direction – IDs 0x002 and 0x003 (see Fig. 10). This type of communication between a master and a slave is called node-polling and it describes a connection where the master always requests the data before the slave sends it back in a follow-up-message. The first pair of a request- and response-messages carry a heartbeat-signal and the second one the data itself.

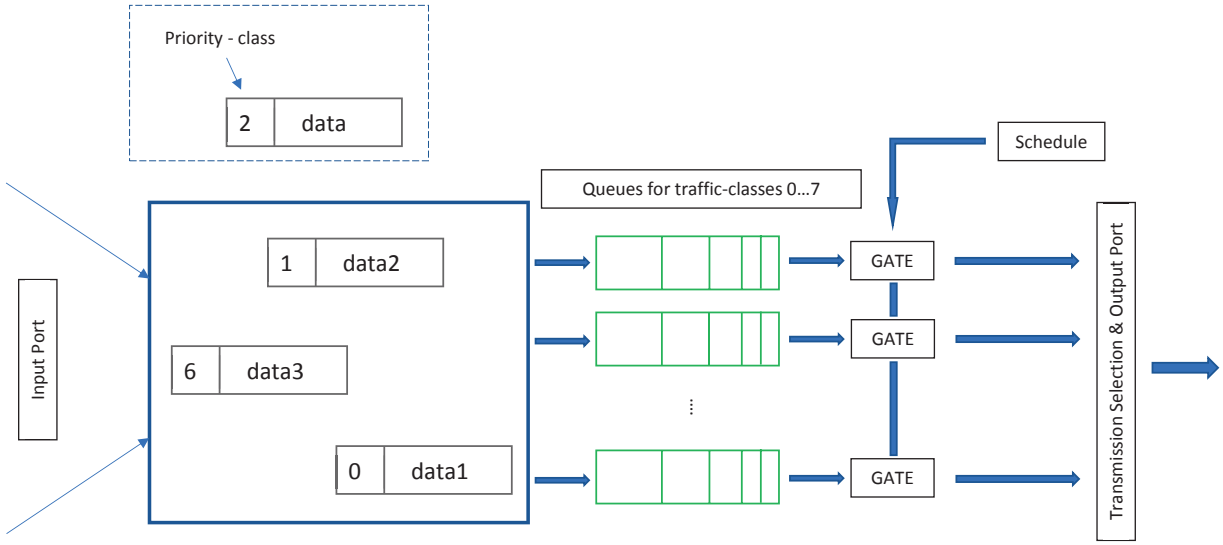

Fig. 4. Message queuing and gate mechanism in a TNS-enabled switching device with 802.1Qbv

B. TSN

TSN-networks have special requirements concerning the order in which messages and data-packages are transmitted between network components. TSN is a set of different IEEE standards [4] – each of them offering functionalities that can be applied to IEEE 802 networks, including the well-known 802.3 "Ethernet" – rather than being a single standard itself. The combination of standards may differ between applications and have to meet given requirements like time-critical communication or reliability.

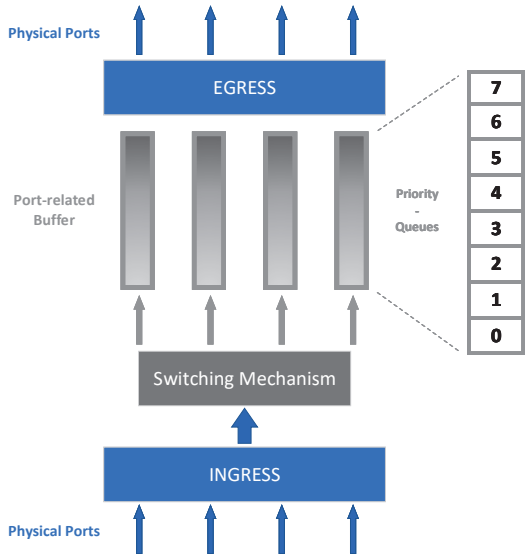

Fig. 5. Ingress and Egress of a TSN-enabled switching device

There are applications with different criticality that can be assigned to priorities in network frames. The IEEE "802.1Qbv - Enhancements for Scheduled Traffic" standard [5] is based on a queuing-system positioned between INGRESS and EGRESS of a TSN-enabled switching device (see Fig. 5). This system includes 8 queues with controllable gates on their outgoing side (see Fig. 4). In order to prioritize certain traffic, the header of theses frames has to be modified. Within the 802.1Q-Tag of Ethernet frames, 12 bits at the end of the Tag-Control-Information are reserved for the VLAN-ID [6]. According to this VLAN-ID, the package will be added to one of the 8 queues – normally a first-in-first-out system – to wait for its transmission to the next hop or to its final destination. Note that even though TSN offers other standards to achieve comparable results, Qbv is a suitable solution due to its low complexity.

Since more than 8 different VLAN-IDs are possible a manual mapping has to be done beforehand that tells the switch which ID goes to which queue and therefore represents which priority class – like for example VLAN-ID 20 assigns to queue 5 or VLAN-ID 122 goes to queue 1. Gates can be in one of two possible states: open or closed. Switching the state is based on a predefined communication schedule created by the user, which should guarantee the required time-slots and bandwidth for high-priority packages in their corresponding queues. The implementation of such a schedule is a complicated task since a lot of parameters and requirements have to be taken into account. This is especially the case with multi-hop networks where several links have to be scheduled over multiple hops. There are already scheduling solutions like [7] that configure this TSN mechanism easily. The transmission selection sits on the very end of the switch and is responsible for selecting the next frame out of one of the queues to be sent through the output-port.

It is only possible to select frames from queues with gates in open-state – defined by the schedule – for further transmission. If more than one gate is in open-state, the transmission selection mechanism has to choose the next frame to be sent according to the predefined selection process. For example, it

could always send frames from the highest priority queue first as long as it is not empty.

Scheduled traffic introduces an important topic in TSN-networks with more than one scheduled switching point: clock synchronization. If several devices within one network run time-critical and precise schedules, they have to have the same notion of time. Since this paper features just one TSN-switch in the following sections, "IEEE 802.1AS" [5], which is a standard dealing with this topic should be mentioned here but will not be discussed in detail.

Safety is the major topics of this paper, so fault tolerance of TSN should also be discussed briefly. The "Path Control and Reservation" amendment (IEEE 802.1Qca) calculates the shortest route for frames through a network with a cost-based system where the shortest route represents the one with the lowest costs. When discovering two or more paths with equal costs a copy of the data frame is sent on each of them. With proper tuning to the mechanism it is possible to implement a second optional route (e.g. next cheapest) for replicated frames permanently. The mechanisms described in the "Frame Replication and Elimination for Reliability" amendment (IEEE 802.1CB) then are responsible to discard duplicated data frames taking different paths at the switches in which their routes merge.

III. State-of-the-art

While basics about openSAFETY and TSN were discussed in previous parts, this section deals with references to projects that have a relevant connection to this paper.

The demand for an integrated safety architecture within existing factory networks is high due to its promised advantages. [8] has a similar approach to this paper and focuses on using OPC Unified Architecture (OPC UA), instead of openSAFETY, alongside with TSN – all vendor-neutral technologies – to achieve this goal. It shows how to define a safety information model for OPC UA enabling devices in future developments to "offer" their safety functions to the network and therefore make them easier accessible. While both of these papers mainly cover two different priority classes – priority or not – a third class is discussed in [9]. Besides Time-Triggered (TT) and Best-Effort (BE) communication, Audio-Video-Bridging (AVB) streams with bounded end-to-end latency are introduced. It proposes a GCL synthesis approach based on a Greedy Randomized Adaptive Search Procedure, which takes the AVB-traffic into account, such that both TT and the AVB-traffic are schedulable.

Future tasks will be dealing with the re-configuration of these safety networks since all safety-features integrated into a factory have to follow the changes of flexible adapting production lines. Both, [10] and [11], contribute their information to this topic, but with different approaches. [10] focuses on a dynamic configuration process for the IEEE 802.1Qcc standard by implementing a fully centralized model for the dynamic configuration of safety-critical systems using the Riverbed Modeler simulation framework. Therefore, especially the re-configuration of TSN in a changing network topology is discussed and how to recognize these changes as fast as possible. In contrast to that, [11] focuses on reducing expensive downtime of machines while re-configuration by presenting a method of self-configuration. It shows how the requirements for such a process look like and emphasis how it allows a convenient re-configuration of safety functions. The potential for mistakes compared to manual configuration would be reduced significantly while also offering the possibility to configure safety features directly on the machine.

IV. Hosting openSAFETY with TSN

The capabilities of TSN described in section II show its potential to be a suitable platform for openSAFETY-applications. 802.1Qbv should act as an illustration of one possible that fits the mentioned requirements. Both of its functionalities – scheduling and prioritization – enable the network to fast forward important data like the openSAFETY-messages and reserve free bandwidth for periodic scheduled traffic. This paper will focus on Qbv, but it is worth mentioning that there are various TSN-standards supporting the demands that openSAFETY has to its transportation medium. QCB [4] is a promising candidate for future integration since it describes a way to increase reliability. Frames are duplicated and sent over different paths through the network which creates a second virtual connection between two openSAFETY-endpoints. If one of these connections breaks or one of the duplicated frames gets lost there is no impact to the safety application itself because of openSAFETY´s black channel principal. In addition the TSN-frame-replication feature described in II-B adds another layer of fault tolerance to the underlying communication technology.

The topology of openSAFETY described in section II-A is a virtual one since a single physical device can contain multiple SNs - one for each safety feature for example. Therefore it does not have to be exactly the same as the physical topology defined by Ethernet. It is possible to define VLANs and configure the schedule on the devices´ port to support the logical topology openSAFETY requires.

V. Test environment

After summarizing relevant theory, this section introduces the testing setup used for a practical evaluation under realistic circumstances. It contains details about its topology, included components and all software implementations running on the hardware, including the required configuration and scheduling of the network.

A. Topology and Components

All tests were performed on a setup shown in figure 6. It contains two separate networks – 1 and 2 – and a TSN-capable switching device (MFN-100 by TTTech, blue lines) in between which acts as the network-connector. Both networks are equipped with industrial CPUs (B&R-X20-series industrial CPU (X20CP1585)). An additional B&R-X20-series-component is connected to CPU 1, which acts as the SCM and

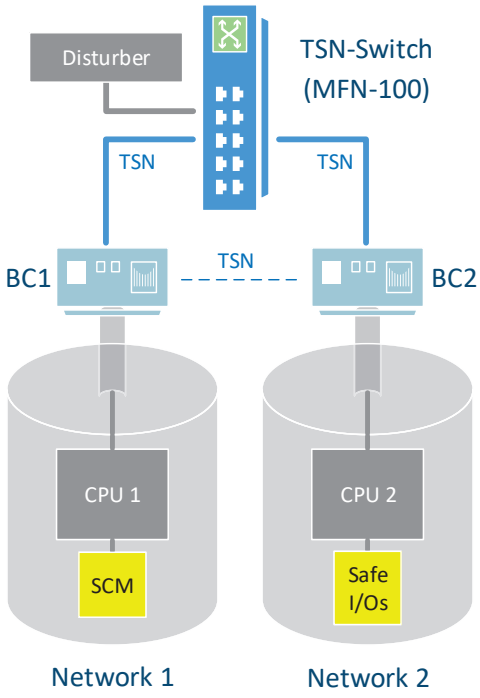

Fig. 6. Topology and components of testing-setup

B&R-X20-series-safe-I/O-ports connected to CPU 2 represent a SN constantly communication with the SCM.

To enable TSN a buscoupler (B&R X20-series buscouplers (X20BC008T-E01)) for each CPU is introduced – BC1 and BC2 – to realize the connection to the TSN-network. All connections run the Gigabit-Ethernet standard shown with grey and blue lines. The topology of the network therefore includes just one switching point, the MFN-100 where traffic has to be organized according to the TSN-standards. The schedule needed for this is stored on the MFN-Switch.

B. Test application

The hardware runs an openSAFETY-application. Its behavior and feedback helps to visualize reliability and functionality during the testing process. This includes checking the status of an emergency-stop-button via a B&R safe-input and the illustration of its value with a LED, powered by a B&R safe-output. Figure 6 shows how the safety application is split over both networks. The safe I/Os are located in the second network, while the SCM is connected to the first one. By choosing a rather short openSAFETY-cycle time of 500µs for the final test, the system is kept very sensitive to disturbance showing any transmission errors immediately. Therefore, not losing the openSAFETY-connection in this test between these two networks and their components at any point in time while stressing the network is an indicator for a successful implementation.

Testing was performed in 2 steps: basic functionality test of TSN-gate-closing mechanism and using TSN as a black channel for the openSAFETY-application running between SCM and safe I/Os – both tests will be covered in the results-section (see section VI).

C. Background traffic - ”Disturber”

The Disturber is an application running on a separate computer and is introduced into the network to flood it with data/background traffic. This application is directly connected to the TSN-switch via the computer it is running on and broadcasts data packages which is non priority traffic into the network and therefore to each networking device. The amount of data can be set to any value between 0 an the maximum bandwidth the network standard is capable of – in this case 1 Gbit/s.

D. VLAN-Tagging

Since both networks in the test-setup are not TSN-capable, the buscouplers BC1 and BC2 are used to tag and untag priority traffic coming from the networks.

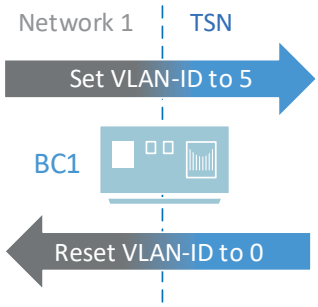

Fig. 7. VLAN-tagging and untagging of priority traffic within the buscouplers

Figure 7 shows - with BC1 as example - how each buscoupler sits at the border between two network sections: the standard Ethernet networks including the components that run the openSAFETY-application and the TSN-capable part with the TSN-switch. Each frame coming from either network 1 or 2 entering one of the buscouplers will be tagged with VLAN-ID 5 in its header which is mapped to queue 5 in the TSN-switch representing the ”priority queue” in this setup instead of queue 0 which contains standard/non-priority traffic. In reverse direction - coming from the TSN-switch into network 1 or 2 - the frames will be untagged meaning VLAN-ID will be reset to 0 so that they can be processed by the CPU. VLAN-tagged traffic would be dropped by the CPU since it is not assigned to a specific VLAN.

E. Scheduling

Focusing on the TSN-switch in this setup shows that incoming frames can have two different VLAN-IDs in their headers:

- 0 for standard/best-effort traffic or
- 5 for priority traffic

The goal of this system is to reserve bandwidth for important periodic traffic within a network that also transmits other traffic with low priority. Reserved bandwidth in terms of TSN can be translated to a specific time slot within the TSN-schedule.

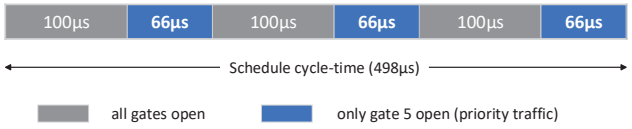

Fig. 8. TSN schedule for priority traffic

A TSN-schedule can be designed individually and perfectly tailored to the requirements of an application. Normally, a special tool takes care of creating a schedule but since the setup in this paper is that simple it was done by hand. The schedule cycle time describes the total time span in which gates can be opened and closed according to a programmed pattern before a new cycle and therefore also the pattern starts again from the beginning (see Fig. 8). The test setup runs a 498µs schedule cycle time starting with all gates open (grey) for 100µs before all of them but gate 5 (priority traffic) will be closed for 66µs (blue) to ensure all priority frames can be transmitted within their timing requirements. This pattern repeats two more times before the cycle time ends and the schedule starts again. All chosen values are based on empirical tests to maximize the time span in which all gates are open and minimize the time of traffic restrictions without losing a priority traffic frame. The chosen cycle time is kept as close as possible to the cycle time of the openSAFETY-application – which is set to 500µs – to prevent the schedule from overlapping more than one openSAFETY-cycle.

VI. Results

This section covers the detailed structure and results of both tests already mentioned in section V-B.

A. Test 1: Basic functionality test

The basic functionality test starts with CPU 1 sending single data-packages containing a timestamp and a package-ID to CPU 2 over the TSN-Switch. By creating a schedule for the gates that closes all gates at specific point in time, the loss of connection could be tested (see Fig. 9).

This figure shows the time gap between two consecutively arriving packages by subtracting the time stamp of the previous package from the latest. A total number of 5000 packages were sent. Since an interval of 2ms is set on the sending side, most of the packages also arrive in this interval on the other side. Package 95 was the last one to make it through the TSN-switch before all gates were closed which can be seen in the time gap between ID 95 and ID 96 that jumps up to 10s – the time for which the gates were closed before opened again. After reopening the packets that have queued up in the meantime are sent immediately in a burst, therefore the time gap between

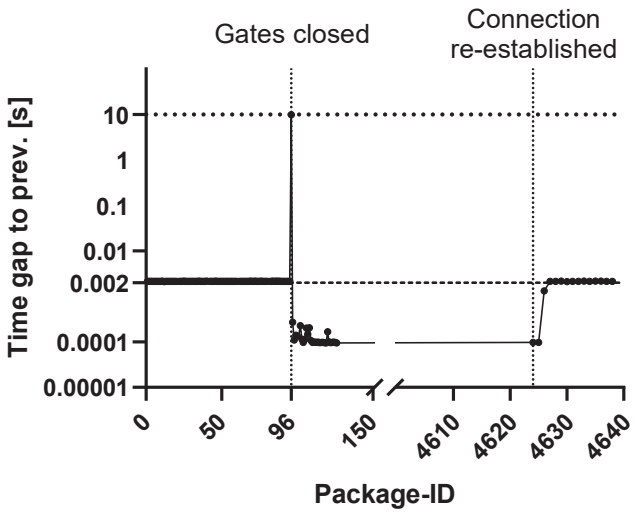

Fig. 9. Test of gate-closing mechanism

them drops significantly to around 100µs. Packages 127 to 4624 (90% of total amount) were dropped by the TSN-switch – no data points for these IDs – due to the blocked gates and filled up buffer of the queues. From package 4624 on, the process starts to normalize again and returns to the sending interval of 2ms which proofs the functionality of the TSN-gate-mechanism.

B. Test 2: TSN hosting openSAFETY

The second test – where TSN is used as black channel for openSAFETY – is split into 3 consecutive sections showing how each step impacts the systems performance and how the final setup proofs the overall concept of this paper:

1) Normal traffic without Disturber or priority traffic
2) Start of Disturber
3) Activation of TSN-schedule

At the beginning of the test both networks are connected to the TSN-switch exchanging data that is tagged and untagged by the buscouplers when entering/leaving the networks. Since there is only one SN together with the SCM in the system, 4 periodic messages which are described in section II-A and figure 10 represent the entire traffic of the network.

Figure 11 summarizes all three test stages and shows how many follow-up-messages of each type (ID 0x002 or ID 0x003 - see Fig. 10) arrive within one time interval of 50ms at the TSN-switch. These numbers are a good representation of the communication quality and if it works or not since follow-up-messages only occur after a request of the master.

The figure shows a red area below 10 messages per interval where the connection quality is too low for the strict timing requirements of the openSAFETY-application. This threshold – the minimum messages per interval or in other words the maximum time for follow-up-messages to arrive at the SCM after a request – can be set to any value and depends on the system the application is running on. Especially the

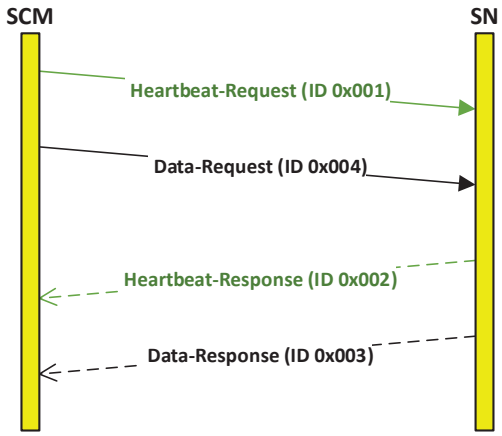

Fig. 10. Sequence of periodic openSAFETY-messages

transmission delays and processing time on each device within the communication chain which adds up to a so called total safety function response times (SFRT) [12] should be taken into consideration when choosing this time limit. The lower this time limit is, the quicker the system has to respond and the faster a failure like the loss of connection can be detected. The problem with a value too low is the more frequent occurrence of situations where a slight transmission delay leads to a complete system being set to safe-mode, even though there was no major problem, just a few frames arriving 1ms or less too late. Even though the time limit for this test was set to a rather low value of 5ms (10 messages/interval) to keep the system sensitive to disturbance and delays which gives a better picture of the TSN-performance.

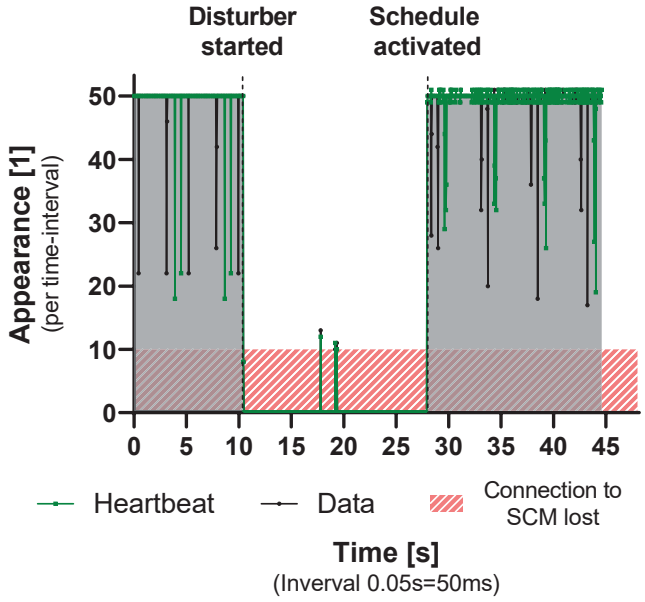

Fig. 11. openSAFETY test with Disturber and TSN-scheduling

Starting at 0s the figure shows how the operational-state looks like with constantly 50 messages of both types arriving within a 50ms interval. This equals to 1 message/ms, a value that has been expected having a cycle time of 500µs and the fact that it takes the openSAFETY-application two full cycles to send the requests and detect the reception of the follow-up-messages. At this point all messages leaving one of the networks are already tagged with VLAN-ID 5, but there is no active TSN-schedule, which means that the TSN-switch ignores this tag.

After 10s into the test the Disturber was activated and started to flood the network with broadcast-messages at maximum bandwidth. This led to queues being filled up on each network component and a delay of all messages – also those of the openSAFETY-application. The amount of arriving messages between the SCM and the safe-I/Os dropped almost to 0 into the red area which basically means that the connection broke immediately and the SN switched to safe-mode.

After 27s the TSN-schedule – mentioned in section V-E – was activated. From this point on all frames with VLAN-ID 5 queued up in queue 5 got a 66µs-time-slot every 166µs which is more than enough for all openSAFETY-messages to be transmitted in time. This change could be seen in the figure almost immediately when the value of arrived messages per interval climbed up to 50 again restoring the former connection.

During the time without Disturber and afterwards with Disturber and TSN-schedule activated the performance of the connection was way better than expected with always 18 and almost all of the time way more messages arriving at the TSN-switch per time interval. This is far away from the already in sensitivity-terms critically positioned time-limit of 10 messages/interval.

Since this paper is based on a small and simple test setup with one switch and two endpoints, a brief look on a more complex topology is appropriate. There are two different variables that define the complexity of such a network: the number of endpoints and the number of switches. To expect a big impact on the system by adding more network components seems intuitive, but openSAFETY does not care about what communication technology is used (black channel principle) as long as data is delivered before the deadline. The already mentioned scheduling-tool (see V-E) has an important role in this process. Its task is to proper allocate resources so that data arrives on time, as required by openSAFETY. The scheduler might fail doing that which could be caused by the topology but also by the algorithms used. It is possible that one tool is capable of finding a valid solution to a specific topology/traffic-configuration and another does not, but discussing this in detail would exceed the scope of this paper by far. Besides that clock synchronization stays the crucial part of a TSN-network to offer fundamental capabilities for at least up to 7 hops to work properly (see II-B).

The final test in this paper shows, that an openSAFETY-application can be run on an Ethernet network with TSN-capable devices together with other non-priority traffic without losing the timing requirements that a real-time dependent application – like a safety-related – needs.

VII. Conclusion

This paper shows how TSN can be used to run an openSAFETY-application within existing industrial ethernet-networks together with non-priority traffic.

Referring back to the scenario described at the beginning (section I), openSAFETY offers the possibility to add and remove input- and output-devices for safety-related data during runtime like the mentioned emergency-stop-button. Using an Ethernet network instead of hard-wiring for safety-data transmission reduces the amount of infrastructure required significantly because these networks already exist in most companies.

This paper explained how non-TSN-compatible devices can be connected to a TSN-network by using network-components like buscouplers, that take over the task of tagging/"marking" priority-traffic. In this case only these "tagging-components" and the switches have to be changed – not the wired infrastructure – to realize a real-time enabled industrial network, that is capable of hosting safety-related traffic together with non-priority traffic in one network.

The tests in this paper showed that the combination of these two technologies not only works, but also under extreme conditions with a big amount of background traffic. A "sensitivity"-threshold of max. 5ms between two heartbeats could be realized, even though under realistic circumstances this is not even necessary. The stopping time of a robot with industrial relevant size for example – like the Universal Robot UR10e – is around 20ms with just 33% of extension and 33% of maximum payload [13] – that is 400% longer than the heartbeat interval. In several papers like [14] the total reaction time of Ethernet-based safety systems (SFRT) is calculated by also considering the transmission delays of every single network-component. For a worst case scenario and a realistic view on a possible situation within a factory network these delays, even those of small, non-processing network components are set to be 10ms – so twice as long as the minimal heartbeat interval that could be achieved in this paper.

Summarizing all these facts, the performance of the system presented in this paper was higher than expected and proofs the concept. Even under the worst possible conditions within a network like broadcast-dummy messages on full bandwidth the TSN-infrastructure guaranteed the connection between safety-related components spread over two networks and realized the flawless execution of a real-time-dependent safety-application.

VIII. Acknowledgments

This work has been partially supported and funded by the Austrian Research Promotion Agency (FFG) via the "Austrian Competence Center for Digital Production" (CDP) under the contract number 854187.

References

[1] S. Brown, "Overview of IEC 61508. Design of electrical/electronic/programmable electronic safety-related systems," *Computing and Control Engineering Journal*, vol. 11, no. 1, pp. 6–12, 2000.

[2] EPSG (Ethernet POWERLINK Standardisation Group), *openSAFETY Safety Profile Specification (Working Draft Proposal 304)*, 1st ed. Fredersdorf/Germany: EPSG, 2017.

[3] R. Zurawski, *Industrial Communication Technology Handbook*, ser. Industrial Information Technology. CRC Press, 2017. [Online]. Available: https://books.google.at/books?id=ppzNBQAAQBAJ

[4] A. Ademaj and P. Loschmidt, "IEEE TSN (Time-Sensitive Networking): A Deterministic Ethernet Standard," TTTech Computertechnik AG, Vienna, Tech. Rep., 2015.

[5] A. Ademaj, P. Loschmidt, and W. Steiner, "TSN Overview," TTTech Computertechnik AG, Vienna, Tech. Rep., 2016.

[6] IEEE, *IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks*, 2018.

[7] S. S. Craciunas, R. S. Oliver, M. Chmelik, and W. Steiner, "Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, ser. RTNS '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 183–192. [Online]. Available: https://doi.org/10.1145/2997465.2997470

[8] D. Etz, T. Frühwirth, A. Ismail, and W. Kastner, "Simplifying functional safety communication in modular, heterogeneous production lines," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2018, pp. 1–4.

[9] V. Gavriluţ and P. Pop, "Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2018, pp. 1–4.

[10] M. Pahlevan, J. Schmeck, and R. Obermaisser, "Evaluation of TSN Dynamic Configuration Model for Safety-Critical Applications," in *2019 IEEE Intl Conf on Parallel Distributed Processing with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom)*, 2019, pp. 566–571.

[11] D. Etz, T. Frühwirth, and W. Kastner, "Self-Configuring Safety Networks," in *Kommunikation und Bildverarbeitung in der Automation*, J. Jasperneite and V. Lohweg, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020, pp. 232–245.

[12] PROFIBUS Nutzerorganisation e.V., "PROFIsafe – Safety Technology for PROFIBUS and PROFINET," Karlsruhe, Tech. Rep., 2007.

[13] Universal Robots, "Universal Robots. UR10e/CB3 User manual," pp. I–55 – I–58, 2015. [Online]. Available: http://www.universal-robots.com/media/8764/ur10_user_manual_en_global.pdf

[14] V. Pimentel and B. G. Nickerson, "A safety function response time model for wireless industrial control," *IECON Proceedings (Industrial Electronics Conference)*, pp. 3878–3884, 2014.